

O'REILLY®

Fourth  
Edition

# Raspberry Pi Cookbook

Software and Hardware  
Problems & Solutions



Simon Monk

# Raspberry Pi Cookbook

FOURTH EDITION

Software and Hardware  
Problems and Solutions

**Dr. Simon Monk**

**O'REILLY®**

Beijing • Boston • Farnham • Sebastopol • Tokyo

# Raspberry Pi Cookbook

by Simon Monk

Copyright © 2023 Simon Monk. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://oreilly.com>). For more information, contact our corporate/institutional sales department: 800-998-9938 or [corporate@oreilly.com](mailto:corporate@oreilly.com).

- Acquisitions Editor: Zan McQuade
- Development Editor: Jeff Bleiel
- Production Editor: Clare Laylock
- Copyeditor: Penelope Perkins
- Proofreader: Piper Editorial Consulting, LLC
- Indexer: Sue Klefstad
- Interior Designer: David Futato
- Cover Designer: Karen Montgomery
- Illustrator: Kate Dullea
- August 2014: First Edition
- June 2016: Second Edition
- October 2019: Third Edition
- December 2022: Fourth Edition

## Revision History for the Fourth Edition

- 2022-12-08: First Release

See <http://oreilly.com/catalog/errata.csp?isbn=9781098130923> for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Raspberry Pi Cookbook*, the cover image, and related trade dress are

trademarks of O'Reilly Media, Inc.

The views expressed in this work are those of the author, and do not represent the publisher's views. While the publisher and the author have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-098-13092-3

[LSI]

## **Dedication**

To my late mother Anne Kemp (1924–2022), whose good cheer and ability to laugh in the face of adversity was an example to us all

# Preface to the Fourth Edition

---

Launched in 2011, the Raspberry Pi has found a role both as a very low-cost Linux-based computer and as a platform for embedded computing. It has proven popular with educators and hobbyists alike.

As of this writing, more than 40 million Raspberry Pis have been sold. The Raspberry Pi 4 with an option of 8 GB of memory makes the Raspberry Pi more than powerful enough to use as a replacement for a desktop computer, and the Pi 400 with its built-in keyboard makes a very capable replacement for a desktop computer.

The availability of open source Linux software for internet browsing, email, office suites, and photo editing is set to make the Raspberry Pi even more popular.

Even the latest Raspberry Pi 4 and Pi 400 still includes the general-purpose input/output (GPIO) pins that allow the hobbyist to add their own electronic contraptions to the Raspberry Pi.

This edition has been thoroughly updated to encompass the new models of Raspberry Pi, as well as the many changes and improvements to its Raspberry Pi OS. In particular you will find new chapters on:

- Machine Learning
- Raspberry Pi Pico and Pico W

This book is designed so that you can read it linearly, as you would a regular book, or access recipes at random. You can search the table of contents or index for the recipe that you want and then jump directly to it. If the recipe requires you to know about other things, it will refer you to other recipes, rather like a cookbook might refer you to base sauces before showing you how to cook something fancier.

The world of Raspberry Pi moves quickly. With a large, active community, new interface boards and software libraries are being developed all the

time. In addition to examples that use specific interface boards or software, the book also covers basic principles so that you can have a better understanding of how to use new technologies that come along as the Raspberry Pi ecosystem develops.

As you would expect, a large body of code (mostly Python programs) accompanies the book. These programs are all open source and available on [GitHub](#). For most of the software-based recipes, all you need is a Raspberry Pi. I recommend a Raspberry Pi 3 or 4 model B. When it comes to recipes that involve making your own hardware to interface with the Raspberry Pi, I have tried to make good use of ready-made modules as well as solderless breadboard and jumper wires to avoid the need for soldering.

If you want to make breadboard-based projects more durable, I suggest using prototyping boards with the same layout as a half-size breadboard, such as those sold by Adafruit and elsewhere, so that the design can easily be transferred to a soldered solution.

## Using This Book

The cookbook style of this book means that it is not a book that you must read in order from front to back. The book is made up of individual recipes grouped into chapters. Where a recipe needs you to have prior knowledge of some other topic, the recipe will send you off to another recipe for that topic.

You'll probably find that you jump around from recipe to recipe as you try to get your Raspberry Pi project to do what you want.

I have mapped out a few paths through the book that I think would be useful to different types of readers:

Complete Raspberry Pi beginner

Read most of Chapters [1](#), [2](#), and [3](#)—in particular, start with Recipes [1.1](#), [1.2](#), and [1.4](#)—and then wander at will.

Python learner

If you want to use your Raspberry Pi to learn how to program in Python, work your way through Chapters 4 to 7. You will probably find that you need to jump off to various recipes in earlier chapters.

### Electronics hobbyist

If you don't already have them, you'll need to pick up some Python skills by working through Chapters 4 to 7, and then work through Chapters 8 and 9 before picking out some interesting recipes in the later chapters to start making yourself some Raspberry Pi electronics projects.

## Conventions Used in This Book

The following typographical conventions are used in this book:

### *Italic*

Indicates new terms, URLs, email addresses, filenames, and file extensions.

### Constant width

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

### **Constant width bold**

Shows commands or other text that should be typed literally by the user.

### *Constant width italic*

Shows text that should be replaced with user-supplied values or by values determined by context.

### **TIP**

This icon signifies a tip, suggestion, or general note.



## WARNING

This icon indicates a warning or caution.

## NOTE

This icon points you to the related video for that section.

## Using Code Examples

Supplemental material (code examples, etc.) is available for download at [https://github.com/simonmonk/raspberrypi\\_cookbook\\_ed4](https://github.com/simonmonk/raspberrypi_cookbook_ed4).

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: “*Raspberry Pi Cookbook*, Fourth Edition, by Simon Monk (O'Reilly). Copyright 2023 Simon Monk, 978-1-098-13092-3.”

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at [permissions@oreilly.com](mailto:permissions@oreilly.com).

## O'Reilly Online Learning

## NOTE

For more than 40 years, *O'Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, conferences, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, please visit <http://oreilly.com>.

## How to Contact Us

Please address comments and questions concerning this book to the publisher:

- O'Reilly Media, Inc.
- 1005 Gravenstein Highway North
- Sebastopol, CA 95472
- 800-998-9938 (in the United States or Canada)
- 707-829-0515 (international or local)
- 707-829-0104 (fax)

We have a web page for this book where we list errata, examples, and any additional information. You can access this page at <https://oreil.ly/raspberry-pi-cookbook-4e>.

Email [bookquestions@oreilly.com](mailto:bookquestions@oreilly.com) to comment or ask technical questions about this book.

For news and information about our books and courses, visit <https://oreilly.com>.

Find us on LinkedIn: <https://linkedin.com/company/oreilly-media>.

Follow us on Twitter: <http://twitter.com/oreillymedia>.

Watch us on YouTube: <http://www.youtube.com/oreillymedia>.

## **Acknowledgments**

As always, I thank my wife, Linda, for her patience and support.

I also thank the technical reviewers Ian Huntley, Mike Bassett, Kevin McAleer, and Matthew Monk for their excellent help and suggestions that have without a doubt contributed greatly to this book.

Thanks also to the very helpful and well-organized Jeff Bleiel and all the O'Reilly team and, of course, Penelope Perkins for her keen eye.

# Chapter 1. Setup and Management

---

## 1.0 Introduction

When you buy a Raspberry Pi, you are essentially buying an assembled printed circuit board, or in the case of a Raspberry Pi 400, a circuit board in a keyboard case. For a fully functioning system, you are going to need at least a suitable power supply, operating system on microSD card, and mouse.

The recipes in this chapter are concerned with getting your Raspberry Pi set up and ready for use.

Because the Raspberry Pi uses standard USB and Bluetooth keyboards and mice, most of the setup is pretty straightforward, so you'll concentrate only on those tasks that are specific to the Raspberry Pi.

## 1.1 Selecting a Model of Raspberry Pi

### Problem

Many models of Raspberry Pi are available, and you are not sure which one to use.

### Solution

The decision as to which Raspberry Pi model to use depends very much on what you plan to do with it. [Table 1-1](#) lists some uses and my model recommendations.

*Table 1-1. Selecting a model of Raspberry Pi*

---

Usage	Suggested model	Notes
Desktop computer replacement	Raspberry Pi 400 or Raspberry Pi 4 model B (4 GB)	You will need the 4 GB of memory if you are web browsing. The Pi 400 offers the convenience being built into a keyboard case.
Electronics experimentation	Raspberry Pi 2 or 3 model B	Reasonably up-to-date hardware will minimize software problems. No need for more performance.
Computer vision	Raspberry Pi 4 model B (4 GB)	Maximum performance required.
Home automation	Raspberry Pi 2 or 3 model B	Low-power consumption and more than enough power.
Media center	Raspberry Pi 3 or 4	For video performance.
Electronic display board	Any model	A model with WiFi, advantageous for remote access.
Embedded wireless electronics project	Raspberry Pi Zero 2 W or Pico W	Low cost and WiFi enabled for IoT (Internet of Things) and other wireless projects.
Embedded electronics project	Pico	Very low cost and little in common with most Raspberry Pis other than the name. See <a href="#">Chapter 19</a> for more information.

If you want a good general-purpose Raspberry Pi, I recommend a Raspberry Pi 4 model B. With four times as much memory as the original Raspberry Pi and a quad-core processor, it will cope with most tasks much better than the Pi Zero, but it doesn't get as hot or use as much power as the Raspberry Pi 4. The Raspberry Pi 3 model B+ also has the great advantage of having WiFi and Bluetooth built in, so there's no need for an extra USB WiFi adapter or Bluetooth hardware.

## THE RASPBERRY PI 4 MODEL B

As of this writing, the Raspberry Pi 4 model B (Figure 1-1) is the latest *standard* Raspberry Pi.

The new model has also, for the first time, allowed users a choice of memory sizes (1 GB, 2 GB, 4 GB, or 8 GB), with the price reflecting your memory choice.

One of the most significant changes is that the micro-USB socket that supplied power to earlier versions has been dropped in favor of the USB-C connector. Also, the single full-size HDMI connector of the earlier versions has been replaced by a pair of micro-HDMI connectors—so you'll need a special HDMI lead or an adapter. And yes, you can connect two monitors at the same time.

Under the hood, this Raspberry Pi is much faster than its predecessors (especially if you go for the 4 GB or 8 GB memory versions). In fact, some benchmarks suggest it is three to four times faster than any previous Raspberry Pi. This comes at the cost of the main chip on the board operating much hotter than in earlier versions—hot enough, in fact, to hurt.

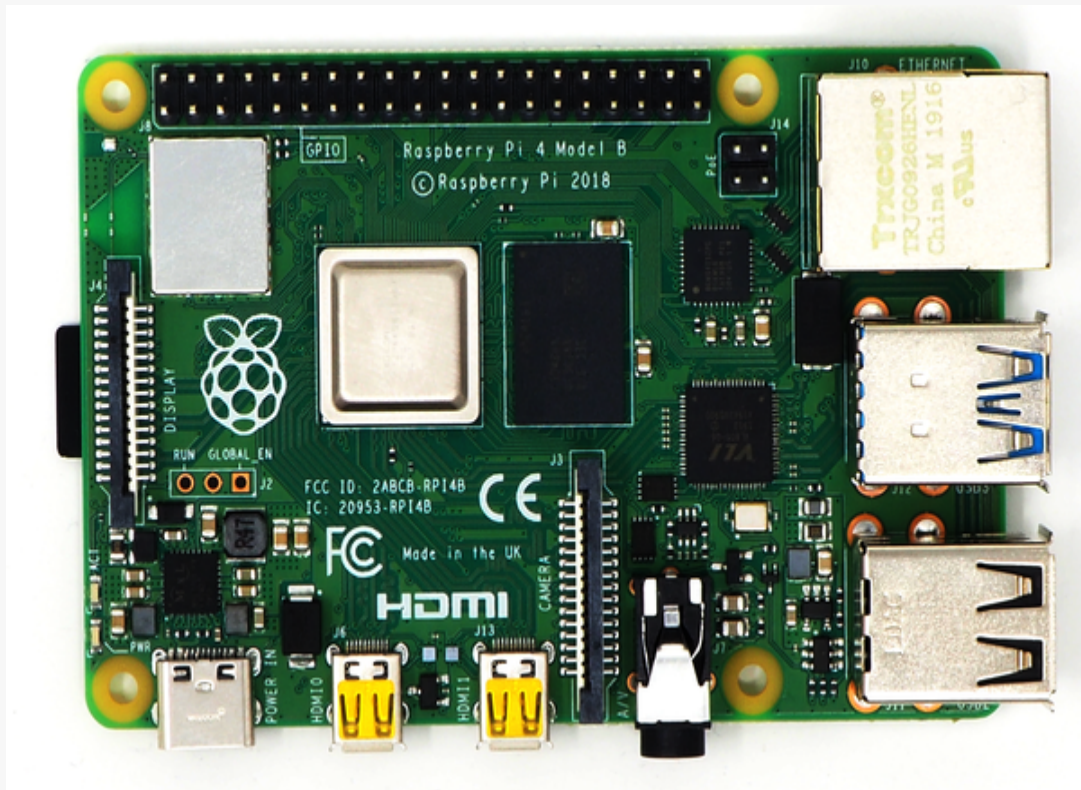


Figure 1-1. The Raspberry Pi 4 model B

If, on the other hand, you are embedding a Raspberry Pi in a project for a single purpose, using a compact Pi Zero W and saving a few dollars might

well be an option.

### THE RASPBERRY PI 400

The Raspberry Pi makes a pretty good desktop computer replacement, and no Raspberry Pi is better suited to this than the Raspberry Pi 400 (Figure 1-2).



Figure 1-2. The Raspberry Pi 400

The Raspberry Pi 400 is essentially the same hardware as the Raspberry Pi 4, but built into a keyboard case, reminiscent of the home computers of the 1990s. The HDMI and USB ports are all accessible at the back of the Pi 400. The GPIO pins are accessible at the back of the computer, but are not as easy to access as a regular Raspberry Pi, so if you are buying a Raspberry Pi to learn about electronics, a regular Pi is probably a better choice. However, if you just want a desktop replacement, then a Pi 400 is a really good choice.

## Discussion

Figure 1-3 shows the Pi Zero W, the Raspberry Pi 3 B, and the Raspberry Pi 4.

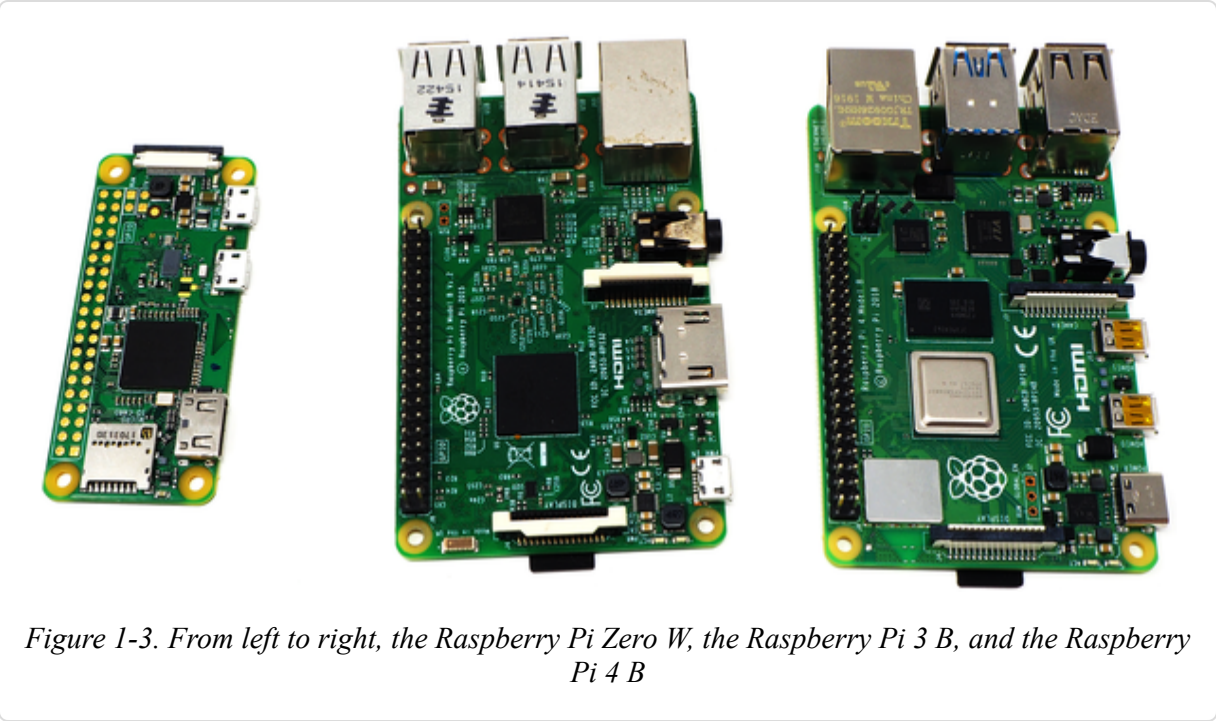


Figure 1-3. From left to right, the Raspberry Pi Zero W, the Raspberry Pi 3 B, and the Raspberry Pi 4 B

As you can see from **Figure 1-3**, the Pi Zero W is roughly half the size of the Pi 3 B or Pi 4 B, and it has a single micro-USB socket for communication and a second one for power. The Pi Zero W also saves space by using a mini-HDMI socket and micro-USB on-the-go socket. If you want to connect a keyboard, monitor, and mouse to a Pi Zero, you'll need adapters for both the USB and the HDMI ports before you can connect standard peripherals. The Raspberry Pi A+ is larger than the Pi Zero and has full-size USB and HDMI ports.

**Table 1-2** summarizes the differences between all the Raspberry Pi models to date, with the most recently released models toward the top.

Table 1-2. Raspberry Pi models

Model	RAM	Processor (cores * clock)	USB sockets	Ethernet port	Notes
400	4 GB	4 * 1.8 GHz	4 (2 x USB3)	yes	Built into a keyboard
4 B	1/2/4/8 GB	4 * 1.5 GHz	4 (2 x USB3)	yes	2 x micro-HDMI video



Model	RAM	Processor (cores * clock)	USB sockets	Ethernet port	Notes
Compute 4	1/2/4/8 GB	4 * 1.5 GHz	no	no	For embedding in products (see sidebar)
3 A+	512 MB	4 * 1.4 GHz	1	no	WiFi and Bluetooth
3 B+	1 GB	4 * 1.4 GHz	4	yes	WiFi and Bluetooth
3 B	1 GB	4 * 1.2 GHz	4	yes	WiFi and Bluetooth
Zero 2 W	512 MB	4 * 1 GHz	1 (micro)	no	WiFi and Bluetooth
Zero W	512 MB	1 * 1 GHz	1 (micro)	no	WiFi and Bluetooth
Zero	512 MB	1 * 1 GHz	1 (micro)	no	Low cost
2 B	1 GB	4 * 900 MHz	4	yes	
A+	256 MB	1 * 700 MHz	1	no	
B+	512 MB	1 * 700 MHz	4	yes	Discontinued
A	256 MB	1 * 700 MHz	1	no	Discontinued
B rev2	512 MB	1 * 700 MHz	2	yes	Discontinued
B rev1	256 MB	1 * 700 MHz	2	yes	Discontinued

If you have one of the older or discontinued Raspberry Pi models, it is still useful. Those models do not have quite the performance of the latest Raspberry Pi 4, but for many situations, that does not matter.

If you are buying a new Raspberry Pi, I consider the best choice for use as a general-purpose computer to be the Raspberry Pi 4 or 400. If you don't need WiFi or want a smaller device, also consider the 3 B, 2 B, or Zero W.

## THE RASPBERRY PI COMPUTE 4

The Raspberry Pi is such a useful device that it has found its way into many commercial products. Sometimes this is a little wasteful, as the product may not have need of all the connectors and other features of, say, a Raspberry Pi 4.

The Raspberry Pi Compute 4 (and its predecessors) provides a neat module (Figure 1-4) that has connectors on the underside designed to mate to a carrier board, as a really easy way for a product to incorporate a Linux computer. Because the module and its associated WiFi and Bluetooth are already certified as CE and FCC compliant, making your product comply with relevant standards is a lot easier.



Figure 1-4. The Raspberry Pi Compute Module 4

### See Also

For more information on the Raspberry Pi models, see [https://oreil.ly/oY-A\\_](https://oreil.ly/oY-A_).

Take a look at the [Raspberry Pi Compute Modules](#).

The low cost of the Pi Zero and Pi Zero W models makes them ideal for embedding in electronics projects without worrying about the expense. See [Recipe 10.18](#).

## 1.2 Connecting the System

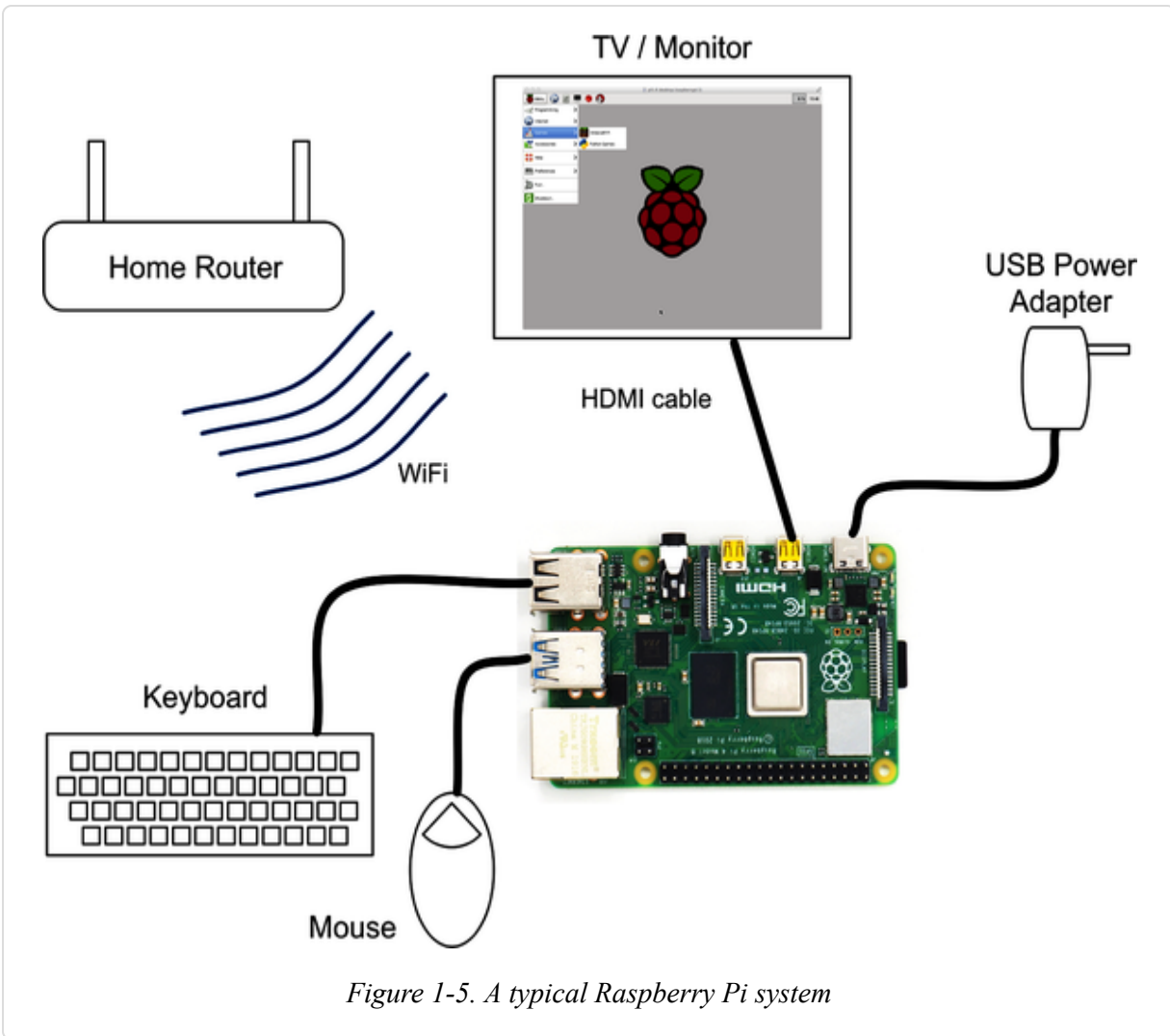
### Problem

You have everything that you need for your Raspberry Pi, and you want to connect it all.

### Solution

Unless you are embedding your Raspberry Pi in a project or using it as a media center, you need to attach a keyboard (unless you are using a Pi 400), a mouse, and a monitor.

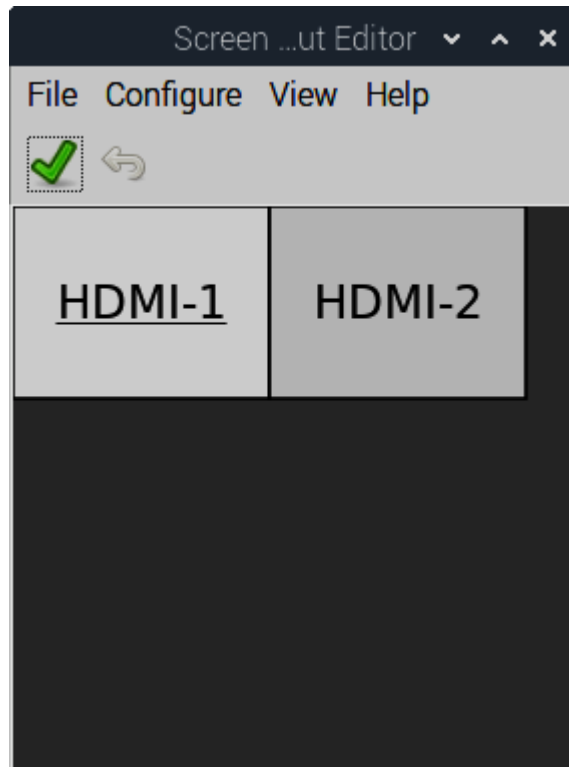
[Figure 1-5](#) shows a typical Raspberry Pi system. If you have a Raspberry Pi 4, you could (if you really wanted) connect a second monitor. However, if you have just one monitor, connect it to the micro-HDMI connector closest to the USB-C power connector.



## Discussion

The Raspberry Pi is perfectly happy with pretty much any USB keyboard and mouse, wired or wireless.

The Raspberry Pi 4 lets you connect two monitors to your system at the same time. When you do so, you'll be able to move your mouse cursor between screens, but Raspberry Pi OS will need to know where the screens are relative to each other. To enable this, open the Raspberry Menu (the one with the Raspberry Pi icon), go to the Preferences section, and open the Screen Configuration tool (Figure 1-6).



*Figure 1-6. Arranging multiple screens*

You can drag the two boxes labeled HDMI-1 and HDMI-2 around to represent the physical position of the two monitors. In [Figure 1-6](#), the monitors are side by side, with the monitor connected to HDMI-1 on the left.

If you have an older Raspberry Pi or a model A or A+ and run out of USB sockets, you will also need a USB hub.

## **See Also**

Check out the [official Raspberry Pi Quick Start Guide](#).

## **1.3 Enclosing a Raspberry Pi**

### **Problem**

You need an enclosure for your Raspberry Pi.

## **Solution**

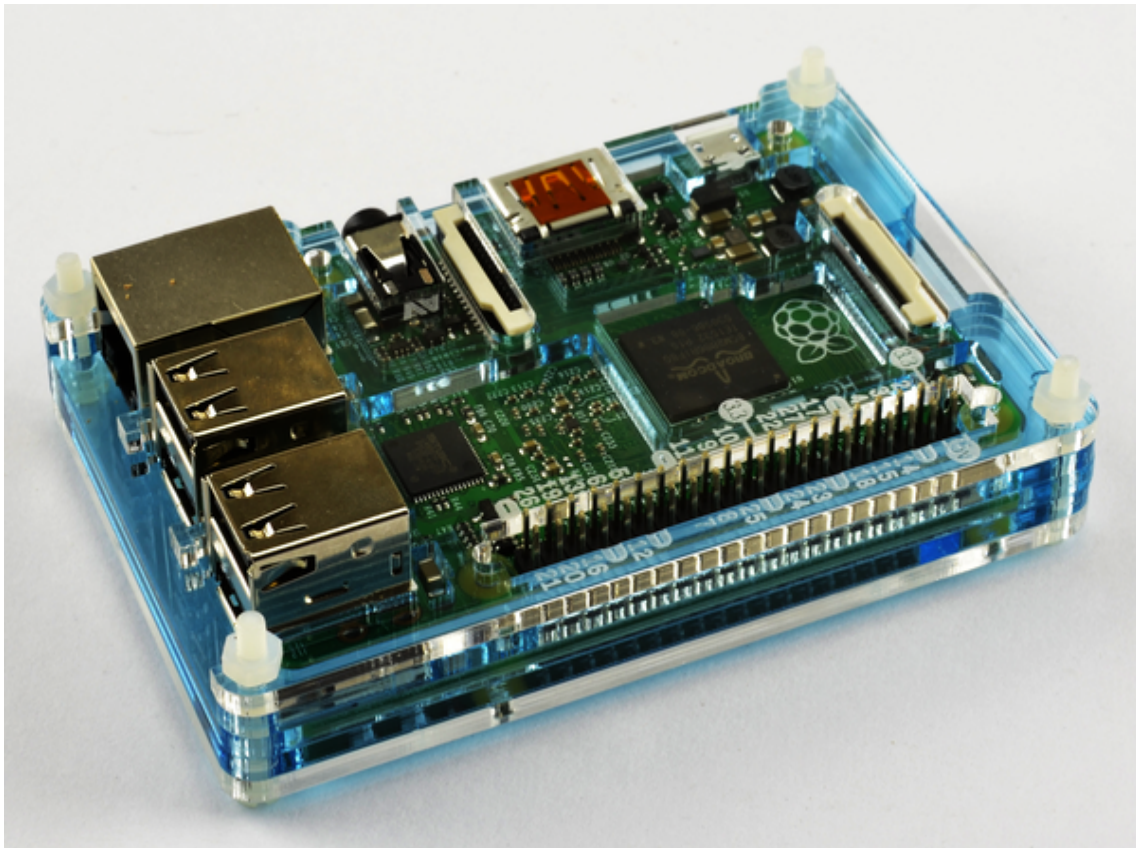
Apart from the Raspberry Pi 400, Raspberry Pis do not come with an enclosure unless you buy one as part of a kit. This makes it a little vulnerable, given that bare connections are on the underside of the circuit board that could easily be short-circuited if the Raspberry Pi is placed on something metal.

It is a good idea to buy some protection for your Raspberry Pi in the form of a case. If you intend to use the Raspberry Pi's general-purpose input/output (GPIO) pins (the pins that allow you to connect to external electronics), the Pibow Coupé shown in [Figure 1-7](#) is a beautiful and practical design that is available for both the Raspberry Pi 4 and earlier versions.

## **Discussion**

A vast array of case styles are available to choose from, including the following:

- Simple, two-part, click-together plastic boxes
- VESA mountable boxes (for attaching to the back of a monitor or TV)
- LEGO-compatible boxes
- 3D-printed box designs
- Laser-cut acrylic designs
- Designs that act as a large heat sink for cooling
- Designs with a built-in fan
- DIN-rail designs for labs and workshops



*Figure 1-7. A Raspberry Pi 2 in a Pibow Coupé*

The case you buy is very much a matter of personal taste. However, consider the following:

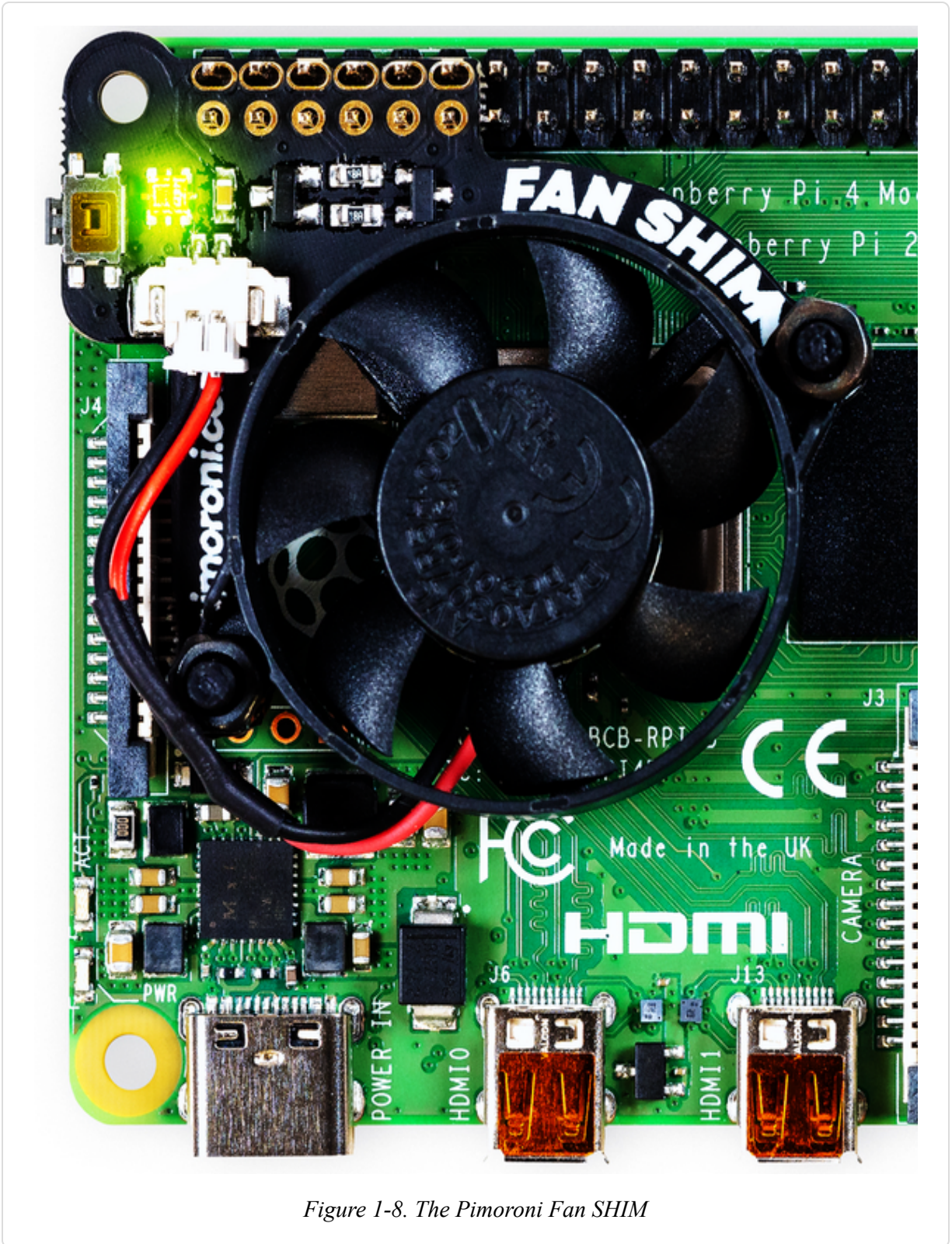
- Do you need to have access to the GPIO connector? This is important if you plan to attach external electronics to your Raspberry Pi.
- Is the case well ventilated? This is important if you plan to overclock your Raspberry Pi ([Recipe 1.13](#)) or run it hard playing videos or games, because these will all generate more heat.
- Finally, make sure you get one that fits your model of Raspberry Pi.

If you have access to a 3D printer, you can also print your own case. Search for Raspberry Pi on [Thingiverse](#) or [MyMiniFactory](#), and you'll find lots of designs.

You will also find kits that have tiny self-adhesive heat sinks to attach to the chips on the Raspberry Pi. These can be of some use if you are demanding a lot of your Raspberry Pi, say, by playing a lot of videos, but generally they are the equivalent of “go-faster” stripes on a car.

If you have a Raspberry Pi 4, you can reduce the temperature by fitting a small fan such as the Pimoroni Fan SHIM, as shown in [Figure 1-8](#).





*Figure 1-8. The Pimoroni Fan SHIM*

**See Also**

You will also find many styles of cases at other Raspberry Pi suppliers and on eBay.

## 1.4 Selecting a Power Supply

### Problem

You need to select a power supply for your Raspberry Pi.

### Solution

The basic electrical specification for a power supply suitable for a Raspberry Pi is that it provides a regulated 5V DC (direct current).

The amount of current that the power supply must be capable of providing depends both on the model of Raspberry Pi and on the peripherals attached to it. It is worth getting a power supply that can easily cope with the Raspberry Pi, and you should consider 1A to be a minimum for any model of Raspberry Pi.

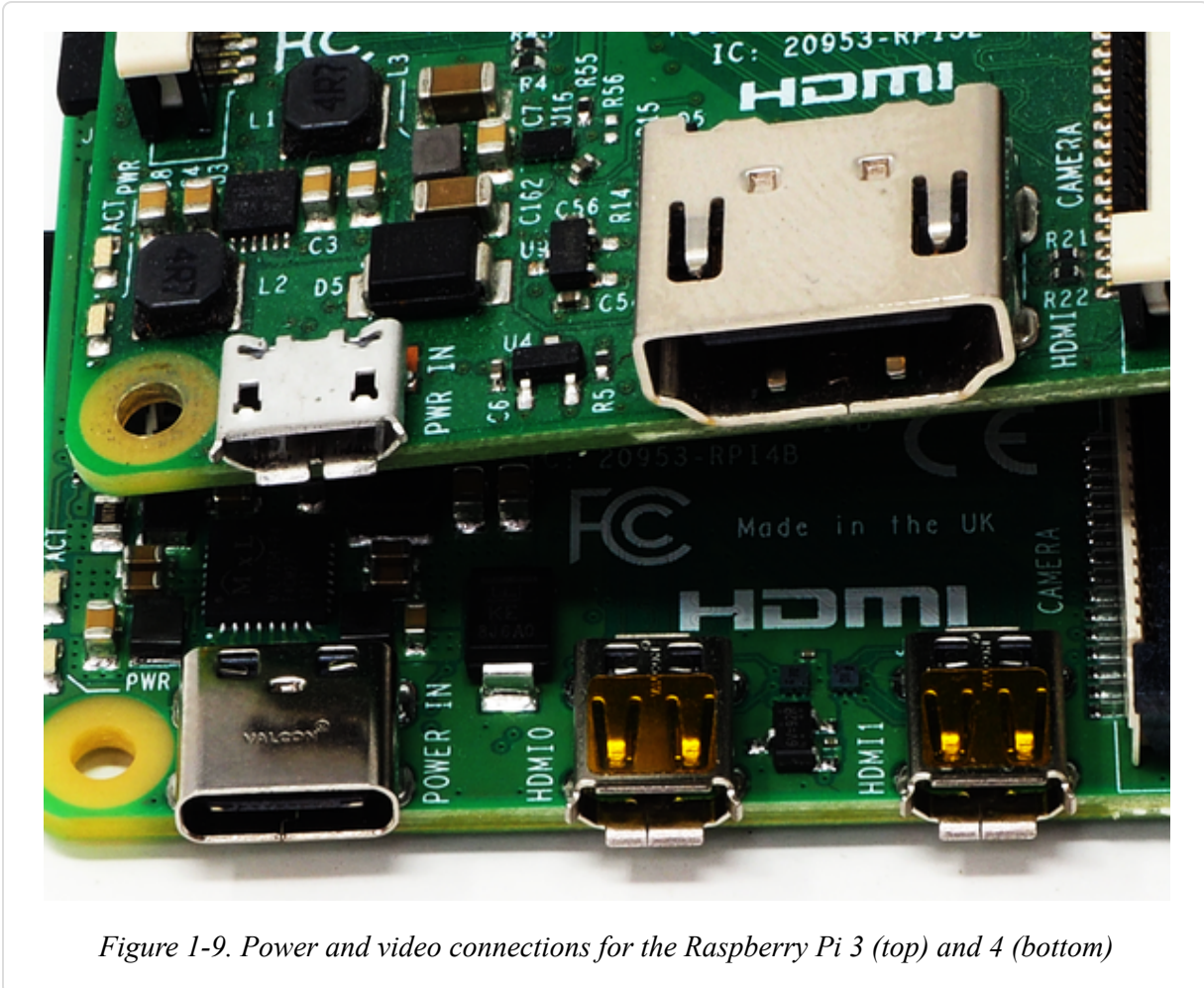
If you buy your power supply from the same place that you buy the Raspberry Pi, the seller should be able to tell you whether it will work with the Raspberry Pi.

The Raspberry Pi 4 should be used with a 3A power supply. This is in part because its greater processing power than earlier models requires more electrical power, but also because its two USB3 ports are able to supply up to 1.2A to high-power USB peripherals such as external USB drives.

If you are going to be using WiFi or USB peripherals that require significant amounts of power with pre-Raspberry Pi 4 models, you should get a power supply capable of 1.5A or even 2A. Also beware of very low-cost power supplies that might not provide an accurate or reliable 5V.

### Discussion

The Raspberry Pi 4 is the first Raspberry Pi to use the more modern USB-C connector. Unlike the micro-USB connector used on earlier boards, this connector is reversible ([Figure 1-9](#)).



*Figure 1-9. Power and video connections for the Raspberry Pi 3 (top) and 4 (bottom)*

In [Figure 1-9](#), you can see the USB-C power connector of a Raspberry Pi 4 below the micro-USB connector of a Raspberry Pi 3. As an aside, you can also see the pair of micro-HDMI video ports that replace the single full-size HDMI connector.

Whether your Raspberry Pi uses a USB-C connector or a micro-USB connector, the power supply and connectors are actually the same as those found in many smartphone chargers. If they terminate in a micro-USB plug, they are almost certainly 5V (but check). The only question, then, is whether they can supply enough current.

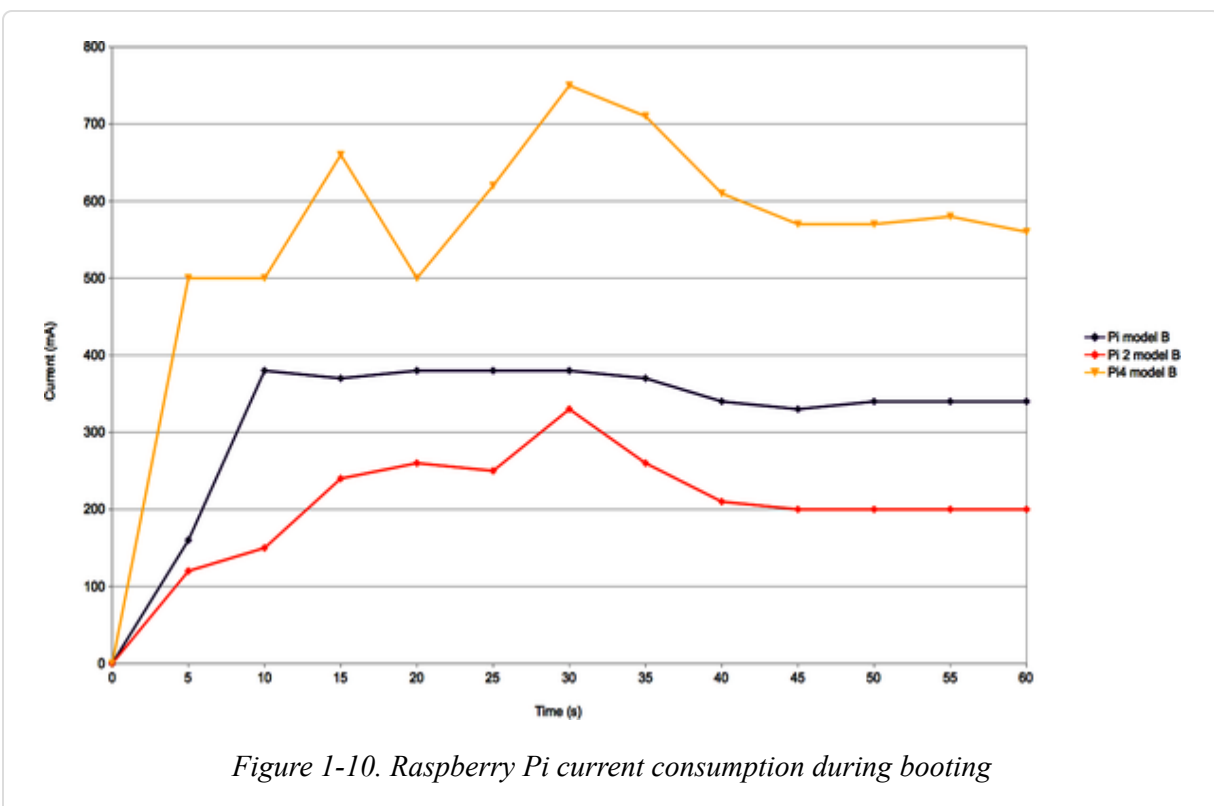
If they can't, a few bad things can happen:

- They might become hot and be a potential fire risk.
- They might simply fail.
- At times of high load (say, when the Pi is using WiFi or playing video), the voltage might dip and the Raspberry Pi might reset itself.

If you are using a Raspberry Pi 3 or earlier, look for a power supply that says it can supply 1A or more. If it specifies a number of watts (W) rather than amps (A), divide the number of watts by 5 to get the number of amps. So a 5V 10W power supply can supply 2A (2,000mA).

Using a power supply with, say, a maximum current of 2A will not use any more electricity than a 700mA power supply. The Raspberry Pi takes only as much current as it needs.

In [Figure 1-10](#), I measure the current taken by a Raspberry Pi model B and compare it with a Raspberry Pi 2 model B and a Raspberry Pi 4.



The newer Raspberry Pis (starting with the A+ and all the way through to the Pi 4) are far more power-efficient than the original Raspberry Pi 1 models, but when the processor is fully occupied and has a lot of peripherals attached, they can still reach similar current requirements, and, in the case of the Raspberry Pi 4, quite a lot more.

As you can see in [Figure 1-10](#), if your Raspberry Pi is going to be on all the time, a Raspberry Pi 2 will run cooler and use a lot less power than the newest Raspberry Pi 4.

In [Figure 1-10](#), you can see that the current rarely exceeds 700mA. However, the processor isn't really doing very much here. Were you to start playing HD video, the current would increase considerably. When it comes to power supplies, it's always better to have something in reserve.

## See Also

You can buy an [uninterruptible power supply \(UPS\)](#) for the Raspberry Pi. This ensures that the Pi can keep running for 10–30 minutes in the event of a power failure.

The Raspberry Pi has no on/off switch, but you can buy a [module that will turn off the power](#) when the Raspberry Pi shuts down.

## 1.5 Selecting an Operating System

### Problem

There are several Raspberry Pi operating systems, and you are not sure which one to use.

### Solution

The answer to this question depends on what you intend to do with your Raspberry Pi.

For general use as a computer or for using in electronic projects, you should use Raspberry Pi OS, the standard and official distribution for the Raspberry Pi.

If you plan to use your Raspberry Pi as a media center, a number of distributions (versions of Linux) are available specifically for that purpose (see [Recipe 4.1](#)).

In this book, we use Raspberry Pi OS almost exclusively, although most of the recipes will work with any Debian-based Linux distribution.

## Discussion

If you are interested in trying out some different distributions, you can purchase some microSD cards, which are not expensive, and copy the various distributions onto them. If you do this, it is a good idea to keep your files that you don't want to lose on a USB flash drive plugged into your Raspberry Pi.

Note that if you are using one of the upcoming recipes to write your own SD card, you need to have a computer that has an SD card slot (and an SD-to-microSD adapter), or you can buy an inexpensive USB SD card reader.

## See Also

Check out the [official list of Raspberry Pi distributions](#).

# 1.6 Installing an Operating System Using Raspberry Pi Imager

## Problem

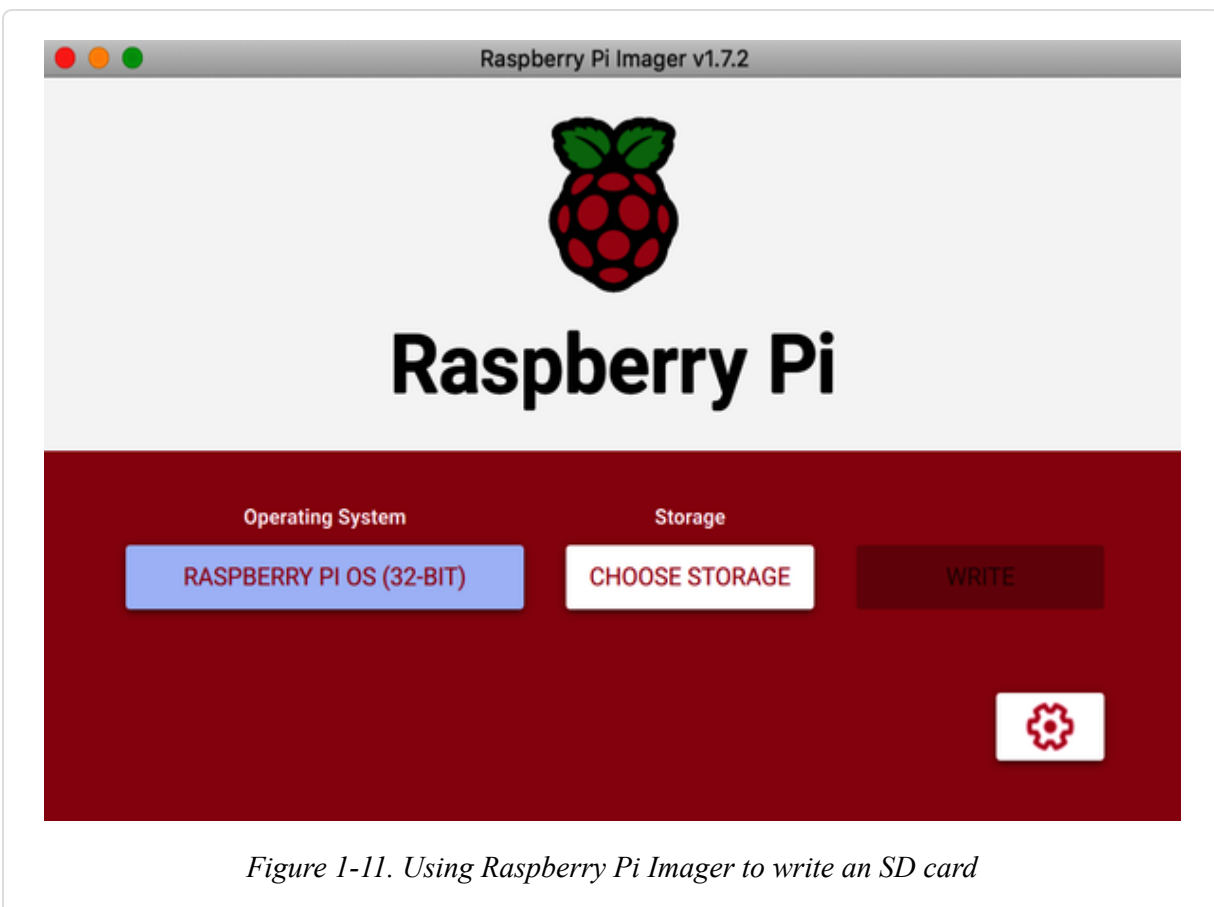
You want to put the operating system for your Raspberry Pi directly onto a microSD card ready for use by your Pi.

## Solution

Before you can use your Raspberry Pi, you must prepare a microSD card with the Raspberry Pi OS operating system by writing a disk image onto the microSD card.

The process of writing the disk image onto the microSD card is as follows:

1. Using a Mac, Windows, or Linux computer (not your Raspberry Pi), download the [Raspberry Pi Imager](#).
2. Insert the microSD card into your computer. It's also a good idea to disconnect any other removable media so that you don't accidentally overwrite the wrong device.
3. Start the Raspberry Pi Imager ([Figure 1-11](#)).
4. Select the Operating System as Raspberry Pi OS (32-bit) and the SD card.
5. Click Write and wait while the image file is copied onto the removable media.



*Figure 1-11. Using Raspberry Pi Imager to write an SD card*

After the SD card or other removable media is prepared, you can plug it in to your Raspberry Pi, and when the Raspberry Pi is powered up, it will boot into whatever operating system distribution you installed.

## **Discussion**

Vendors of hardware sometimes offer their own disk image that has support for their hardware built into it. It's best to avoid using such images because doing so means that you will not get all the benefits of using a standard Raspberry Pi OS distribution and all the pre-installed software. It also means that if you have a problem with a piece of software, it will be a lot more difficult to find support because you are using a nonstandard distribution.

The Raspberry Pi 4 and 400 hardware does have a 64-bit processor and a 64-bit version of the operating system is available, but at the time of writing using the default Raspberry Pi OS 32-bit option is much more stable.



## MICROSD CARDS

Not all microSD cards are created equal, and the performance of your Raspberry Pi will be better with a better card. So look for a card specified as being “class 10.” Raspberry Pi OS includes a utility (Figure 1-12) to test your microSD card. You can find it by selecting Accessories, then Raspberry Pi Diagnostics from the Raspberry Menu.

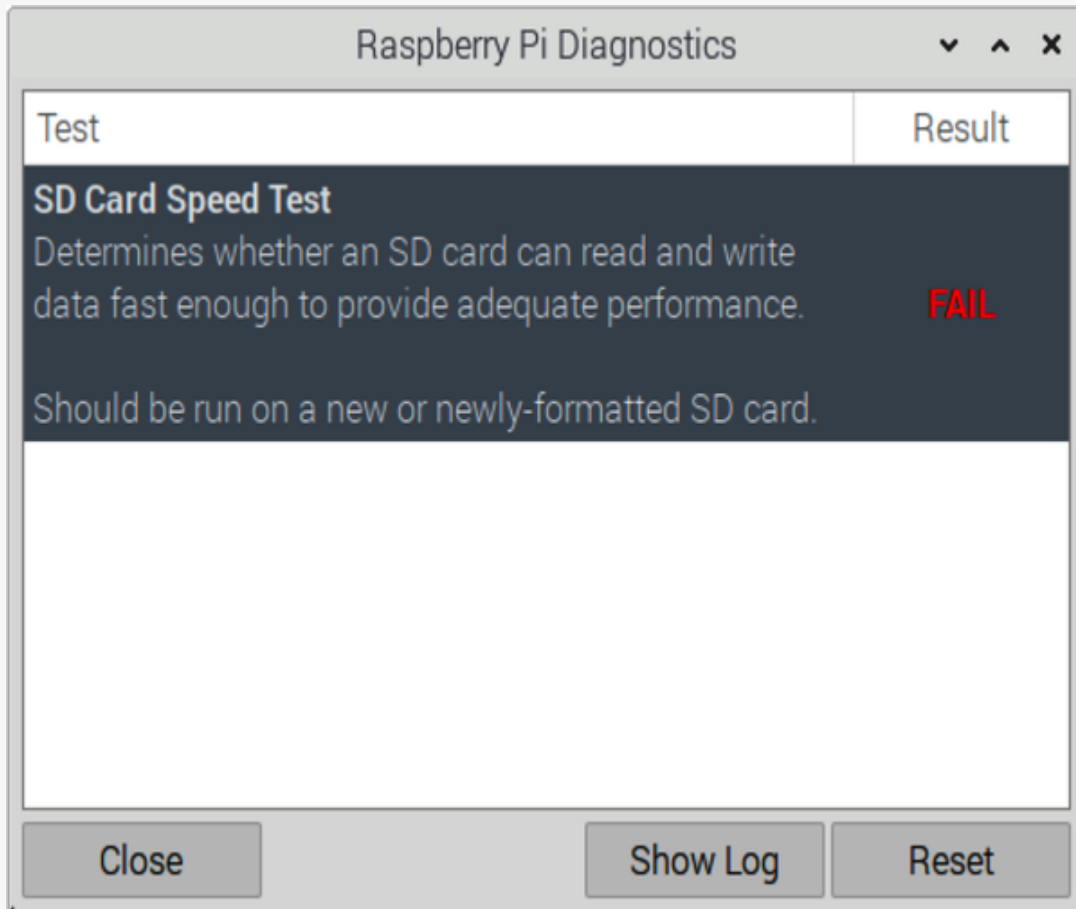


Figure 1-12. Using the Raspberry Pi Diagnostics tool to test your SD card

When it comes to capacity, you should look for cards of at least 16 GB, and, really, given the small difference in price, a 32 GB card is the better choice, as it will give you plenty of room for expansion.

## See Also

The Raspberry Pi guide to installing Raspberry Pi OS can be found at <https://www.raspberrypi.com/software>.

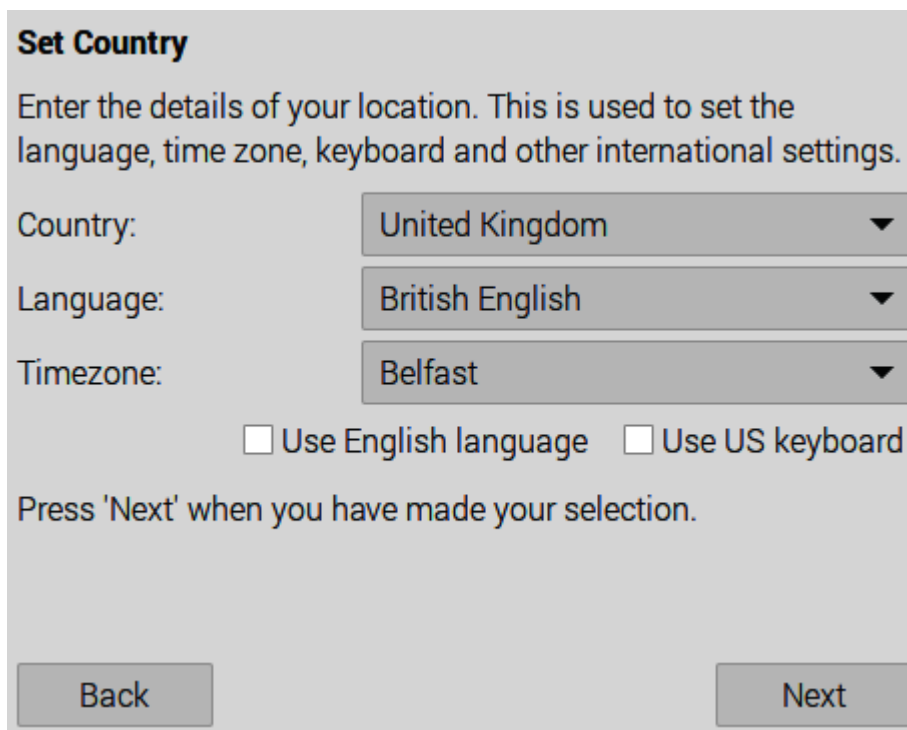
## 1.7 Booting Up Your Raspberry Pi for the First Time

### Problem

You've set up a microSD card, and you want to know how to set up your Raspberry Pi.

### Solution

The first time that you boot up your Raspberry Pi (as shown in [Figure 1-13](#)), you will be taken through some setup questions.



The screenshot shows a configuration window titled "Set Country". The window has a light gray background and contains the following elements:

- Title:** Set Country
- Instruction:** Enter the details of your location. This is used to set the language, time zone, keyboard and other international settings.
- Country:** A dropdown menu with "United Kingdom" selected.
- Language:** A dropdown menu with "British English" selected.
- Timezone:** A dropdown menu with "Belfast" selected.
- Options:** Two checkboxes:  Use English language and  Use US keyboard.
- Instruction:** Press 'Next' when you have made your selection.
- Buttons:** "Back" and "Next" buttons at the bottom.

*Figure 1-13. Configuring Raspberry Pi after installation*

Clicking Next prompts you to create a new user account ([Figure 1-14](#)).

**Create User**

You need to create a user account to log in to your Raspberry Pi.

The username can only contain lower-case letters, digits and hyphens, and must start with a letter.

Enter username:

Enter password:

Confirm password:

Hide characters

Press 'Next' to create your account.

*Figure 1-14. Creating a user account*

Prior to April 2022 this step was not part of the setup procedure, as the username of “pi” was automatically created for you. As a result many tutorials and books assume that your home directory is `/home/pi`. Unless you have strong feelings about your username, I would recommend sticking to the username “pi” but create a strong password (not the default of “raspberry”).

After you are up and running, the first thing you should do is connect your Raspberry Pi to the internet (Recipes 2.1 and 2.5), because next you’ll be asked to connect to a WiFi network to check for updates. Updating requires an internet connection, so it won’t work unless you have connected to your network. If you are connected (either by WiFi or Ethernet), it’s a good idea to check for updates now. If not, you can always check later using [Recipe 3.40](#).

## Discussion

It is a good idea to set the right time zone. If you don't, the Raspberry Pi will show the incorrect time because it gets its time from an internet time server.

## See Also

The Raspberry Pi guide to installing Raspberry Pi OS can be found at <https://www.raspberrypi.com/software>.

# 1.8 Setting Up a Headless Raspberry Pi

## Problem

You want to use a Raspberry Pi without having to connect a keyboard, mouse, and monitor to it.

## Solution

Use the settings option on the Raspberry Pi Imager to start your Raspberry Pi with network credentials and enable SSH ([Recipe 2.7](#)) so that you can connect to the Raspberry Pi from another computer.

After you have selected the operating system in the Raspberry Pi Imager, a cog settings icon will appear. When you click on this, you'll be presented with a list of settings ([Figure 1-15](#)) that you can use to preconfigure your Raspberry Pi so that other computers on your network can connect to it.

To be able to access the Raspberry Pi remotely, as a minimum you must:

- Enable SSH (Secure Shell)
- Set the username and password
- Configure the wireless LAN with the name of your network (SSID, or service set identifier) and password

It's also not a bad idea to set the hostname, especially if you are going to have more than one Raspberry Pi on your network and you want to tell which is which.

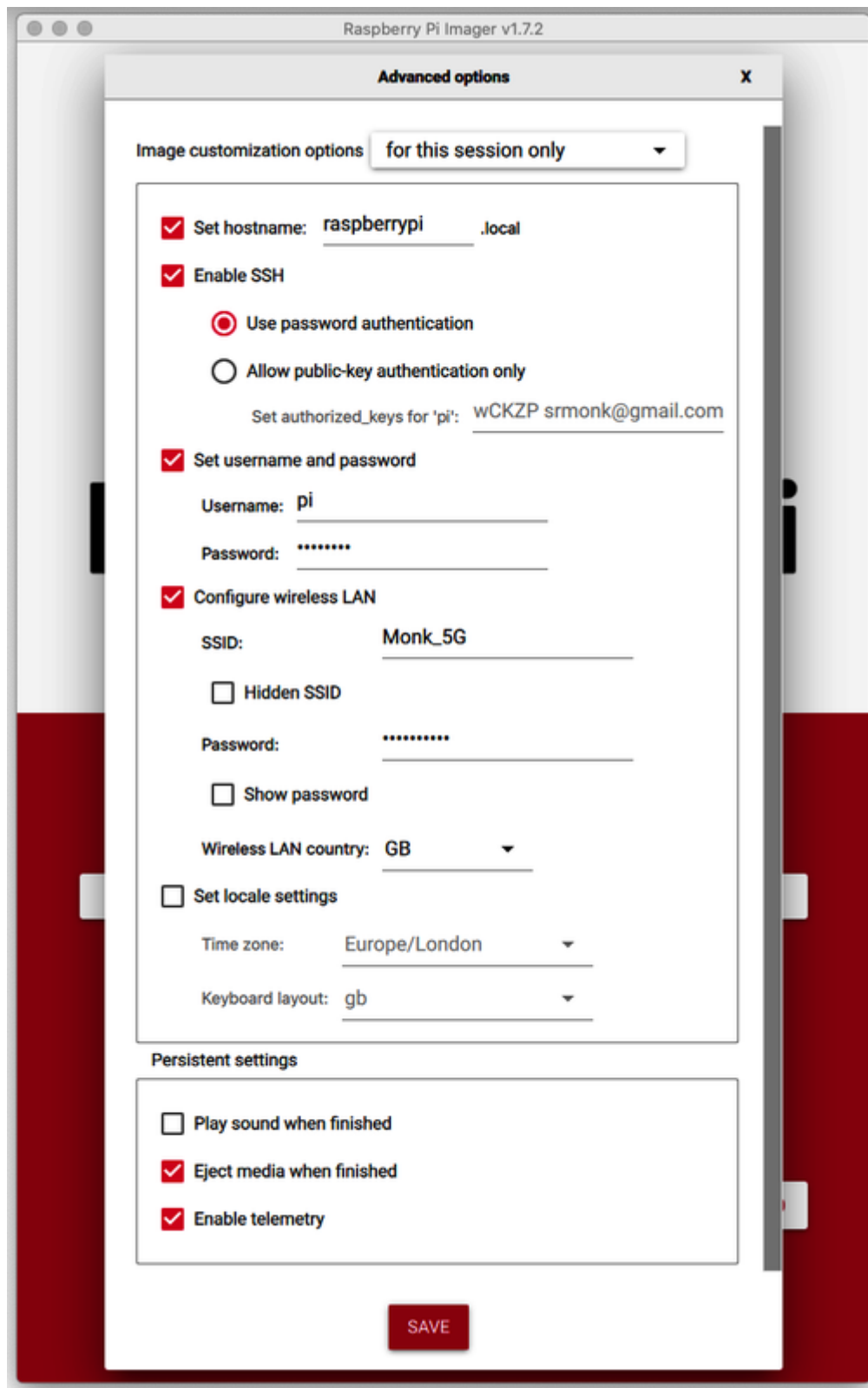


Figure 1-15. Preconfiguring a Raspberry Pi with the Pi Imager

## Discussion

Once you put the microSD card into your Raspberry Pi and boot up, you can connect to your Raspberry Pi from another computer using SSH. The only obstacle to doing this is that you need to know the IP address that your home hub assigned to your Raspberry Pi when it connected to it. You can find this by temporarily connecting keyboard, mouse, and monitor and following [Recipe 2.2](#), after which you should follow [Recipe 2.3](#) so that the IP address will be fixed.

Sometimes it isn't convenient to attach all those things to your Raspberry Pi. In this case, you can use the various tools available on Android and iOS to scan your network from your mobile phone and report a list of connected computers along with their IP addresses ([Figure 1-16](#)).

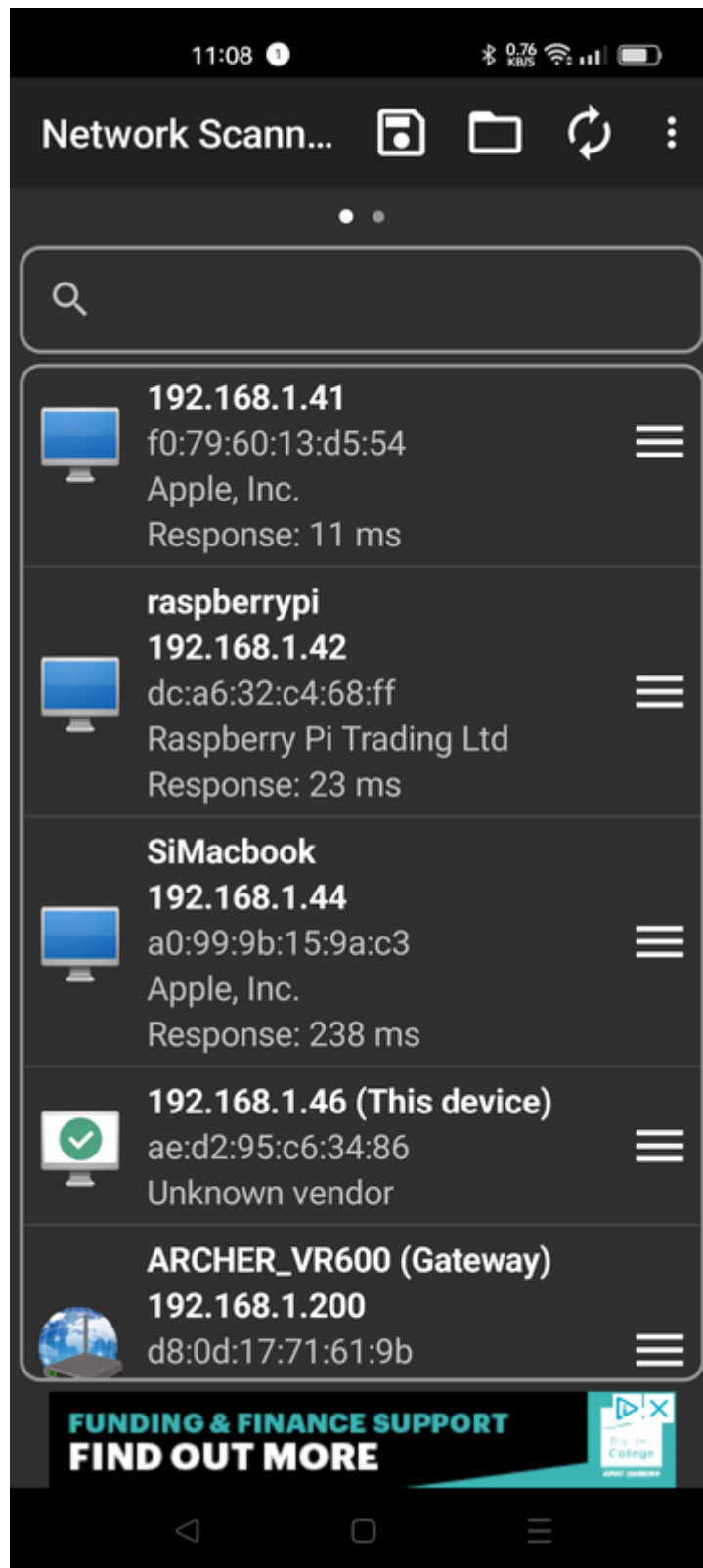


Figure 1-16. Scanning your network to find an IP address

## See Also

For more information on connecting to your Raspberry Pi with SSH, see [Recipe 2.7](#).

# 1.9 Booting from a Real Hard Disk or USB Flash Drive

## Problem

Your microSD card is too small and/or you are concerned about your entire operating system running on an SD card.

## Solution

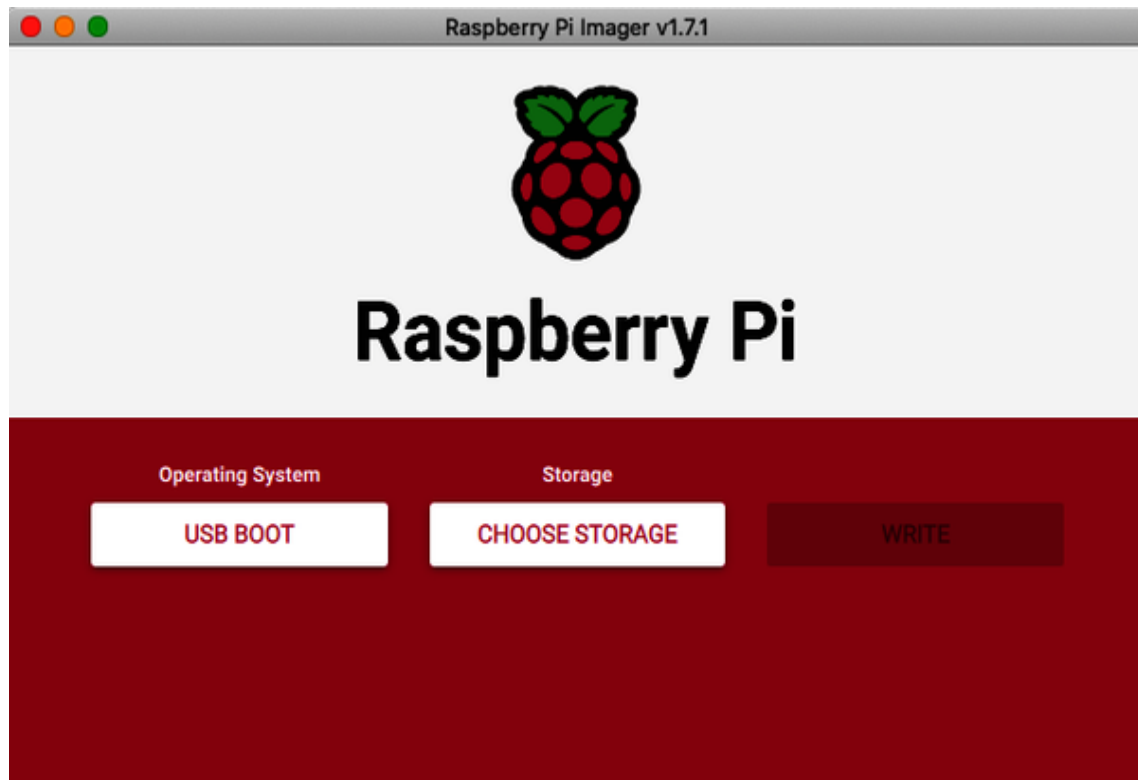
Before the Raspberry Pi 4 and 400 came along, booting from a hard disk or flash drive was possible, but not very easy, involving some complex commands and the possibility of “bricking” your Raspberry Pi if you got it wrong. If you have a Raspberry Pi 4 or 400, then it is now really straightforward and starts with the Raspberry Pi Imager.

The process is similar to setting up a microSD card using the Raspberry Pi Imager. To follow this recipe, you’ll need a Windows, Mac, or Linux computer; a USB SSD (solid state drive); and a microSD card. Even though the whole point of this recipe is to replace the microSD card as the boot device, you still need two microSD cards: one being your existing microSD boot device for your Raspberry Pi and a second blank microSD card.

1. Insert the blank microSD card into your card reader.
2. Start the Raspberry Pi Imager, and then from the Operating System drop-down, select Misc Utility Images and then Bootloader and then USB Boot ([Figure 1-17](#)). Note that if these options aren’t present in your Raspberry Pi Imager, then you probably need to download [the latest version](#).
3. Select your microSD card from the Storage drop-down and then click Write.



4. Put the newly written microSD card into your Raspberry Pi and power it up. The only purpose of this microSD card image is to reconfigure your Raspberry Pi for USB booting. Once it's done its work, the screen will turn green.
5. Power down your Raspberry Pi and swap the microSD cards, so that you now have your original microSD system card in the Raspberry Pi.
6. Boot your Raspberry Pi and from the Raspberry Menu, select Accessories and then SD Card Copier (Figure 1-18). Select your system microSD card as the source and the external USB drive as the destination, then click Start.
7. When the copy is complete, you can shut down your Raspberry Pi again and remove the microSD card. The next time you switch on your Raspberry Pi, it should boot from the USB drive.



*Figure 1-17. Using the Raspberry Pi Imager to configure USB booting*

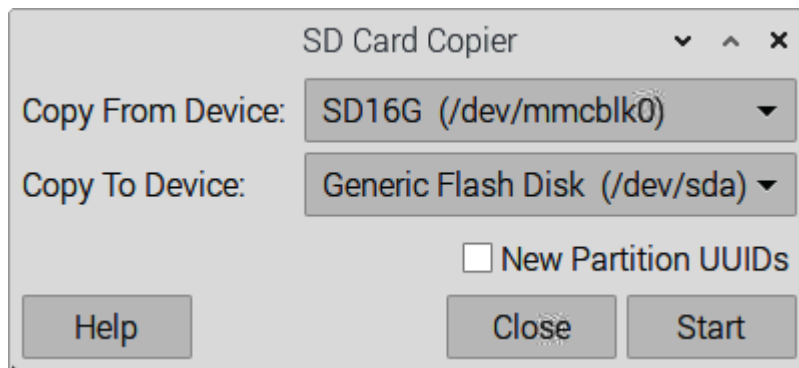


Figure 1-18. Copying your microSD card to an external flash drive

## Discussion

If you are starting with a fresh image, rather than wanting to copy over the current contents of your microSD card, then you can get away with using just one microSD card. Between steps 4 and 5, set up the microSD card with a fresh Raspberry Pi OS image by following [Recipe 1.6](#).

## See Also

Check out the [documentation for the Raspberry Pi Imager](#).

## 1.10 Connecting a DVI or VGA Monitor

### Problem

Your monitor does not have an HDMI connector, but you want to use it with your Raspberry Pi.

### Solution

Many people have been caught out by this problem. Fortunately, it is possible to buy adapters for monitors with a DVI or VGA input, but no HDMI connectors.

DVI adapters are the simplest and cheapest. They are available for less than \$5 if you search for “HDMI male to DVI female converter.”

## Discussion

Using VGA adapters is more complex because they require some electronics to convert the signal from digital to analog, so beware of leads that do not contain these. The official converter is called Pi-View and is available wherever the Raspberry Pi is sold. Pi-View has the advantage of having been tested and found to work with Raspberry Pi. You can find cheaper alternatives on the internet, but often these won't work.

However, given that new monitors all have HDMI connectors, you are probably better off spending your money on a new monitor rather than an adapter.

## See Also

eLinux has [tips on what to look for in a converter](#).

# 1.11 Using a Composite Video Monitor/TV

## Problem

The text on your low-resolution composite monitor is illegible.

## Solution

You need to adjust the resolution of the Raspberry Pi for a small screen.

The Raspberry Pi has two types of video output: (1) HDMI, and (2) composite video from the audio jack, for which you need a special lead. Of these, the HDMI is by far the better-quality option. If you're intending to use a composite video as your main screen, you might want to think again.

If you are using a composite video screen—say, because you need a really small screen—you need to make a few adjustments to fit the video output to

the screen. You need to make some changes to the file */boot/config.txt*.

You can edit this file on your Mac or PC by inserting the SD card back into an SD card reader, or you can edit it on the Raspberry Pi without having to remove the card. Editing files on the Raspberry Pi itself is normally done using the nano editor. This is a little tricky, and I suggest you read [Recipe 3.7](#) thoroughly before you try editing your first file. If you are happy to go ahead and edit the file using nano, enter the following command in a Terminal session:

```
$ sudo nano /boot/config.txt
```

Note that to save and exit nano, press Ctrl-X, then press Y (to confirm), and then press Enter.

### THE TERMINAL

If you are a Mac or Windows user, you may not be familiar with the idea of a Terminal or command line. Raspberry Pi OS is based on Linux, and although most things can be done in Linux by point and click, sometimes a need still exists to run commands when installing software or configuring the operating system in some way. To get an overview of using the Terminal, skip ahead to [Recipe 3.3](#).

If the text is too small to read, it's best to remove the SD card from the Raspberry Pi and insert it into your computer. The file will then be in the top-level directory on the SD card, and you can use a text editor on your PC (such as Notepad++) to modify it.

You need to know the resolution of your screen. For a lot of small screens, this will be 320 x 240 pixels. Find the two lines in the file that read as follows:

```
#framebuffer_width=1280  
#framebuffer_height=720
```

Remove the # from the beginning of each line and change the two numbers to the width and height of your screen. Removing the # enables the line. In

the following example, the screen size has been modified to be 320 by 240:

```
framebuffer_width=320  
framebuffer_height=240
```

Save the file and restart your Raspberry Pi. You should find that everything has become a lot easier to read. You'll probably also find that there is a big, thick border around the screen. To adjust this, see [Recipe 1.12](#).

## Discussion

Many low-cost CCTV monitors that can make a great companion for the Raspberry Pi are available when you're making something like a retro game console ([Recipe 4.4](#)). However, these monitors are often very low resolution.

## See Also

For another tutorial on using composite monitors, see [this Adafruit tutorial](#).

Also, see [Recipes 1.10](#) and [1.12](#) to adjust your picture when you're using the HDMI video output.

# 1.12 Adjusting the Picture Size on Your Monitor

## Problem

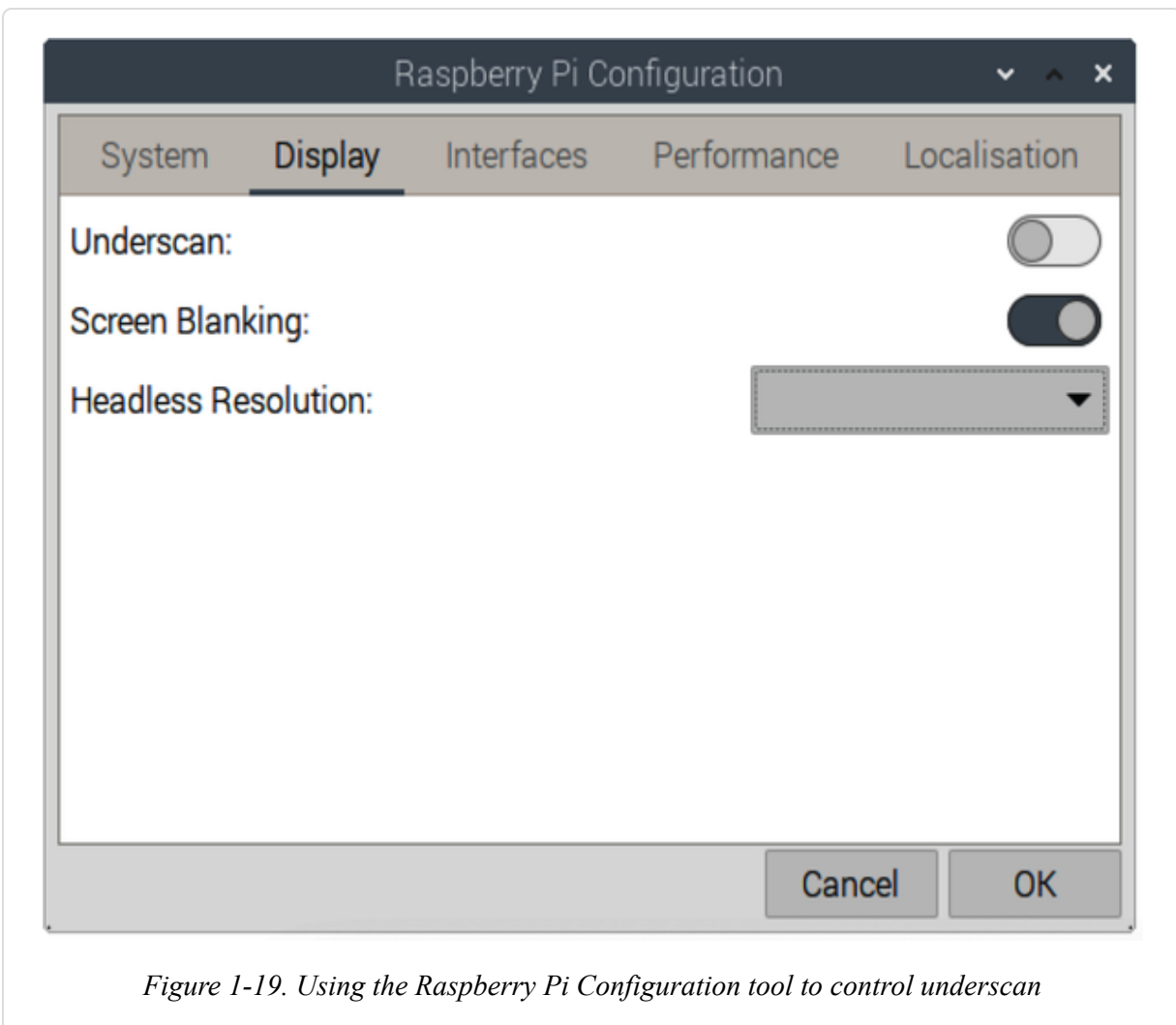
When you first connect a Raspberry Pi to a monitor, you might find that you can't read some of the text because it extends off the screen, or that the picture isn't using all the space available on the screen.

## Solution

If your problem is that a large black border is around the picture, you can make the screen fill the whole area of the monitor using the Raspberry Pi's desktop Configuration tool (see [Figure 1-19](#)). To open this, go to the Raspberry Menu, select Preferences, click Raspberry Pi Configuration, and select the Display tab.

Click the toggle switch next to Underscan. Note that the change will not take effect until you have clicked OK and then rebooted your Raspberry Pi.

If you have the opposite problem and your text extends off the edges of the screen, the solution is the same: click the toggle switch for Underscan.



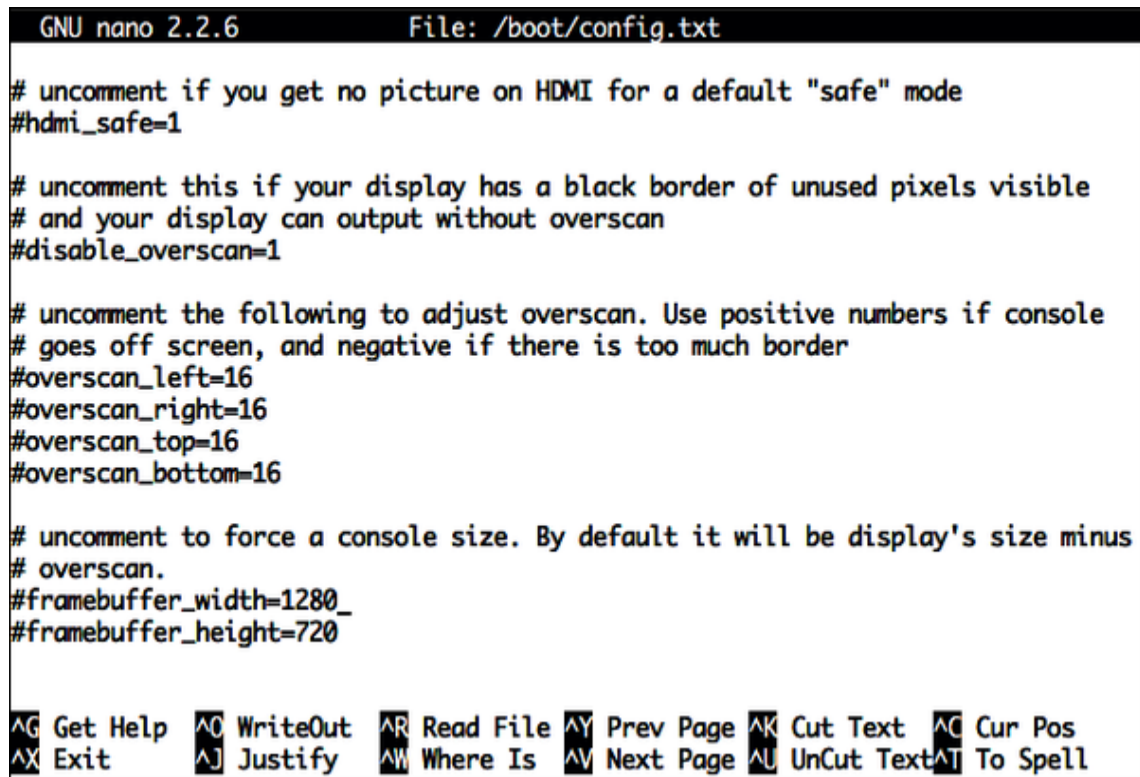
*Figure 1-19. Using the Raspberry Pi Configuration tool to control underscan*

The second step is to edit the file `/boot/config.txt`. You can do this either by removing the SD card and mounting it on your PC or Mac or by editing the

SD card on the Raspberry Pi. Editing files on the Raspberry Pi itself is normally done using the nano editor. This is a little tricky; I suggest you read [Recipe 3.7](#) thoroughly before you try editing your first file. If you are happy to go ahead and edit the file using nano, enter the following command in a Terminal session:

```
$ sudo nano /boot/config.txt
```

Look for the section dealing with overscan. The four lines you need to change are shown in the middle of [Figure 1-20](#), each beginning with `#overscan`.



```
GNU nano 2.2.6 File: /boot/config.txt
# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1

# uncomment this if your display has a black border of unused pixels visible
# and your display can output without overscan
#disable_overscan=1

# uncomment the following to adjust overscan. Use positive numbers if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's size minus
# overscan.
#framebuffer_width=1280_
#framebuffer_height=720

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell
```

*Figure 1-20. Adjusting overscan*

For the lines to take effect, you need to enable them by removing the `#` character from the start of each line.

Then, using trial and error, change the settings until the screen fills as much of the monitor as possible. Note that the four numbers should be negative. Try setting them all to  $-20$  to start with. This will decrease the area of the screen that is used.

To save and exit nano, press Ctrl-X, then press Y (to confirm), and then press Enter.

## Discussion

Having to repeatedly restart the Raspberry Pi to see the effects of the changes in resolution is a little tedious. Fortunately, you need to do this procedure only once. Most monitors and TVs work just fine without any need for underscanning.

## See Also

You can also configure underscanning using the `raspi-config` utility.

# 1.13 Maximizing Performance

## Problem

Your Raspberry Pi seems to be very slow, so you want to overclock it to make it run faster.

## Solution

If you have a Raspberry Pi 3, 4, or 400 with a quad-core processor, you are unlikely to find it to be too slow. However, older Raspberry Pis 1 and 2 can be pretty sluggish.

You can increase the clock frequency of a Raspberry Pi 1 or 2 to make it run a little faster. This will make it consume a bit more power and run a little hotter (see the Discussion section that follows).



The method of overclocking described here is called *dynamic overclocking* because it automatically monitors the temperature of the Raspberry Pi and drops the clock speed back down if things begin to get too hot. This is called *throttling*.

Run the `raspi-config` utility by issuing the following command in an SSH Terminal:

```
$ sudo raspi-config
```

Select the Overclock option. You are then presented with the options shown in [Figure 1-21](#).

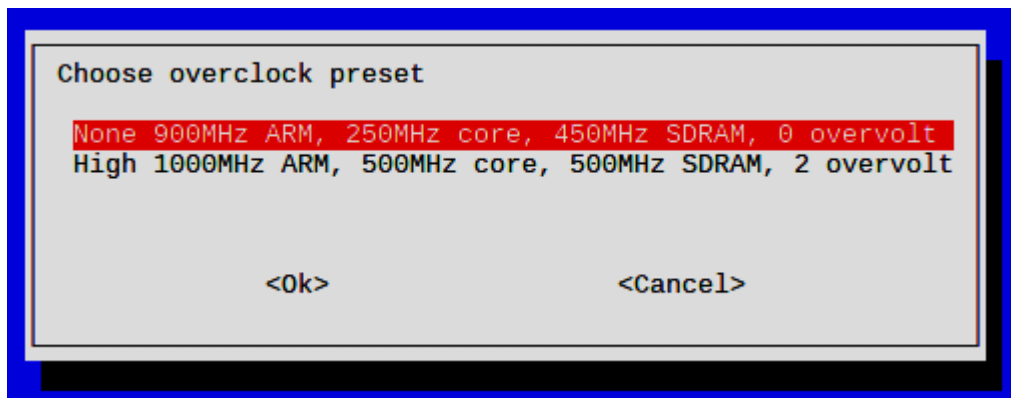


Figure 1-21. Configuring overclocking with the `raspi-config` utility from the command line

Select an option. If you find that your Raspberry Pi starts to become unstable and hangs unexpectedly, you might need to choose a more conservative option or turn overclocking off by setting it back to `None`.

## Discussion

The performance improvements from overclocking can be quite dramatic. To measure these, I used a Raspberry Pi B, without a case, at an ambient room temperature of 60 degrees (15 degrees C).

The test program was the following Python script. This just hammers the processor (that is, makes it work really hard) and is not really representative

of the other things that go on in a computer, such as writing to the SD card, graphics, and so on. However, it does give a good indication of raw CPU performance if you want to test the effect of overclocking on your Raspberry Pi:

```
import time

def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

before_time = time.process_time()
for i in range(1, 10000):
    factorial(200)
after_time = time.process_time()

print(after_time - before_time)
```

Note that we are jumping ahead a lot here, so if you are not familiar with Python, come back to this when you have read through [Chapter 5](#).

Check out the results of the test in [Table 1-3](#). The current and temperature were measured using test equipment.

*Table 1-3. Overclocking*

	<b>Speed test</b>	<b>Current</b>	<b>Temperature (degrees C)</b>
700 MHz	15.8 seconds	360mA	27
1 GHz	10.5 seconds	420mA	30

As you can see, the performance has increased by 33% but at a cost of drawing more current and a slightly higher temperature.

A well-ventilated enclosure will help to keep your Raspberry Pi running at full speed. Some efforts to add water cooling to the Raspberry Pi have also been made. Frankly, this is just silly.

## See Also

Much more information about the `raspi-config` utility is available at <https://oreil.ly/1hwy6>.

# 1.14 Changing Your Password

## Problem

You want to change your password.

## Solution

After you install Raspberry Pi OS onto your SD card, you are prompted to create a user account and associated password. You can change the password any time using the Raspberry Pi Configuration tool. To open this, go to the Raspberry Menu, select Preferences, and then click Raspberry Pi Configuration. Click the System tab. There, you'll find the Change Password option ([Figure 1-22](#)).

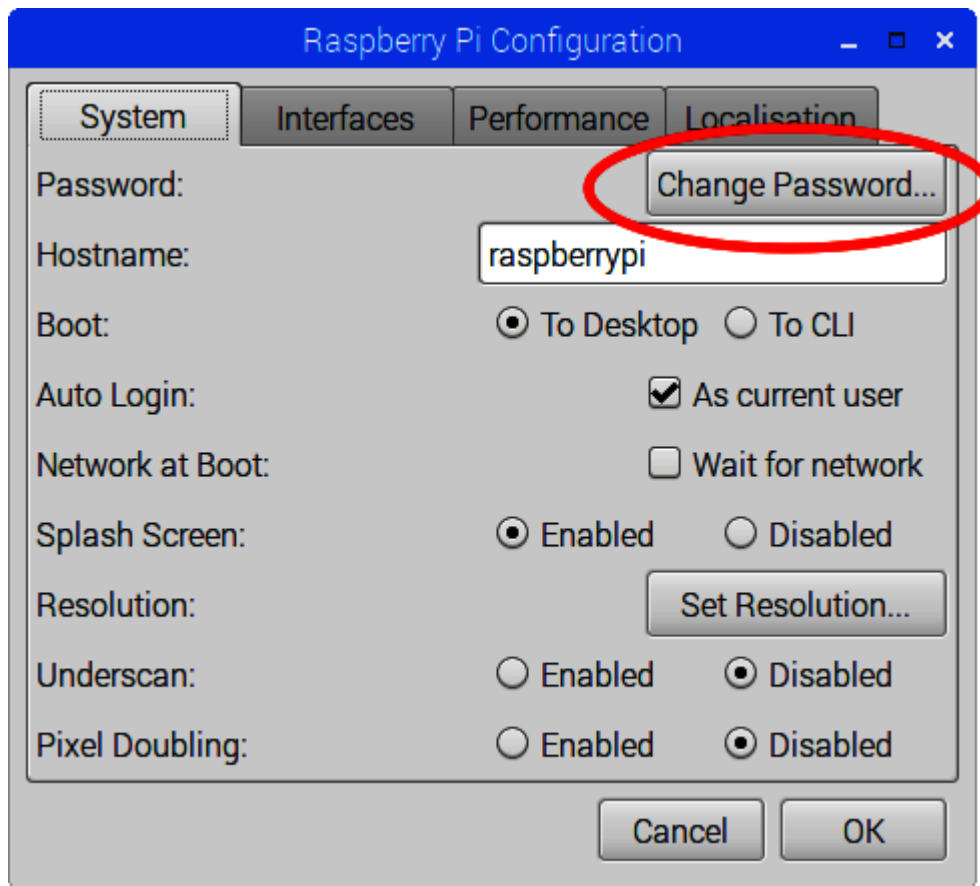


Figure 1-22. Changing your password with the Raspberry Pi Configuration tool

Changing your password is one setting for which you do not need to restart your Raspberry Pi for the change to take effect.

## Discussion

You can also change the password from a Terminal session simply by using the `passwd` command, as follows:

```
$ passwd
Changing password for pi.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

## See Also

You can also change your password using the `raspi-config` utility.

# 1.15 Shutting Down Your Raspberry Pi

## Problem

You want to shut down your Raspberry Pi.

## Solution

In the upper lefthand corner of the desktop, click the Raspberry Menu. A dialog box opens, offering three shutdown options (Figure 1-23):

### Shutdown

Shuts down the Raspberry Pi. You will need to unplug the power and then plug it back in to get the Raspberry Pi to boot up again. Or, if you have a Pi 400, press the Power button on the keyboard.

### Reboot

Reboots the Raspberry Pi.

### Logout

Logs you out and displays a prompt to enter your login credentials so that you can log back in.

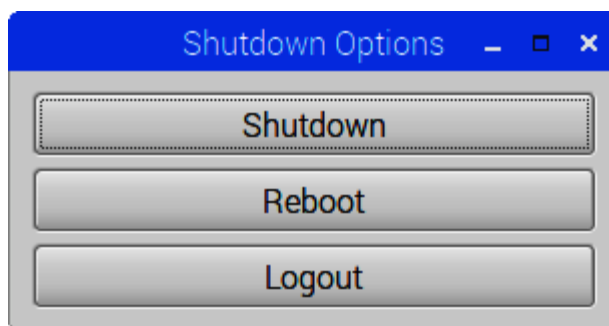


Figure 1-23. Shutting down your Raspberry Pi

You can also reboot using the Terminal by issuing the following command:

```
$ sudo reboot
```

You might need to do this after installing some software. When you do reboot, you'll see the message shown in [Figure 1-24](#), which illustrates the multiuser nature of Linux and warns all users connected to the Pi.



*Figure 1-24. Shutting down your Raspberry Pi from the Terminal*

## Discussion

It is better to shut down your Raspberry Pi as described than to simply pull out the power plug because your Raspberry Pi might be in the middle of writing to the microSD card as you power it down. This could lead to file corruption.

Shutting down a Raspberry Pi does not actually turn off the power. It goes into a low-power mode—and it is a pretty low-power device anyway (but the Raspberry Pi hardware has no control over its power supply).

When a Raspberry Pi 400 (with its built-in keyboard) has been shut down, you can turn it on by pressing the F10 key, which also has an on/off icon on the key.

## **See Also**

You can buy a [module that will turn off the power](#) when the Raspberry Pi shuts down.

For information on adding a Start button to your Raspberry Pi, see [Recipe 13.13](#).

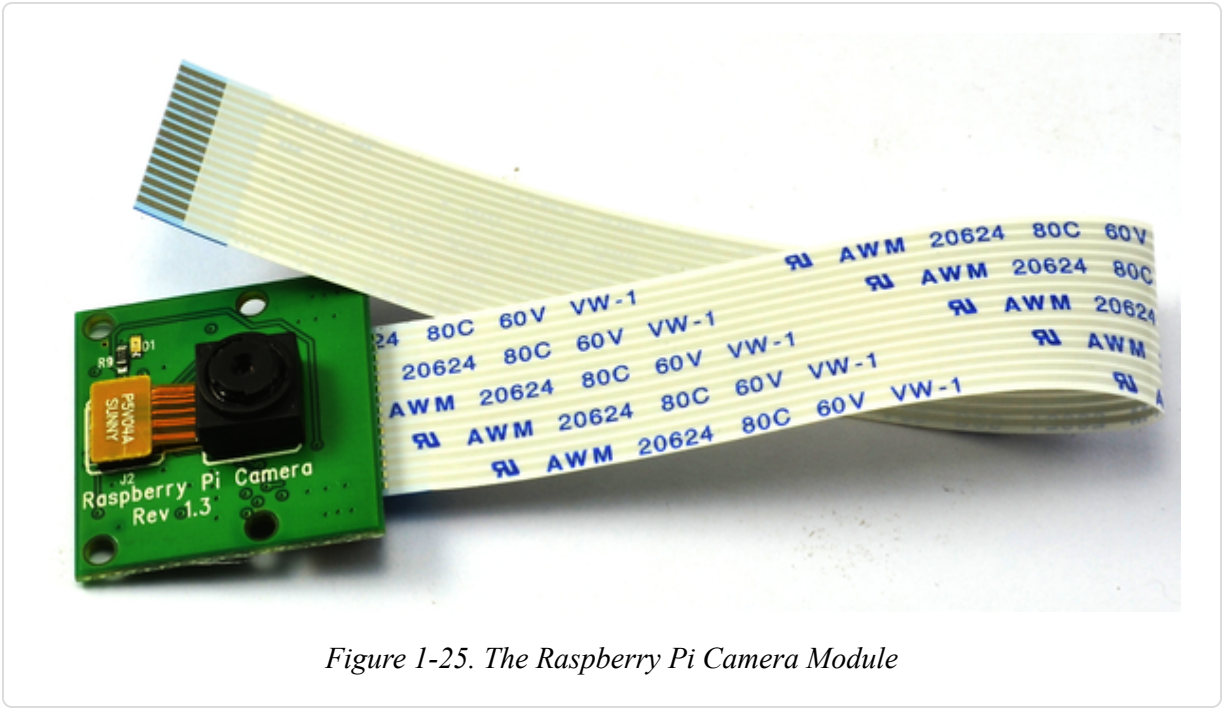
## **1.16 Installing the Raspberry Pi Camera Module**

### **Problem**

You want to use the Raspberry Pi Camera Module.

### **Solution**

The Raspberry Pi Camera Module ([Figure 1-25](#)) is attached to a Raspberry Pi by a ribbon cable.



*Figure 1-25. The Raspberry Pi Camera Module*

There are three versions of the Pi Camera: the original version 1 (as shown in [Figure 1-25](#)); the newer, higher-resolution version 2; and the HQ (High Quality) camera, which takes interchangeable lenses and boasts a resolution of 12 megapixels.

The ribbon cable attaches to a special connector between the audio and HDMI sockets on a Raspberry Pi 2, 3, or 4. To fit the cable onto your Pi, gently pull up the levers on either side of the connector so that they unlock, and then press the cable into the slot with the shiny metal connector pads of the cable facing away from the Ethernet socket. Press the two levers of the connector back down to lock the cable in place ([Figure 1-26](#)).



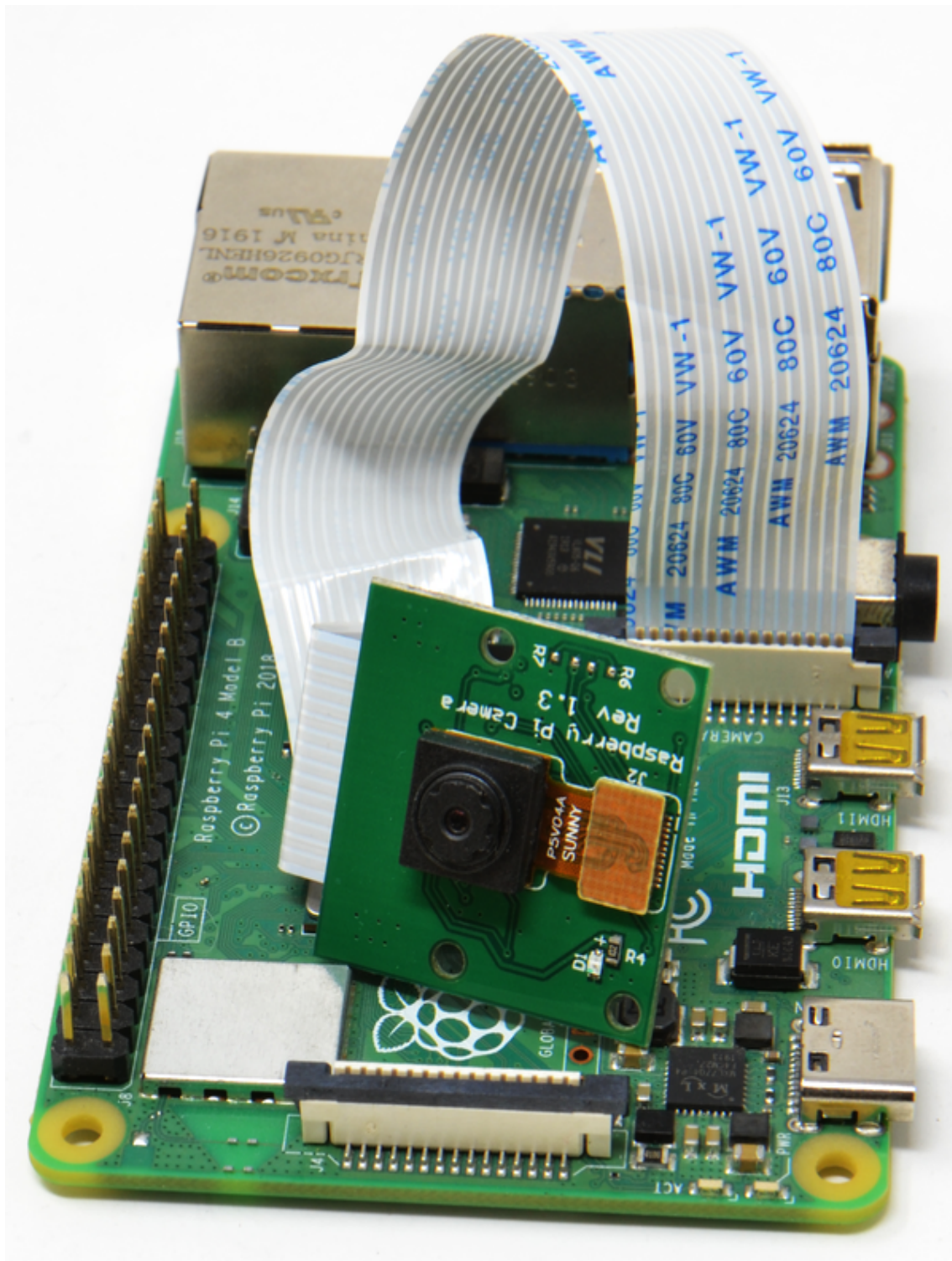


Figure 1-26. Attaching a Raspberry Pi Camera Module to a Raspberry Pi 4 model B

## WARNING

The Camera Module packaging states that it is sensitive to static. Before handling it, ground yourself by touching something grounded, like the metal case of a PC.

Note that the Raspberry Pi Zero requires a special cable or adapter because its Camera connector is smaller than that of a full-size Raspberry Pi (see “[Modules](#)”).

The Camera Module requires some software configuration. At the time of writing, the camera interface is moving from Raspberry Pi–specific software to the `libcamera` library. So, before following the instructions, you may want to [check to see the current status of the Camera Module’s software](#).

These instructions describe the “legacy” Raspberry Pi camera software that works on even quite ancient models of Raspberry Pi.

The graphical Raspberry Pi Configuration tool doesn’t include an option to enable the camera, so you must start the `raspi-config` utility from a Terminal session.

```
$ sudo rasp-config
```

This will display options for configuring your Raspberry Pi.

Select Interfacing Options, and you’ll see the Camera option ([Figure 1-27](#)). Select the first option to enable legacy support, and then reboot your Raspberry Pi.

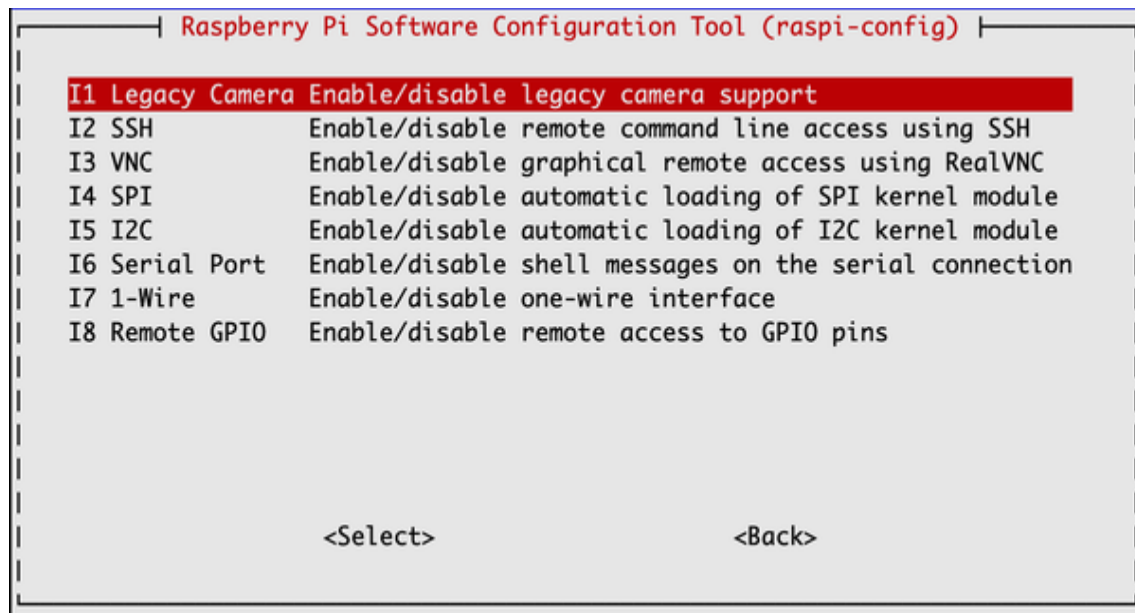


Figure 1-27. Enabling the camera using `raspi-config` from the command line

Two commands are available for capturing still images and videos:

`raspistill` and `raspivid`.

To capture a single still image, use the `raspistill` command:

```
$ raspistill -o image1.jpg
```

A preview screen displays for about five seconds and then takes a photograph and stores it in the file `image1.jpg` in the current directory.

To capture video, use the command `raspivid`:

```
$ raspivid -o video.h264 -t 10000
```

The number at the end of the line is the recording duration in milliseconds—in this case, 10 seconds.

## Discussion

Both `raspistill` and `raspivid` have a large number of options. If you type either command without any parameters, help text displays the available options.

You can also buy a NoIR (no infrared) version of the camera that has the infrared filter removed from the Camera Module to allow it to work at night under infrared illumination.

An alternative to the Camera Module is to use a USB webcam (see [Recipe 8.2](#)).

## See Also

Find out more about the Raspberry Pi Camera Module at <https://oreil.ly/diUuB>.

# 1.17 Using Bluetooth

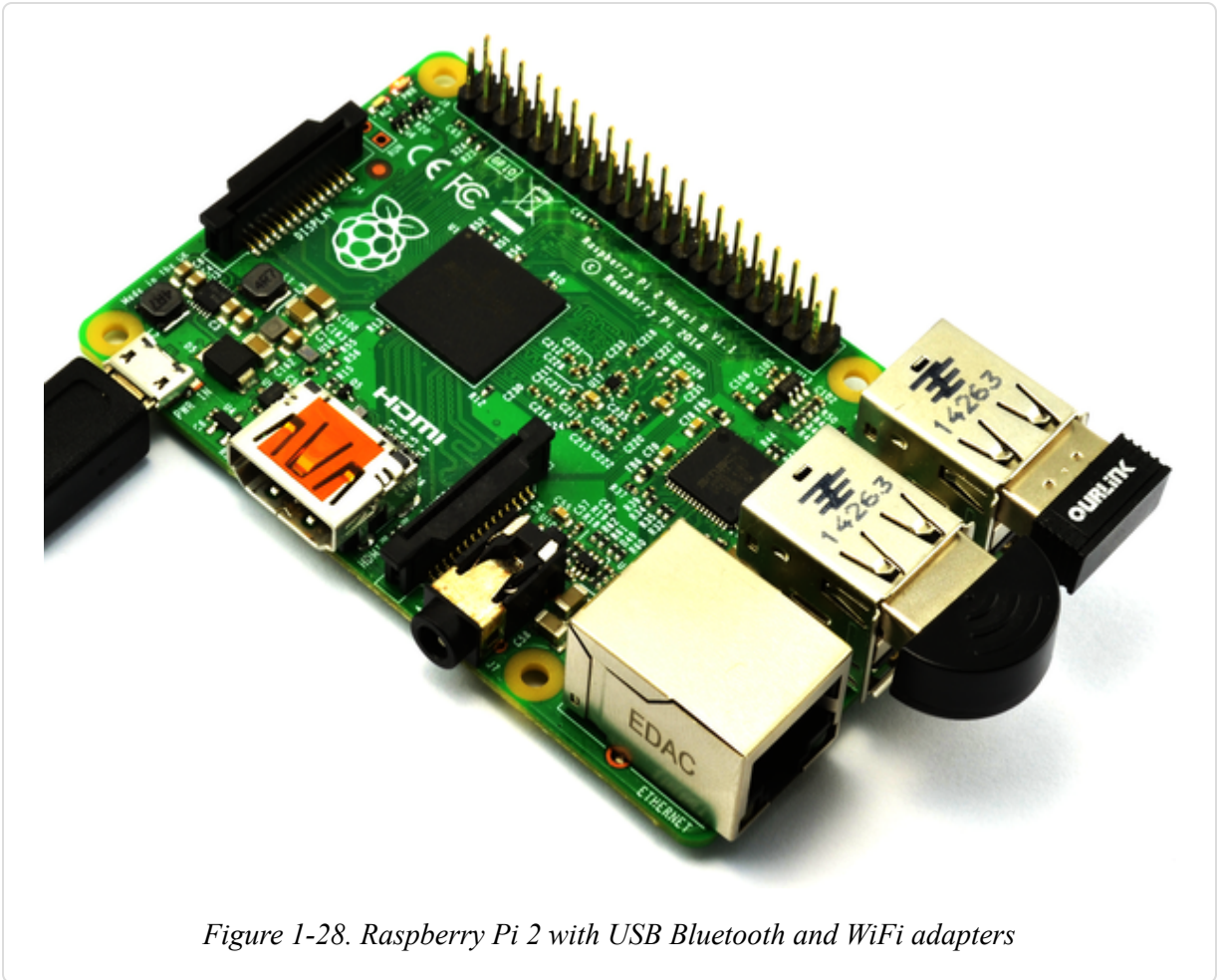
## Problem

You want to use Bluetooth with your Raspberry Pi.

## Solution

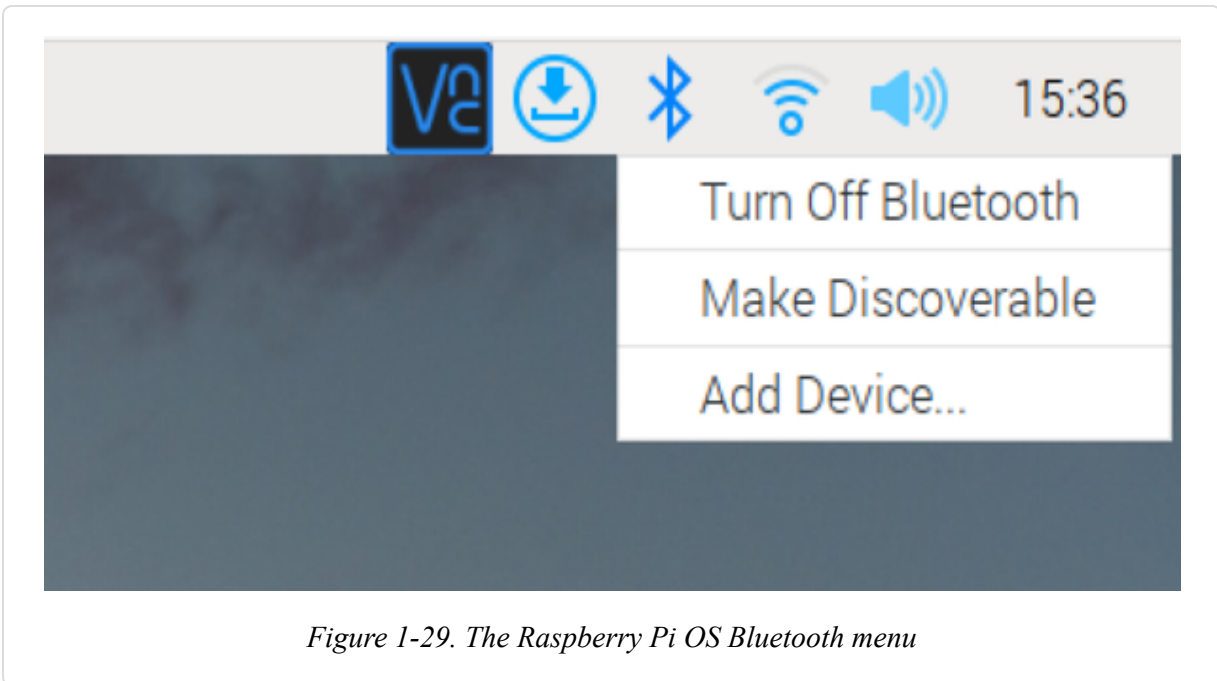
If you have a Raspberry Pi 3, 4, or 400, the good news is that along with WiFi, you also get Bluetooth hardware. If you have an older Raspberry Pi, you can attach a USB Bluetooth adapter to it. In both cases, the software that you need for Bluetooth is now included in Raspberry Pi OS.

If you have an older Raspberry Pi, be aware that not all Bluetooth adapters are compatible with the Raspberry Pi. Most are, but to be sure, buy one that is advertised as working with the Raspberry Pi. [Figure 1-28](#) shows a Raspberry Pi 2 equipped with both a USB Bluetooth adapter (nearest to the camera) and a USB WiFi adapter.



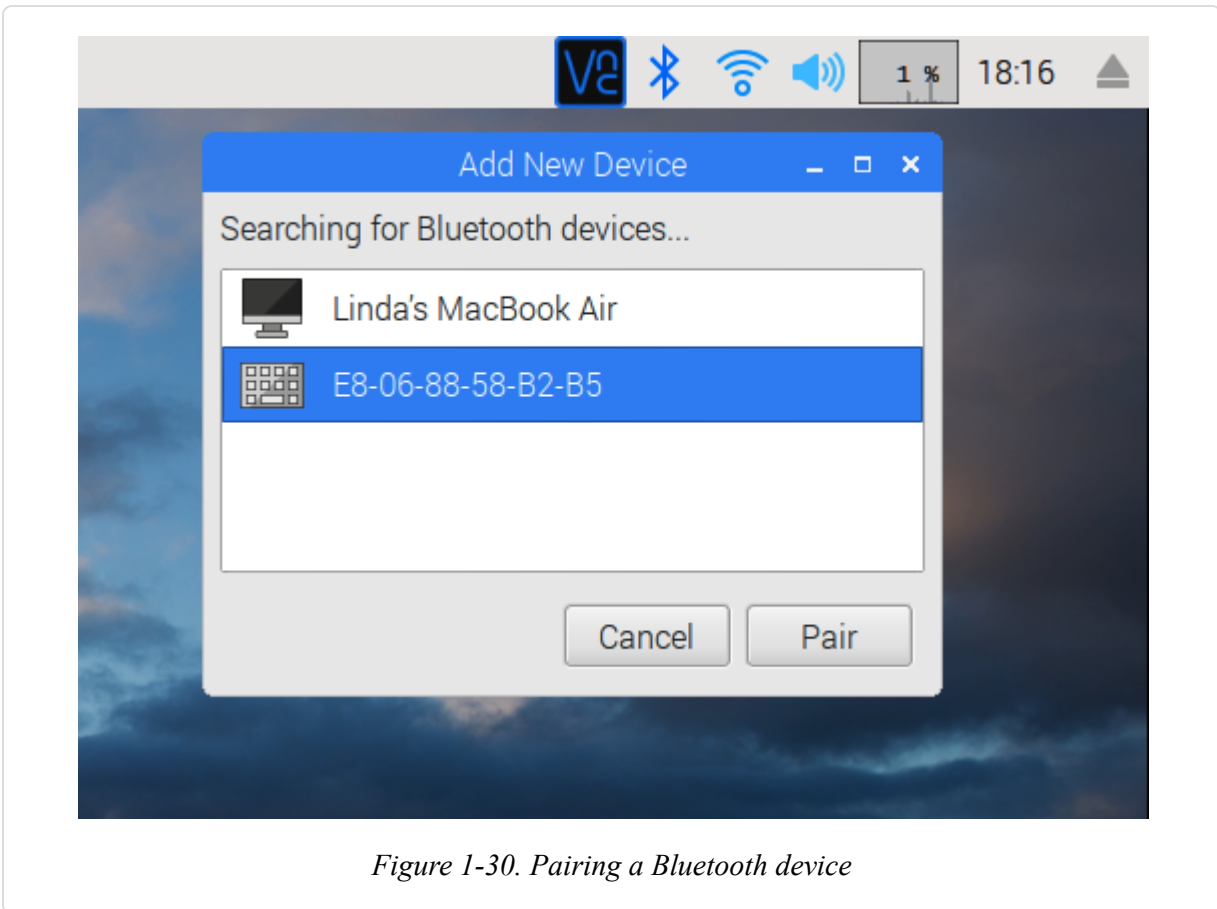
*Figure 1-28. Raspberry Pi 2 with USB Bluetooth and WiFi adapters*

Bluetooth features are integrated into the Raspberry Pi OS desktop in much the same way as on a Mac. In the upper-right corner of the screen, you will see the Bluetooth icon ([Figure 1-29](#)). Click this to open a menu of Bluetooth options.



*Figure 1-29. The Raspberry Pi OS Bluetooth menu*

If you want to connect a Bluetooth peripheral such as a keyboard, click Add Device. The Add New Device dialog box opens, showing a list of available devices with which you can connect, or *pair* (Figure 1-30).



*Figure 1-30. Pairing a Bluetooth device*

You can then select the device that you want to pair with, and then follow the instructions that appear on your Raspberry Pi and the device you are pairing with.

## **Discussion**

You can pair phones, Bluetooth speakers, keyboards, and mice to your Raspberry Pi. I find that connecting a new Bluetooth device doesn't always work the first time. So if you initially have a problem pairing with a device, try a few more times before you give up.

Most of the time, using the desktop interface to add Bluetooth devices to your Raspberry Pi system is convenient; however, you can also pair Bluetooth devices using the command-line interface.

To run Bluetooth commands from the command line, use the `bluetoothctl` command:

```
$ bluetoothctl
[NEW] Controller B8:27:EB:50:37:8E raspberrypi [default]
[NEW] Device 51:6D:A4:B8:D1:AA 51-6D-A4-B8-D1-AA
[NEW] Device E8:06:88:58:B2:B5 si's keyboard #1
[bluetooth]#
```

This scans for Bluetooth devices and also provides a `pair` command that will allow you to pair with the device using its ID—for example:

```
[bluetooth]# pair E8:06:88:58:B2:B5
```

## See Also

Take a look at a [list of Bluetooth adapters that are compatible with the Raspberry Pi](#).

The Blue Dot software for Android phones enables you to control hardware attached to your Raspberry Pi using Bluetooth and your mobile phone. You will find an example of this in [Recipe 11.8](#).

If you pair your Raspberry Pi with a Bluetooth speaker, you also need to set the speaker to be the output for sound ([Recipe 16.2](#)).



# Chapter 2. Networking

---

## 2.0 Introduction

The Raspberry Pi is designed to be connected to the internet. Its ability to communicate on the internet is one of its key features and opens up all sorts of possible uses, including home automation, web serving, network monitoring, and so on.

The connection can be wired through an Ethernet cable (in the case of most models), and newer models have built-in WiFi.

Having a connected Raspberry Pi also means that you can connect to it remotely from another computer. This is very useful for situations in which the Raspberry Pi itself is used as a *headless* server and doesn't have a keyboard, mouse, and monitor attached to it.

This chapter gives you recipes for connecting your Raspberry Pi to the internet and controlling it remotely over a network.

## 2.1 Connecting to a Wired Network

### Problem

You want to connect your Raspberry Pi to the internet using a wired network connection.

### Solution

First, if you have an old model A Raspberry Pi or a Pi Zero, there is no RJ45 connector for Ethernet. In this case, your best option for internet access is to use a wireless USB adapter (see [Recipe 2.5](#)).

If you have a model B or B+ Raspberry Pi (1, 2, 3, 4, or 400) then you are in luck; just plug an Ethernet patch cable into its RJ45 socket and then

connect the other end to a spare socket on the back of your home router (Figure 2-1).

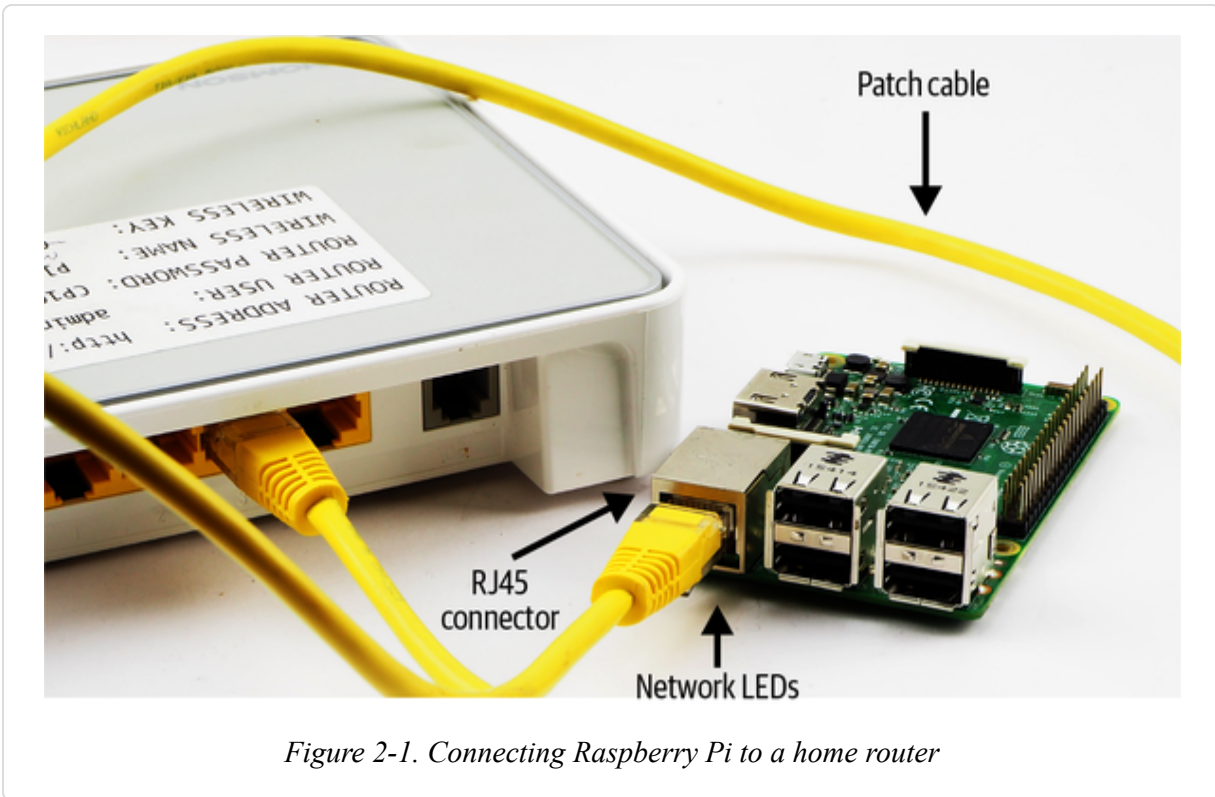


Figure 2-1. Connecting Raspberry Pi to a home router

The network LEDs on your Raspberry Pi should immediately begin to flicker as the Raspberry Pi connects to your network.

## Discussion

Raspberry Pi OS is preconfigured to connect to any network using Dynamic Host Configuration Protocol (DHCP). It will automatically be assigned an IP address as long as DHCP is enabled on your network.

If the LEDs blink, but you cannot connect to the internet on your Raspberry Pi using a browser, check that DHCP is enabled on your network management console. Go to the admin page of your home router, sign in with the admin password, and look for an option like that shown in Figure 2-2.

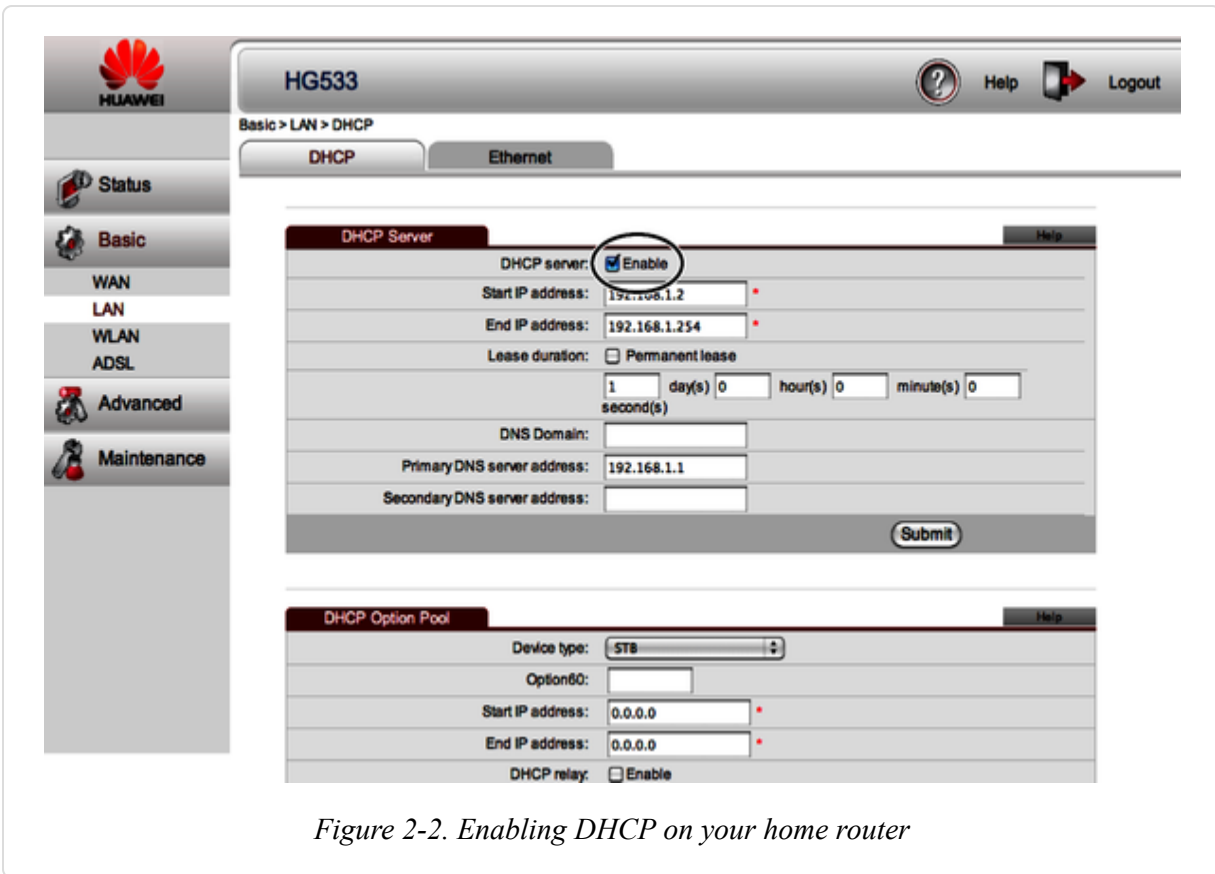


Figure 2-2. Enabling DHCP on your home router

## See Also

To connect to a wireless network, see [Recipe 2.5](#).

## 2.2 Finding Your IP Address

### Problem

You want to know the IP address of your Raspberry Pi so that you can communicate with it, whether connecting to it as a web server, exchanging files, or controlling it remotely with SSH ([Recipe 2.7](#)) or VNC ([Recipe 2.8](#)).

### Solution

An IPv4 address (as used for local addresses) is a four-part number uniquely identifying a computer's network interface within a network. Each

part is separated from the next part by a dot.

To find the IP address of your Raspberry Pi, you need to issue this command in a Terminal window:

```
$ hostname -I
192.168.1.16 fd84:be52:5bf4:ca00:618:fd51:1c .....
```

The first part of the response is the local IP address of your Raspberry Pi on your home network.

## Discussion

A Raspberry Pi can have more than one IP address (i.e., one for each network connection). So if you have both a wired connection and a wireless connection to your Pi, it would have two IP addresses. Normally, however, you would connect it by only one method or the other, not both. To see all the network connections, use the `ifconfig` command:

```
$ ifconfig

eth0      Link encap:Ethernet  HWaddr b8:27:eb:d5:f4:8f
          inet addr:192.168.1.16  Bcast:192.168.255.255
Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1114 errors:0 dropped:1 overruns:0 frame:0
          TX packets:1173 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:76957 (75.1 KiB)  TX bytes:479753 (468.5 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan0     Link encap:Ethernet  HWaddr 00:0f:53:a0:04:57
          inet addr:192.168.1.13  Bcast:192.168.255.255
Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

```
RX packets:38 errors:0 dropped:0 overruns:0 frame:0
TX packets:28 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:6661 (6.5 KiB) TX bytes:6377 (6.2 KiB)
```

Looking at the results of running the `ifconfig` command, you can see that the Pi in question is connected by both a wired connection (`eth0`) with an IP address of `192.168.1.16` and a wireless connection (`wlan0`) with an IP address of `192.168.1.13`. The `lo` network interface is a virtual interface that allows the computer to communicate with itself.

## See Also

[Wikipedia](#) has everything you want to know about IP addresses.

## 2.3 Setting a Static IP Address

### Problem

You want to set the IP address of your Raspberry Pi so that it does not change.

### Solution

Although it is possible to set the static IP address of a Raspberry Pi on the Pi itself, it is better practice to configure this on your network, as it may lead to problems if you move the Raspberry Pi to a different network.

All the computers, TVs, phones, and other internet-enabled devices in your home are generally connected to the internet via the router that links your phone line, 4G, or a fiber-optic cable to your house. All of these devices, whether they connect to the router by WiFi or via a direct cable connection, are said to be part of your *local area network* (LAN).

By default, when you connect a new device to your LAN (such as a Raspberry Pi), either by plugging it in with an Ethernet cable or by using WiFi, the LAN controller (your router) will use a system called DHCP to

allocate an IP address for the device. This address will be allocated from a pool of IP addresses that might range from, for example, 192.168.1.2 to 192.168.1.199 (or maybe 10.0.0.2 to 10.0.0.199). In other words, just the last part of the four-part IP address changes for each device connected to the LAN.

When DHCP allocates an IP address to a device, it does so with a lease time, which is how long that device will be guaranteed to keep the IP address without risk of it being allocated to some other device. Generally speaking, the default for this lease time is fairly short; on my router, it's a week. This means that the IP address of my Raspberry Pi can change without warning after a week of inactivity, and if the Pi is being used in a project without keyboard, mouse, and monitor, it can be difficult to find its IP address to allow me to connect to it. This is why you might want to set a *static IP address* for your Raspberry Pi.

One way to ensure that your Raspberry Pi's IP address doesn't change is to simply go to your router control interface and change the DHCP lease time to a much higher value. To access this interface, you will need to use a computer (it could be your Raspberry Pi, but it doesn't have to be) and go to a specific address that is often written on the router, described as *router address* or *admin console address*. For my router this is *http://192.168.1.1*. A username and password will also need to be entered. These are not the same as the WiFi access point name and password. They're often also written somewhere on the router and often have default values of *admin* and *password*, respectively.

After you are connected, you will need to hunt around your admin console's various pages for any mention of DHCP settings, which should look something like [Figure 2-3](#).

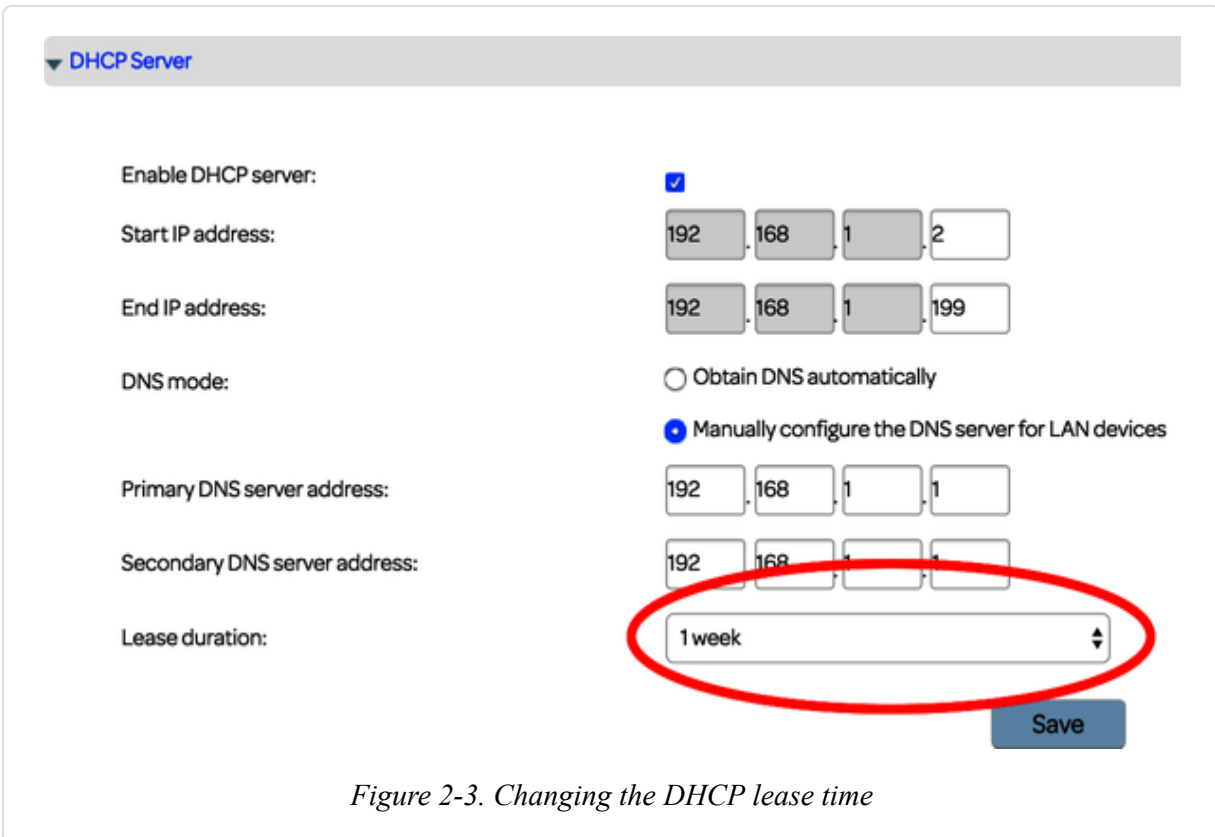


Figure 2-3. Changing the DHCP lease time

Change the “Lease duration” (or whatever it’s specifically labeled for your router) to the maximum allowed.

One downside of extending the lease duration like this is that it applies to all the devices on your LAN. So if you have a lot of devices, it’s possible that you might run out of IP addresses because DHCP is unable to reallocate old IP addresses until the lease period has expired.

A better approach is to use something called DHCP reservation. This instructs DHCP to permanently allocate a particular IP address to a particular device. In [Figure 2-4](#), you can see that I have allocated the IP address of 192.168.1.3 to the device raspberrypi-Ethernet (a Raspberry Pi connected by Ethernet cable to the router).

From now on, whenever that Raspberry Pi is connected to the LAN, it will be assigned the IP address 192.168.1.3, and DHCP will not allocate that IP address to any other device.

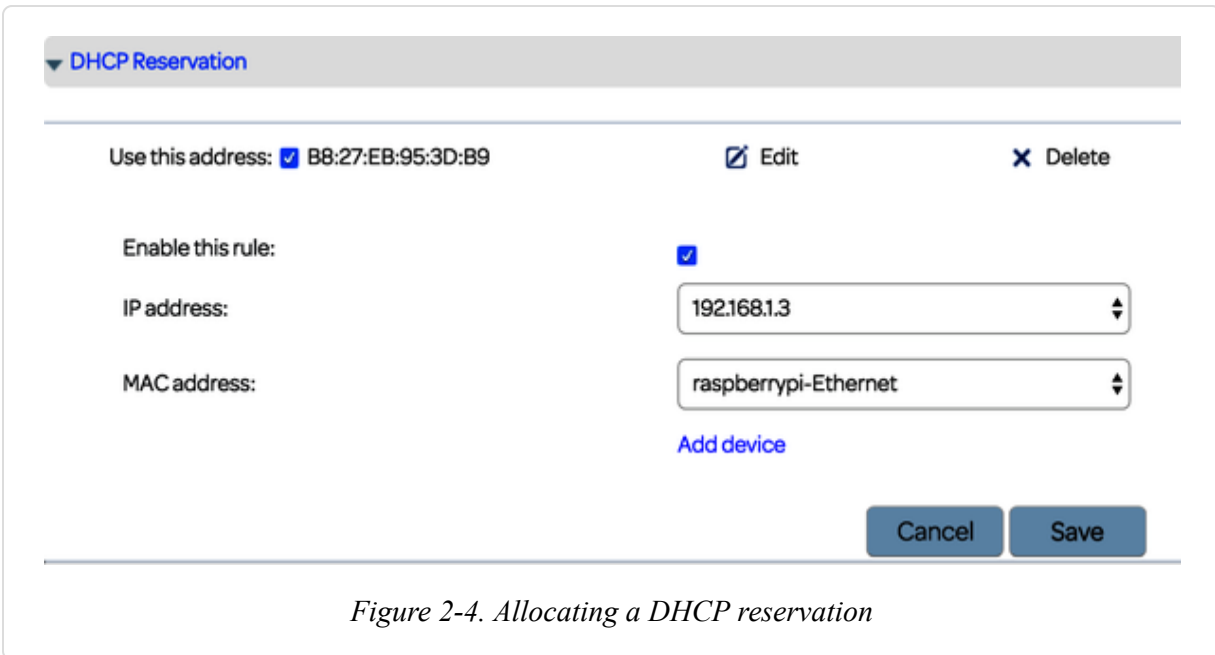


Figure 2-4. Allocating a DHCP reservation

## Discussion

Networking has changed a lot with different versions of Raspberry Pi OS. These instructions apply to the latest (as of this writing) version. If you don't have the latest version of Raspberry Pi OS, you should get it, because Raspberry Pi OS is always evolving and improving. You can learn how to do this in [Recipe 3.40](#). To find out what version of the OS you have, see [Recipe 3.39](#).

## See Also

[Wikipedia](#) has everything you want to know about IP addresses.

## 2.4 Setting the Network Name of a Raspberry Pi

### Problem

You want to change the name of your Raspberry Pi as it appears on your network so that it's not just called "raspberrypi."



## **Solution**

There are several ways of doing this. Whichever method you use, make sure that the network name you choose does not contain spaces and contains only letters, numeric digits, and the hyphen (-) character.

In all three methods, you also need to restart your Raspberry Pi for the changes to take effect.

### **Setting the network name using the Raspberry Pi Configuration tool**

Unless you are running your Raspberry Pi headless (without monitor and keyboard attached), the simplest way to set the network name of your Raspberry Pi is to use the Raspberry Pi Configuration tool. To open this, go to the Raspberry Menu, select Preferences, and then click Raspberry Pi Configuration. Then click the System tab ([Figure 2-5](#)).

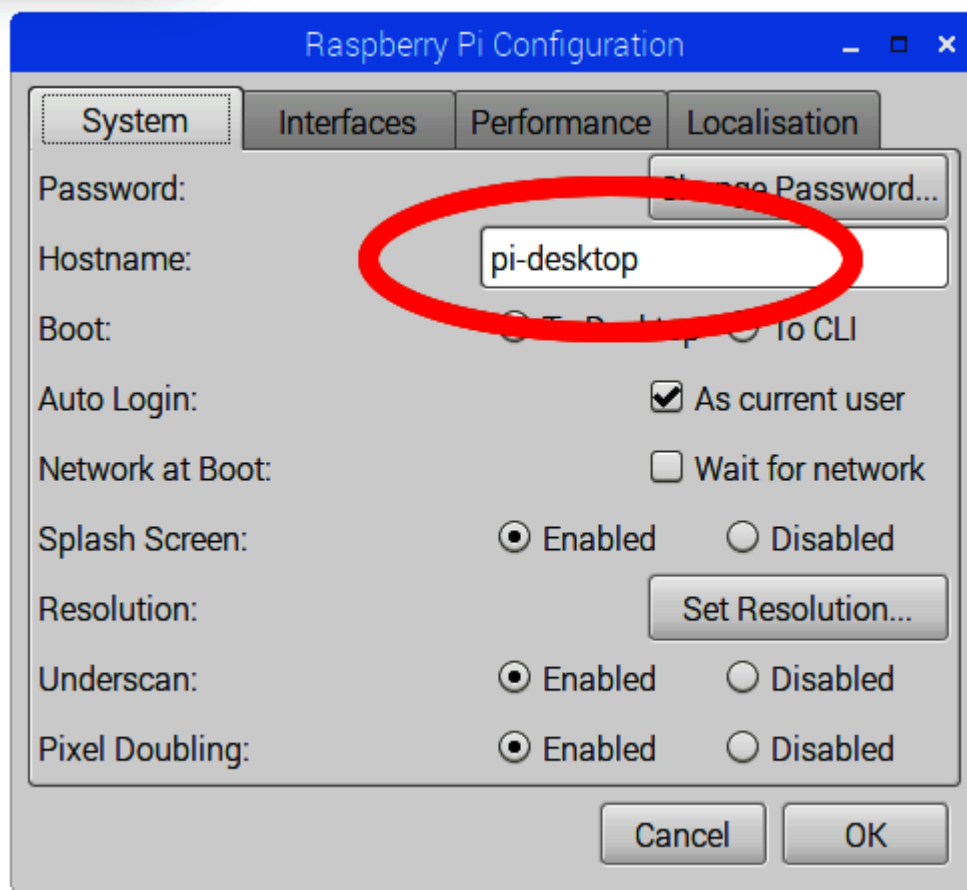


Figure 2-5. Changing the hostname using the Raspberry Pi Configuration tool

Change the name in the Hostname field to your preferred name and click OK. You are prompted to reboot for the changes to take effect ([Recipe 1.15](#)).

### Setting the network name using the command line (the easy way)

You can also change your Raspberry Pi's network name from the command line using the `raspi-config` utility. Run the following command in a Terminal session:

```
$ sudo raspi-config
```

This opens the `raspi-config` utility. Use the up/down arrow keys to select Network Options and then press Enter. This opens a form in which you can enter the new network name (Figure 2-6). Note that this interface uses only the command line, so you can use it from an SSH session (Recipe 2.7).

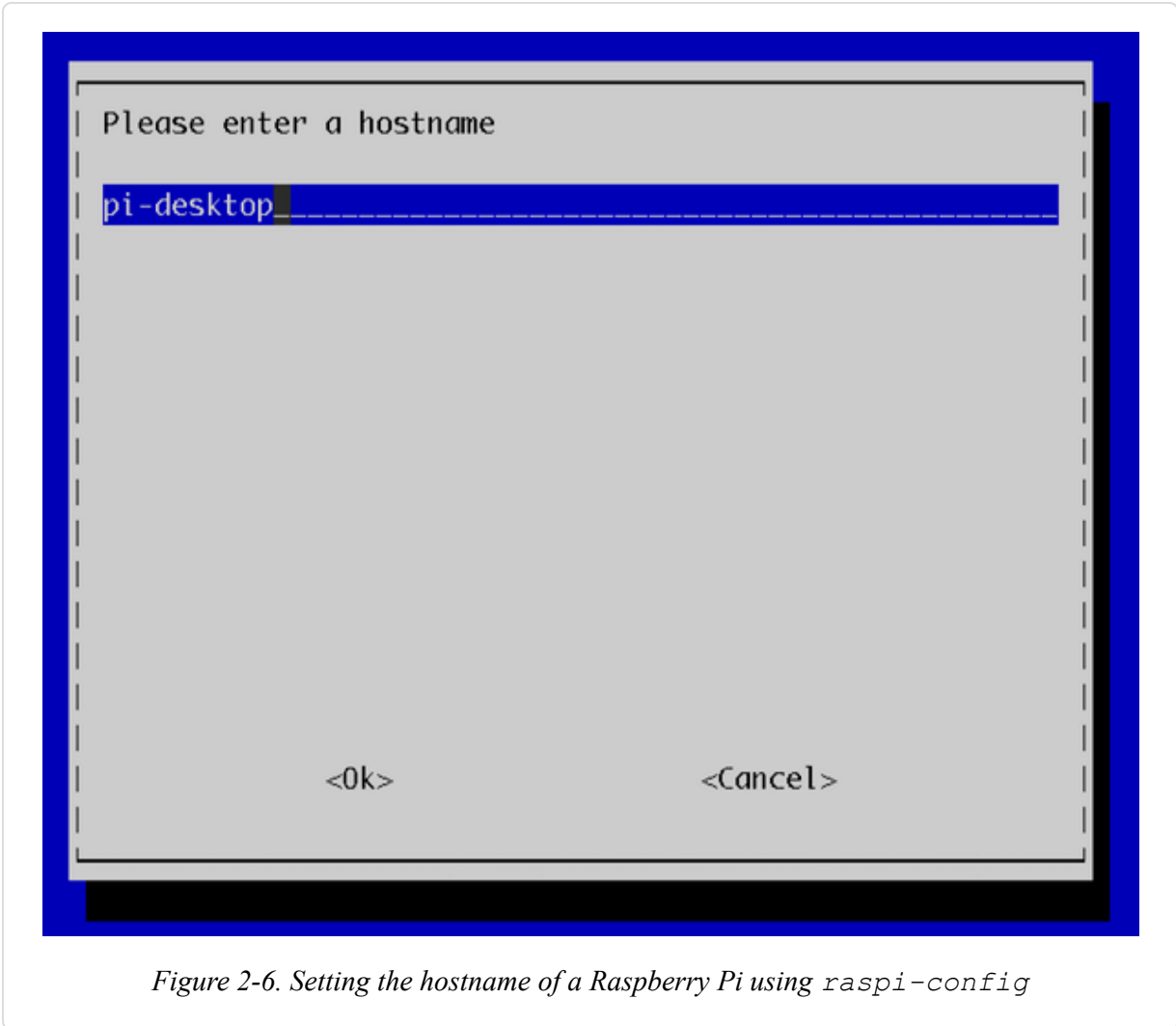


Figure 2-6. Setting the hostname of a Raspberry Pi using `raspi-config`

### Setting the network name using the command line (the hard way)

If you really want to do it the hard way, you can directly edit the files that control the Raspberry Pi's network name. There are two files that you need to change.

First, edit the file `/etc/hostname`. You can do this by opening a Terminal window and typing the command:

```
$ sudo nano /etc/hostname
```

Replace `raspberrypi` with a name of your choice.

Second, open the file `/etc/hosts` in an editor using the command:

```
$ sudo nano /etc/hosts
```

The file will look something like this:

```
127.0.0.1    localhost
::1         localhost ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters

127.0.1.1   raspberrypi
```

Change the name at the end of the file (`raspberrypi`) to your preferred new name.

## Discussion

Changing the name of your Pi can be very useful, especially if you have more than one Pi connected to your network.

## See Also

See [Recipe 2.3](#) to change the IP address of your Raspberry Pi.

## 2.5 Setting Up a Wireless Connection

## Problem

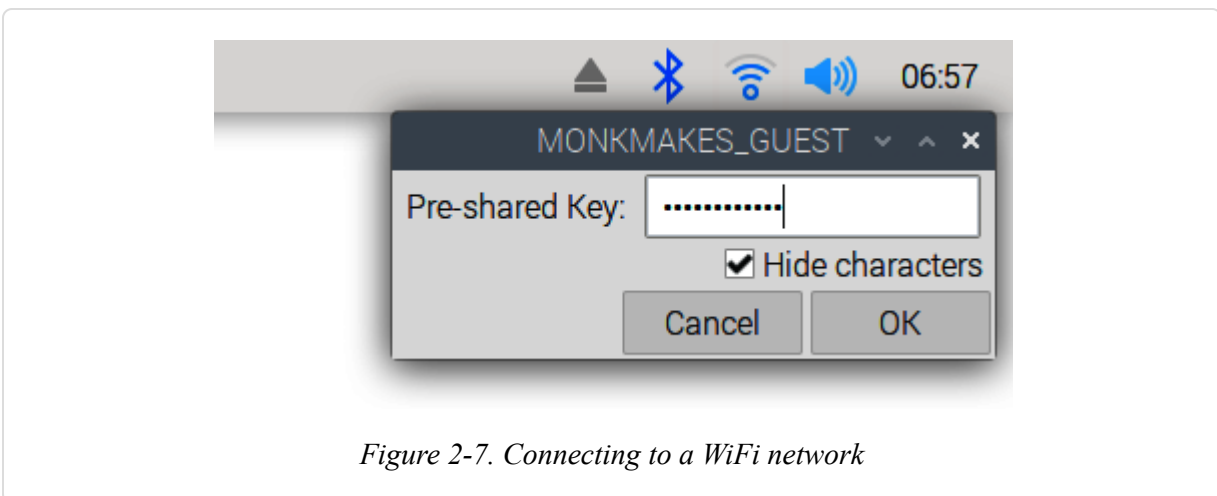
You want to connect your Raspberry Pi to the internet using WiFi.

## Solution

There are various methods of setting up a WiFi connection with your Raspberry Pi.

### Setting up WiFi from the desktop

Setting up WiFi in the latest Raspberry Pi OS is really easy. In the upper-right corner of your screen, click the network icon (the two computers). You are then presented with a list of WiFi networks. Select your network; a prompt appears in which you can enter your pre-shared key (password). Enter your password. After a while, the network icon will switch to the standard WiFi symbol, and you'll be connected ([Figure 2-7](#)).



*Figure 2-7. Connecting to a WiFi network*

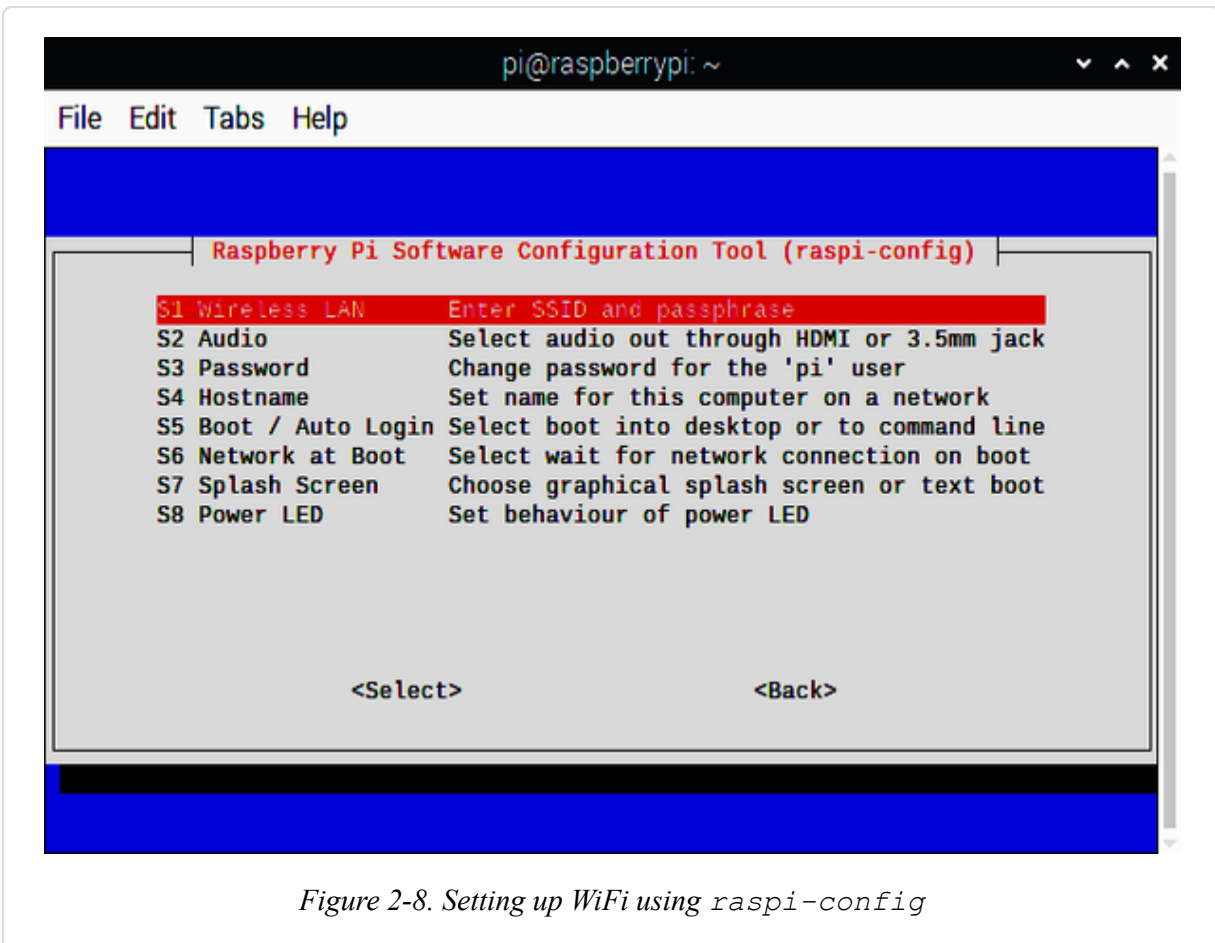
### Setting up WiFi using the command line

This method of setting up WiFi is great if you are setting up your Raspberry Pi so that, after you configure it, you can use it without keyboard and monitor attached. However, you'll need to temporarily have network access to the Raspberry Pi by connecting it to your router using an Ethernet cable ([Recipe 2.1](#)).

Run the following command to start the `raspi-config` utility:

```
$ sudo raspi-config
```

Then, from the menu that opens, select System Options (use the arrow keys and press Enter) and then select Wireless LAN (Figure 2-8).



A prompt appears, asking for the SSID (WiFi name) and password.

## Discussion

WiFi uses quite a lot of power, so if you find your Pi rebooting unexpectedly or not booting properly, you might need to use a larger power supply for it. Look for a supply that is 1.5A or higher; if you are using a Raspberry Pi 4 and are hanging high-power USB peripherals on it, use a 3A power supply.

If you are using your Raspberry Pi as a media center (see [Recipe 4.1](#)), there will be a settings page on which you can connect the media center to your network using WiFi.

## See Also

eLinux maintains a list of [WiFi adapters](#) that are compatible with the Raspberry Pi.

For more information on setting up a wired network, see [Recipe 2.1](#).

## 2.6 Connecting with a Console Lead

### Problem

No network connection is available, but you still want to be able to remotely control your Raspberry Pi from another computer.

### Solution

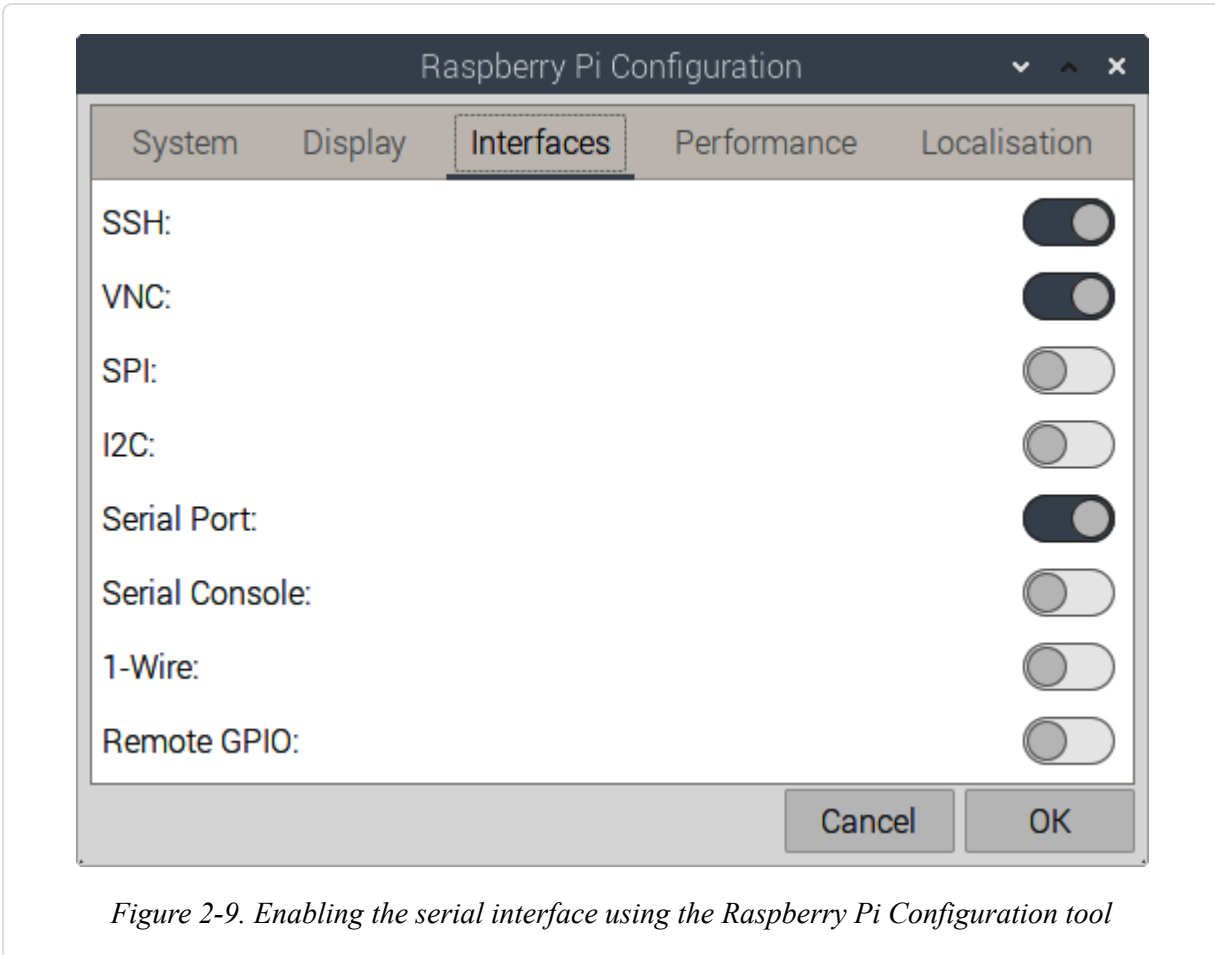
Use a *console cable* (a special lead that you need to buy separately—see “[Miscellaneous](#)”) to connect to a Raspberry Pi.

### Power Consumption

The console lead will typically only be able to supply 500mA, which is fine for early Raspberry Pis, but not enough for a Raspberry Pi 4 or 400. If you are using a 4 or 400, then you will need to use a power adapter with it ([Recipe 1.4](#)) and leave the red power lead of the console lead (described after [Figure 2-11](#)) unconnected to the 5V Raspberry Pi pin.

To use this method, you’ll need to enable the serial interface. This means that, at least while you are setting up your Raspberry Pi, you need to have a keyboard, monitor, and mouse attached.

To enable the serial interface, go to the Raspberry Menu, select Preferences, and then click Raspberry Pi Configuration. Select the Interfaces tab and then click the toggle switch for Serial Port, as shown in [Figure 2-9](#).



As with most Raspberry Pi configurations, you can also use the command-line `raspi-config` utility by running this command in a Terminal session:

```
$ sudo raspi-config
```

Select Interfacing Options and then Serial, as shown in [Figure 2-10](#).



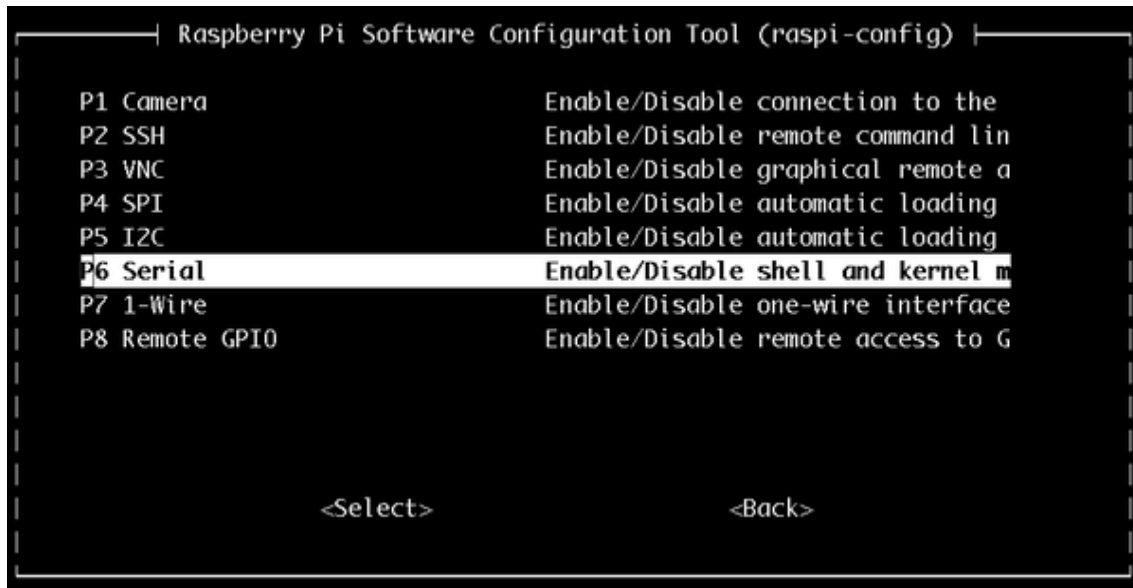
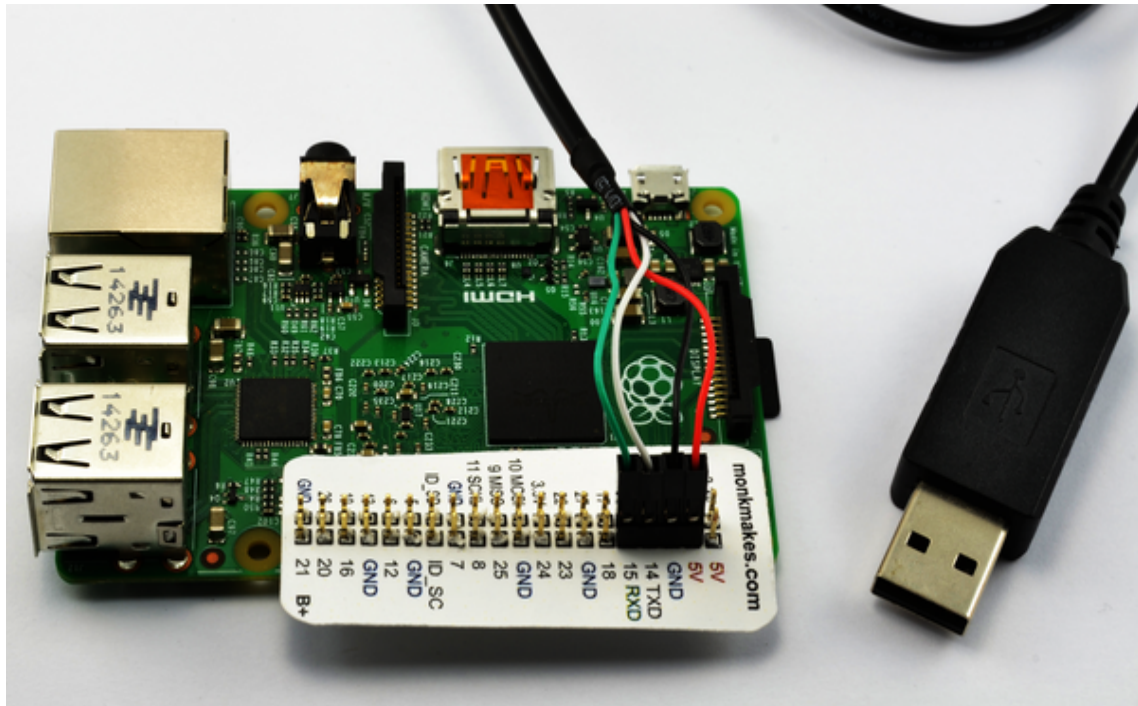


Figure 2-10. Enabling the serial interface using the *raspi-config* utility

Console cables are great for a Pi that is going to be used headless—that is, without keyboard, mouse, or monitor.

The console cable shown in [Figure 2-11](#) is available from [Adafruit](#).



*Figure 2-11. A console cable*

Connect the lead as follows and by referring to **Figure 2-11**:

1. Connect the red (5V) lead to the 5V pin, one pin to the left of the edge of the GPIO header. Note that if your console lead is a 3V lead, or you have a Raspberry Pi 4 or 400 that takes too much current for the serial lead, then leave this wire unconnected and power the Raspberry Pi as usual using its USB power connector.
2. Connect the black (GND) lead to GND on the next pin to the left on the Raspberry Pi.
3. Connect the white lead (Rx) to Raspberry Pi GPIO 14 (TXD), to the left of the black lead.
4. Connect the green lead (Tx) to GPIO 15 (RXD), to the left of the white lead.

If you use a different lead, the wire colors might well be different, so always check the documentation for the lead or else you risk damaging your Raspberry Pi.

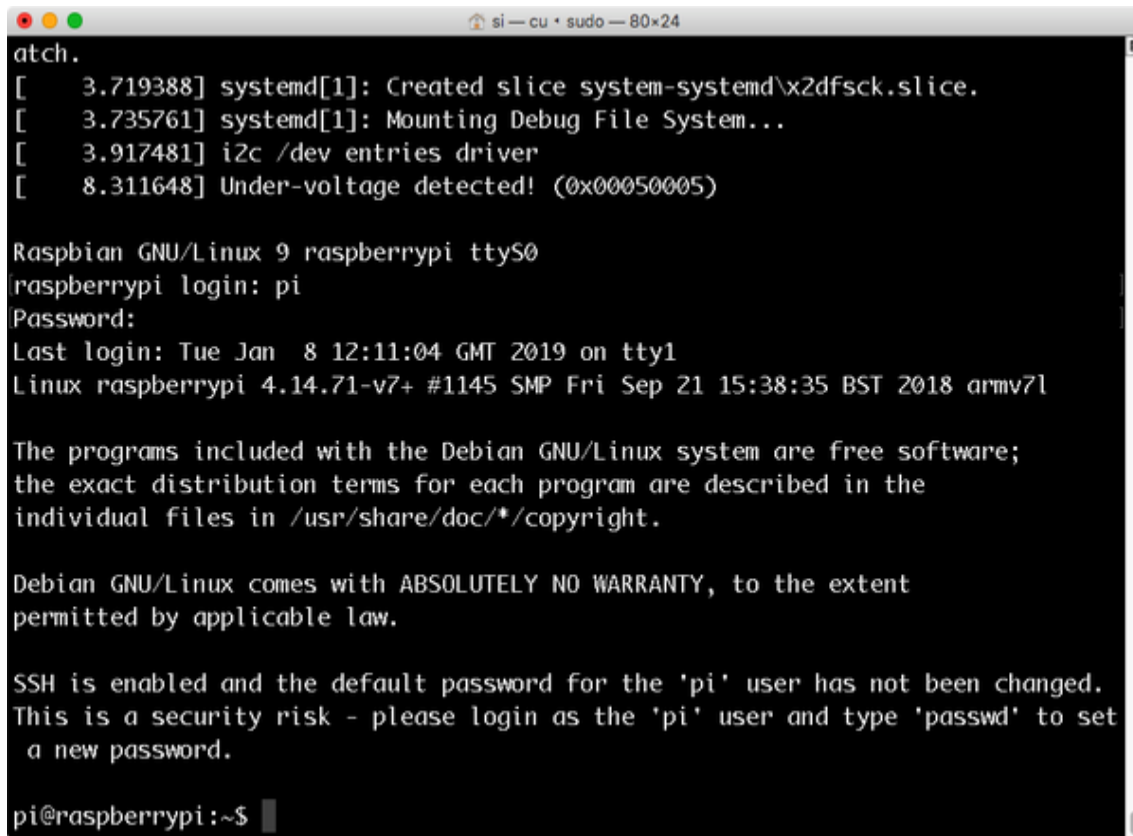
Note that the USB lead also provides 5V on the red wire, with enough power for the Pi on its own but not with a lot of devices attached.

If you are a Windows or macOS user, you will need to install drivers for the USB lead; the driver that you will need to install depends on the chip used by the console lead manufacturer. Please refer to the manufacturer's website. For a Mac, you can find out more information on that at this [Adafruit tutorial](#).

To connect to the Pi from macOS, you will need to run the Terminal and enter the following command:

```
$ sudo cu -l /dev/cu.usbserial -s 115200
```

After connecting, press Enter, and the Raspberry Pi login prompt should appear ([Figure 2-12](#)). The default username and password are *pi* and *raspberrypi*, respectively.



```
at ch.
[ 3.719388] systemd[1]: Created slice system-systemd\x2dfsck.slice.
[ 3.735761] systemd[1]: Mounting Debug File System...
[ 3.917481] i2c /dev entries driver
[ 8.311648] Under-voltage detected! (0x00050005)

Raspbian GNU/Linux 9 raspberrypi ttyS0
raspberrypi login: pi
Password:
Last login: Tue Jan  8 12:11:04 GMT 2019 on tty1
Linux raspberrypi 4.14.71-v7+ #1145 SMP Fri Sep 21 15:38:35 BST 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~$
```

*Figure 2-12. Logging in with a console cable*

If you are trying to connect to your Raspberry Pi from a Windows computer, you need to download the Terminal software called **PuTTY**.

When you run PuTTY, change the “Connection type” to Serial and set the speed to 115200. You also need to set the “Serial line” to be the COM port in use by the cable. This might be COM7, but if that doesn’t work, check it using Ports in the Windows Device Manager.

When you click Open and press Enter, the Terminal session should start with a login prompt.

## **Discussion**

The console cable can be a very convenient way of using your Pi if you are traveling light because it provides both power and a way to control the Pi remotely.

The console lead has a chip in the USB end that provides a USB-to-serial interface. This sometimes (depending on your operating system) requires the installation of drivers on your PC. You should be able to use any USB-to-serial converter as long as it has the necessary drivers for your PC.

Plugging the sockets of the lead into the right places is made easier if you carefully glue (or tape) the four sockets together in the right order so that they fit over the GPIO header in a block.

Finding the right position on the GPIO header is made easier if you use a GPIO template like the Raspberry Leaf (see [Recipe 10.1](#)). The Raspberry Pi pinouts are shown in [Appendix B](#).

## See Also

You can find out more about using the serial console at this [Adafruit tutorial](#). Adafruit also sells console cables. The lead used here is one supplied by Adafruit (product code 954).

## 2.7 Controlling the Pi Remotely with SSH

### Problem

You want to connect to a remote Pi from another computer using Secure Shell (SSH).

### Solution

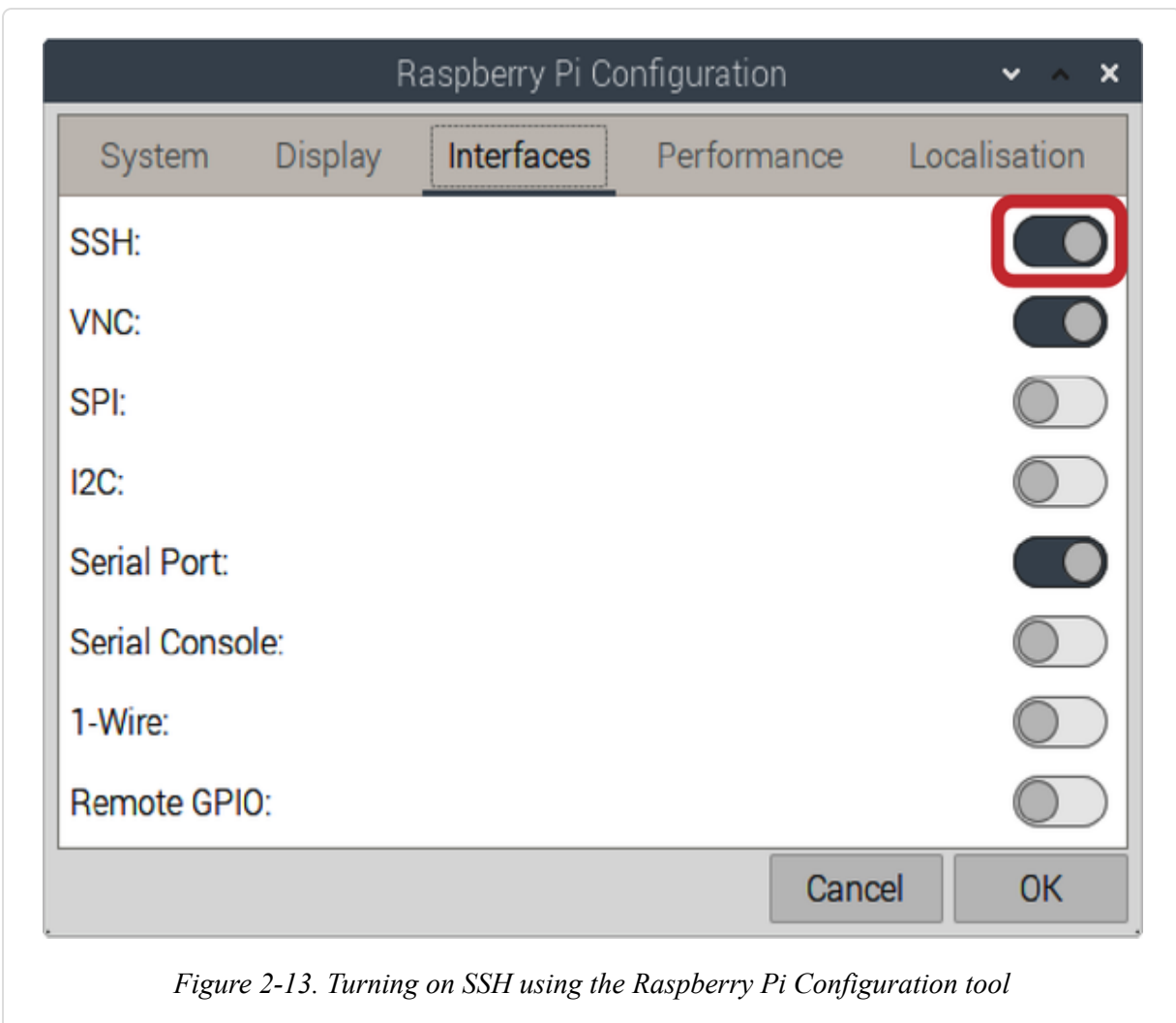
Before you can connect to your Raspberry Pi using SSH, you must turn on SSH. On newer versions of Raspberry Pi OS, you can use the Raspberry Pi Configuration tool ([Figure 2-13](#)), which you can find on the Raspberry Menu under Preferences. Just select the enabled radio button for SSH and click OK. You are then prompted to restart.

You can also enable SSH from the Raspberry Pi Imager in its Advanced options dialog ([Recipe 1.8](#)).

If you prefer to use the command line, use the `raspi-config` utility. You can start this at any time by entering the following command in a Terminal:

```
$ sudo raspi-config
```

Select the Interfaces tab, scroll down to the SSH option, and then click the Enabled button.

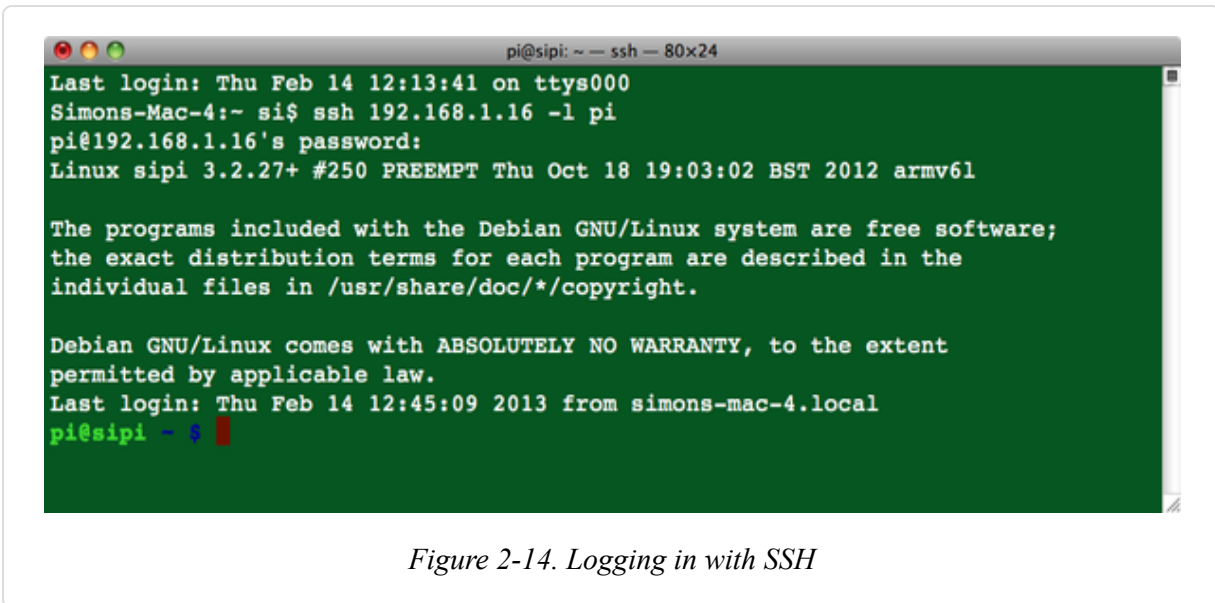


*Figure 2-13. Turning on SSH using the Raspberry Pi Configuration tool*

If you are using macOS or have Linux installed on the computer from which you want to connect your Pi, all you need to do to connect is open a Terminal window and enter the following command:

```
$ ssh 192.168.1.16 -l pi
```

Here, the IP address (192.168.1.16) is the IP address of your Pi (see [Recipe 2.2](#)). You are then prompted for your password and logged in to the Pi ([Figure 2-14](#)).



To connect from a Windows computer, you will need to use PuTTY ([Recipe 2.6](#)) to start an SSH session.

## Discussion

SSH is a very common way of connecting to remote computers; any commands that you could issue on the Pi itself, you can use from SSH. It is also, as its name suggests, secure because the communication is encrypted.

Unlike the console lead solution of [Recipe 2.6](#), SSH will only work if your Raspberry Pi is connected to the same network as the computer trying to connect to it.

Perhaps the only drawback to SSH is that it is a command-line rather than graphical environment. If you need remote access to the full Raspberry Pi desktop environment, you need to use VNC ([Recipe 2.8](#)).

## See Also

Check out this [Adafruit tutorial](#).

## 2.8 Controlling the Pi Remotely with VNC

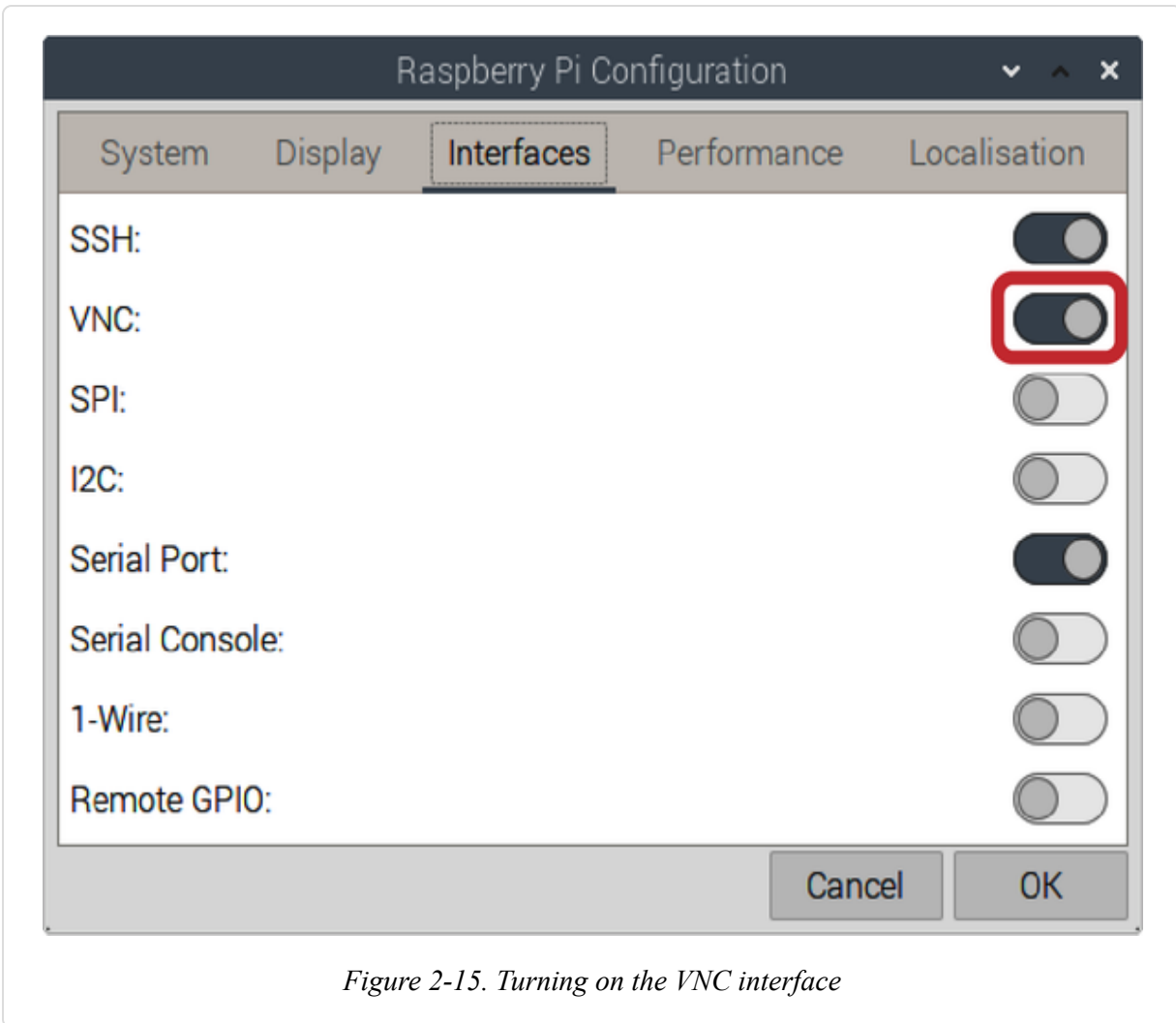
### Problem

You want access to the full Raspberry Pi OS graphical desktop from a PC (Windows or Linux) or macOS using virtual network computing (VNC).

### Solution

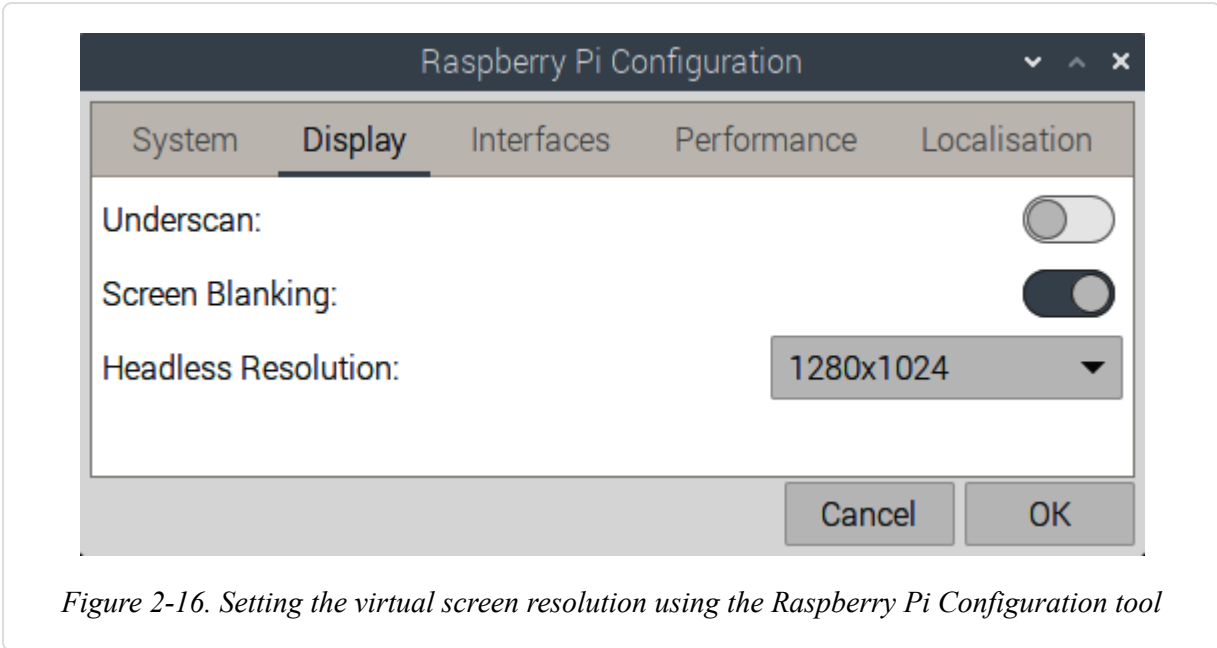
Use the pre-installed VNC software on Raspberry Pi OS. However, to do so, you first need to configure your Raspberry Pi to turn it on. You can do this using the Raspberry Pi Configuration tool, which you can find in the Preferences section of the Raspberry Menu on the Raspberry Pi desktop. Click the Interfaces tab, scroll down to the VNC option, select the enabled radio button, and click OK ([Figure 2-15](#)).





*Figure 2-15. Turning on the VNC interface*

If the Raspberry Pi doesn't have a monitor attached, you need to specify the resolution of the virtual monitor that you'll see when you connect to the Raspberry Pi from another computer using VNC. You can do this from the Display section of the Raspberry Pi Configuration tool (Figure 2-16). To avoid scrolling, select a resolution that's a bit lower than that of the monitor from which you will be viewing the virtual screen.

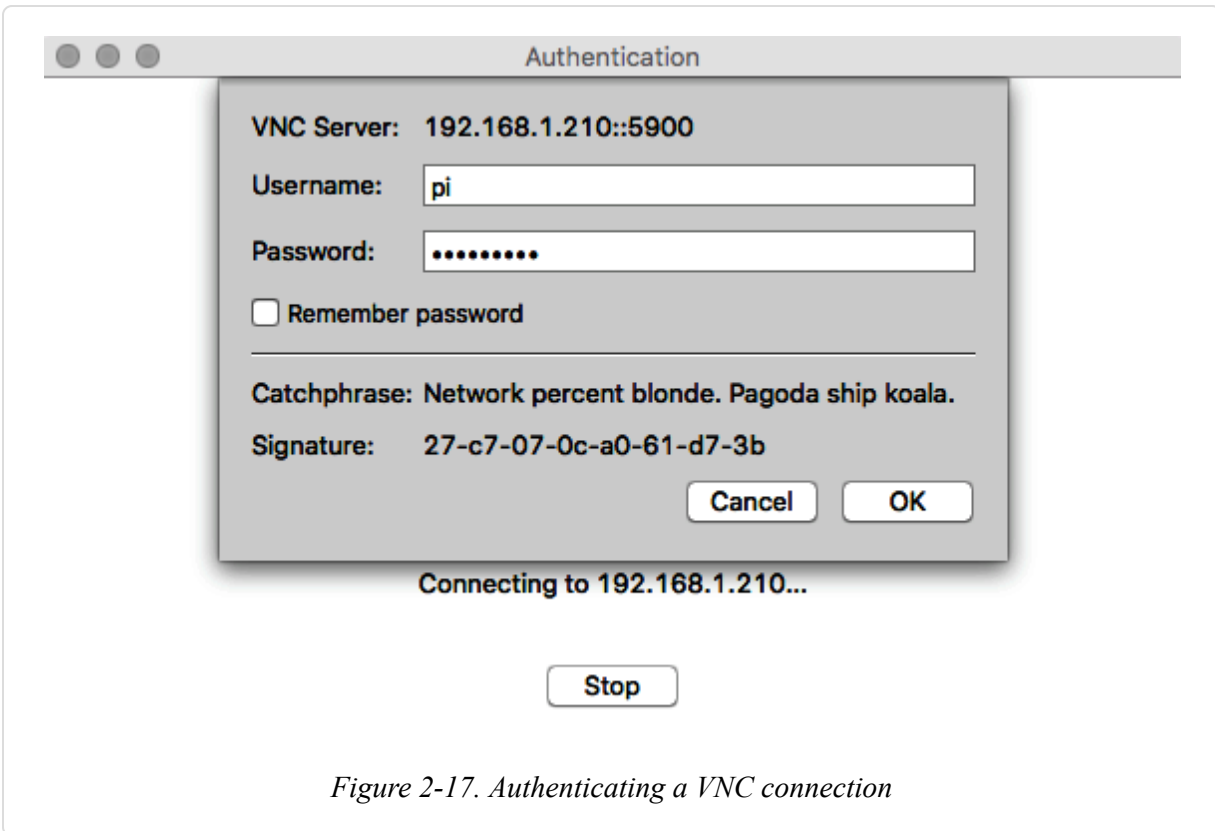


*Figure 2-16. Setting the virtual screen resolution using the Raspberry Pi Configuration tool*

To connect to the Pi from a remote computer, you need to install VNC client software. **RealVNC VNC Viewer** is a popular choice and is available for Windows, Linux, and macOS.

When you run the client program on macOS or a PC, you'll be asked to enter the IP address of the VNC server to which you want to connect (the IP address of your Pi).

You are then prompted for your password (**Figure 2-17**).



*Figure 2-17. Authenticating a VNC connection*

The catchphrase and signature are security devices designed to alert you if someone is hacking your Raspberry Pi. If either changes when you authenticate another time, your Raspberry Pi might be compromised.

## Discussion

VNC will only work if the Raspberry Pi and the remote computer are on the same network. Some VNC clients allow you to transfer files between your computer and the Raspberry Pi and to copy and paste text between them.

Although you can do most things with SSH using the command line, sometimes it is useful to have access to the graphical environment of your Raspberry Pi.

The Raspberry Pi's VNC server automatically starts when you reboot, as long as the VNC option is enabled.

## See Also

Check out this [Adafruit tutorial](#).

You can also enable VNC while setting up a new Raspberry Pi (see [Recipe 1.6](#)).

## 2.9 Using a Raspberry Pi for Network-Attached Storage

### Problem

You want to use your Raspberry Pi as network-attached storage (NAS) by accessing a large USB drive attached to your Raspberry Pi from computers on your network.

### Solution

The solution to this problem is to install and configure Samba. To do this, issue the following commands:

```
$ sudo apt update
$ sudo apt install samba
$ sudo apt install samba-common-bin
```

Now, attach the USB hard drive to the Raspberry Pi. It will automatically mount in your */media/pi* folder. To check that it's there, run this command:

```
$ cd /media/pi
$ ls
```

The drive should be listed with whatever name you gave it when you formatted it. It will automatically mount itself whenever the Raspberry Pi reboots. Make a note of this name, as you will need it in a moment.

Next, you need to configure Samba so that the drive can be shared on the network. To do this, you first need to add a Samba user (pi). Enter the

following command and type in a password:

```
$ sudo smbpasswd -a pi
New SMB password:
Retype new SMB password:
Added user pi.
```

You now need to make some changes to the file */etc/samba/smb.conf*, so enter this command:

```
$ sudo nano /etc/samba/smb.conf
```

The first line you're looking for is near the top of the file:

```
workgroup = WORKGROUP
```

You only need to change this if you plan to connect from a Windows machine. This should be the name of your Windows workgroup. For recent versions of Windows, this will be WORKGROUP. Note that generally connecting to NAS in a mixed network of Macs and Windows PCs (and, for that matter, Linux machines) works just fine.

Finally, scroll to the end of the file and add the following lines, changing NAS to the name of your USB drive that you noted earlier:

```
[USB]
path = /media/pi/NAS
comment = NAS Drive
valid users = pi
writeable = yes
browseable = yes
create mask = 0777
public = yes
```

Save the file and then restart Samba by entering the following:

```
$ sudo systemctl restart smb
```

If all is well, your USB drive should now be shared on your network.

## Discussion

To connect to the drive on macOS, select Go, and then, from the Finder menu, click Connect to Server. Next, in the Server Address field, enter **smb://raspberrypi/USB**. A login dialog box opens, in which you need to change the username to **pi** (Figure 2-18).



*Figure 2-18. Connecting to NAS with the macOS Finder*

If you are connecting to NAS from a Windows machine, the exact procedure will vary depending on your version of Windows. However, the basic principle is that at some point you will need to enter the network address, which should be **\\raspberrypi\USB** (Figure 2-19).

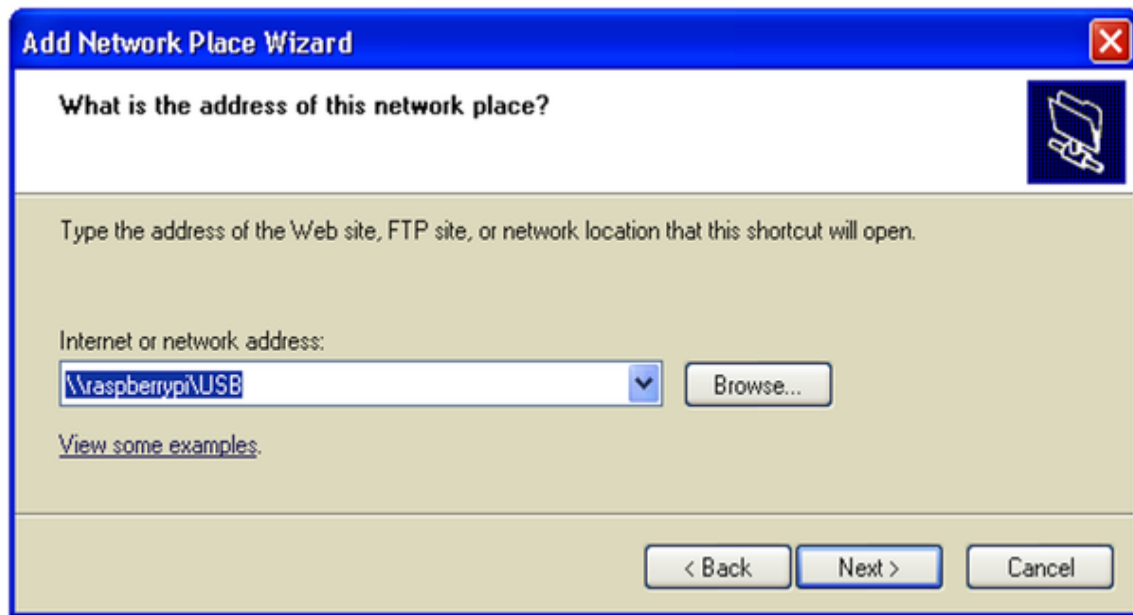


Figure 2-19. Connecting to NAS from Windows

You are then prompted for the username and password before you can use the NAS disk (Figure 2-20). You should only have to do this the first time. After the network place is added, you should be able to navigate directly to it in File Explorer.

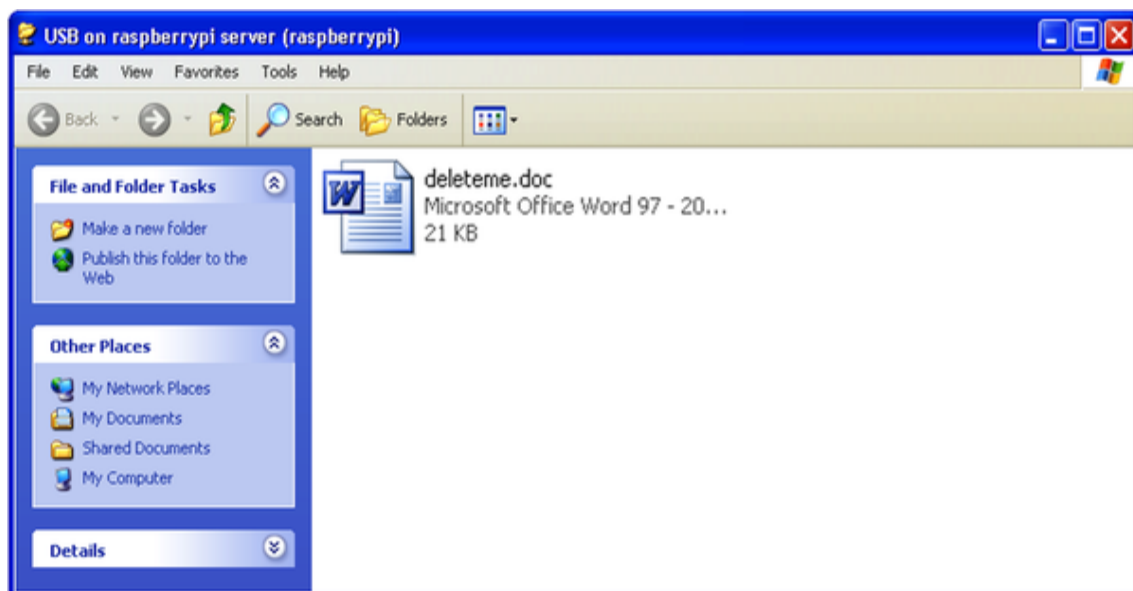


Figure 2-20. Browsing NAS on Windows

If you are a Linux user, the following command should mount the NAS drive for you:

```
$ sudo mkdir /pishare
$ sudo smbmount -o username=pi,password=raspberry
//192.168.1.16/USB /pishare
```

You should be able to connect to the Raspberry Pi using the hostname (raspberrypi), but if this does not work, try using the IP address of your Raspberry Pi, something like **smb://192.168.1.16/USB**.

## See Also

You might want to change your Raspberry Pi's network name to something inappropriate like "piNAS" (see [Recipe 2.4](#)).

## 2.10 Setting Up a Network Printer

### Problem

You want to print to a network printer from your Raspberry Pi.

### Solution

Use Common Unix Printing System (CUPS) software.

Start by entering the following commands into a Terminal to install CUPS (this can take some time):

```
$ sudo apt update
$ sudo apt install cups
```

Give yourself admin privileges for CUPS by entering the following command:



```
$ sudo usermod -a -G lpadmin pi
```

This last command adds the `lpadmin` group used by CUPS to the user `pi` so that you have permission to print.

CUPS is configured via a web interface, so start your Chromium web browser from the Raspberry Pi Menu and then enter the address **`http://localhost:631`** into the address bar.

On the Administration tab, choose the Add Printer option. This displays a list of printers that are on the network or are connected directly to the Raspberry Pi's USB port (Figure 2-21).

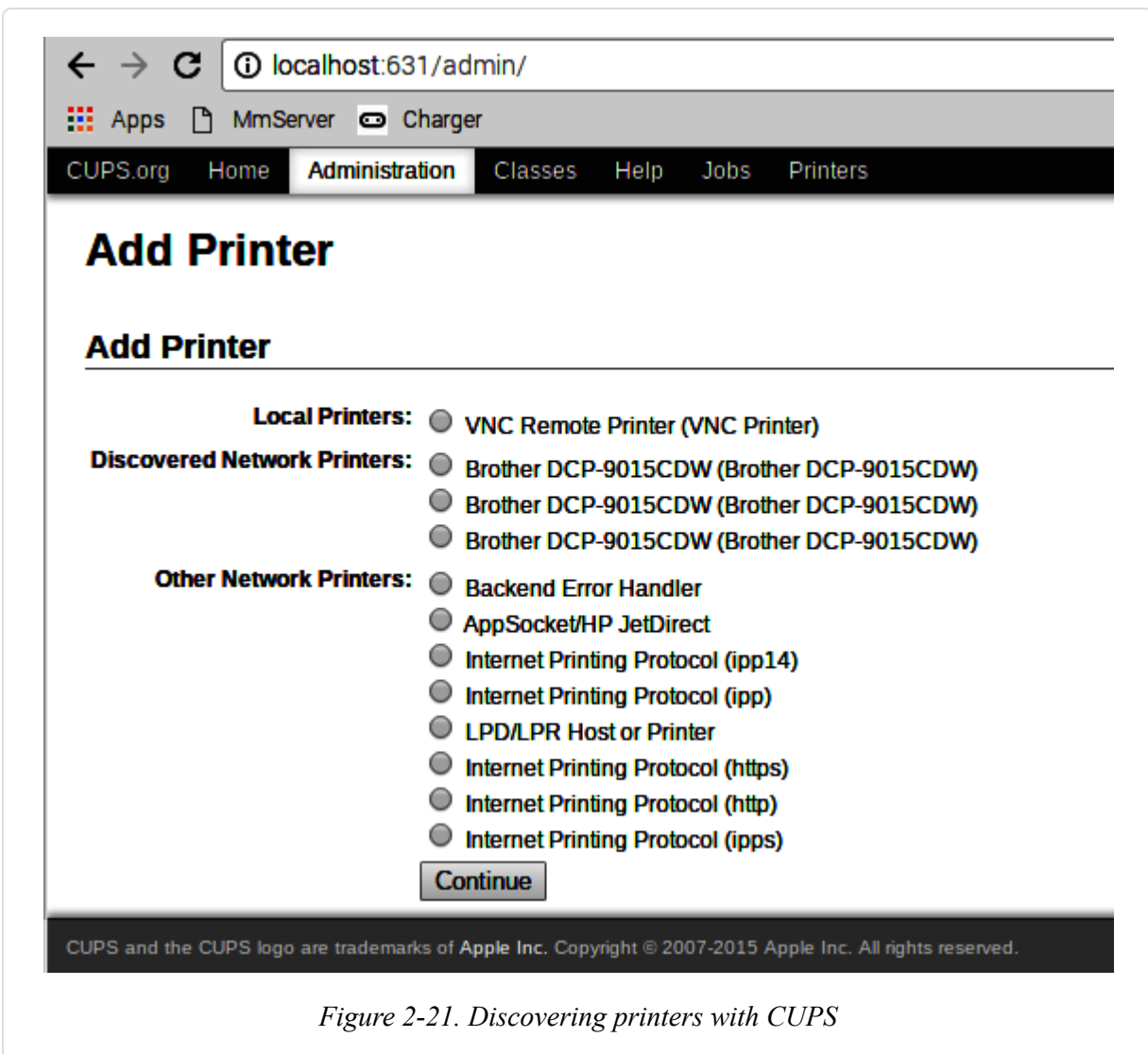
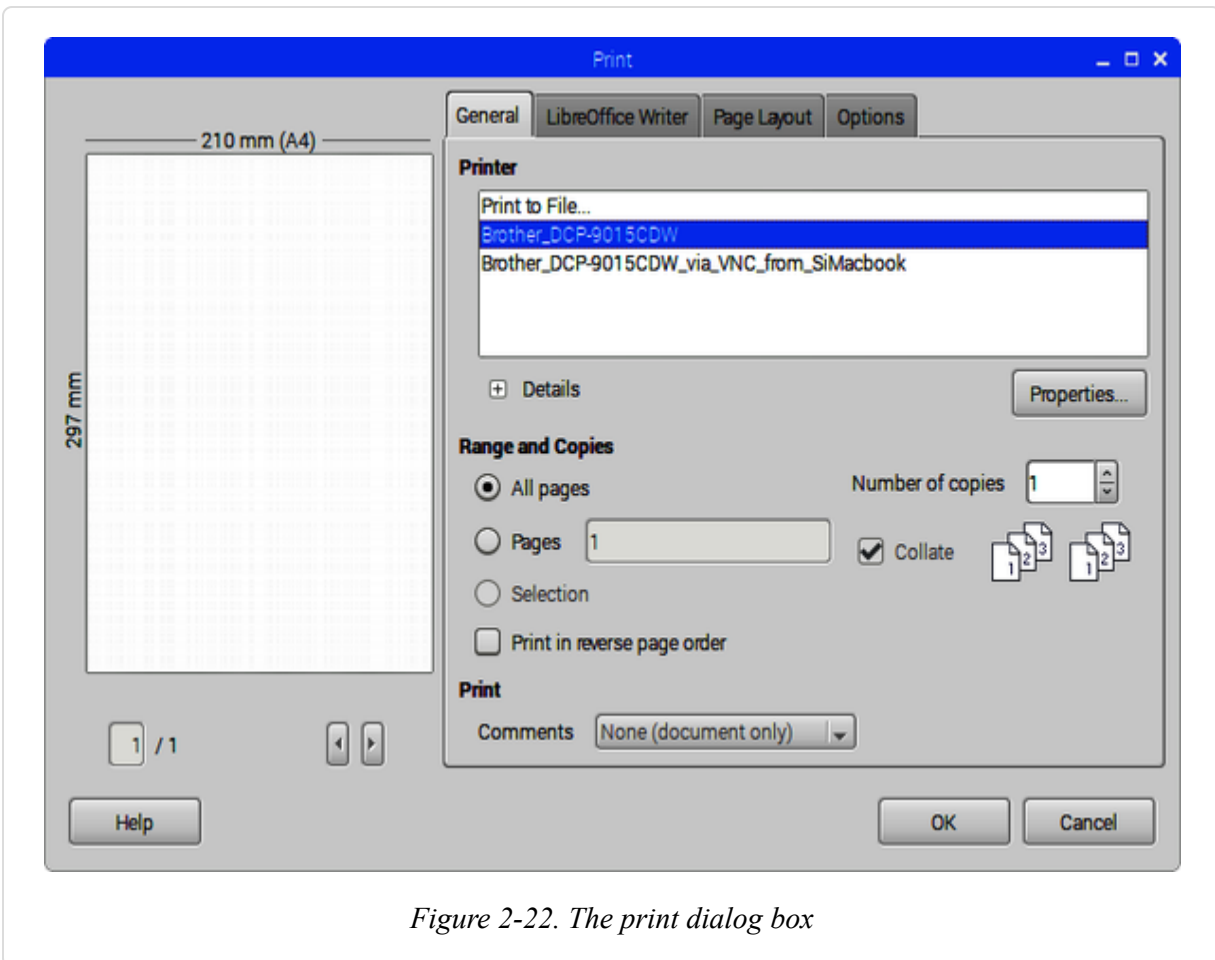


Figure 2-21. Discovering printers with CUPS

Follow the series of dialog boxes to set up the printer.

## Discussion

When you're finished, you can test out the printer by firing up LibreOffice ([Recipe 4.2](#)). Type some text, and when you go to print it, you should see your newly added printer available for printing ([Figure 2-22](#)).



## See Also

Visit the official [CUPS website](#).

# Chapter 3. Operating System

---

## 3.0 Introduction

This chapter explores many aspects of the Linux operating system used by the Raspberry Pi. A lot of this involves the use of the command line. If you are used to Windows or macOS, this can come as a bit of a shock. However, when you get used to it, doing things with the command line can be surprisingly effective.

You can accomplish many simple file operations like moving files around, renaming, copying, and deleting files graphically using a more Windows or macOS approach, and this is the subject of our first recipe.

## 3.1 Browsing Files Graphically

### Problem

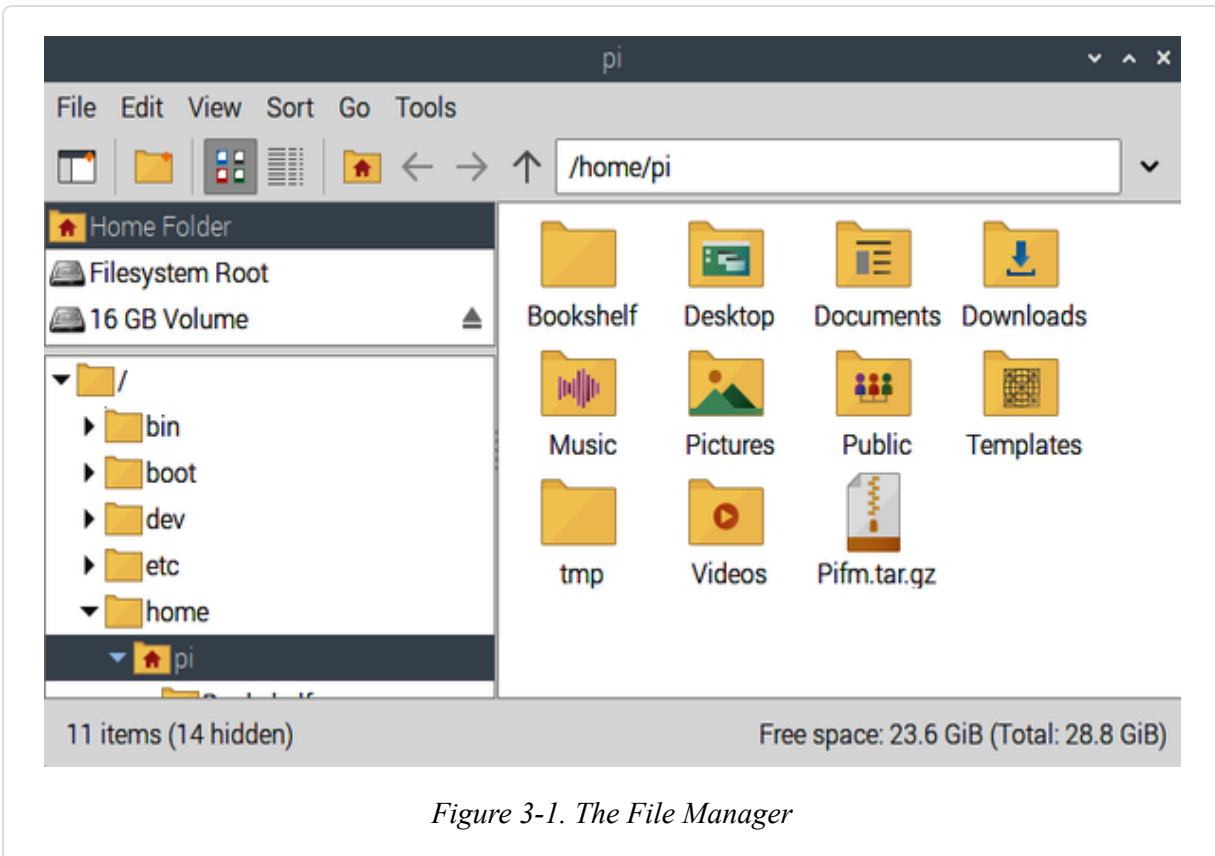
You want to move files around using a graphical interface like you can on a macOS-based machine or Windows PC.

### Solution

Use the File Manager.

You can find this program on the Raspberry Menu, in the Accessories group ([Figure 3-1](#)).

Using the File Manager, you can drag a file or directory from one directory to another or use the Edit menu to copy a file from one location and paste it to a second. This operates in much the same way as the Windows File Explorer or macOS Finder.



*Figure 3-1. The File Manager*

## Discussion

The lefthand side of the File Manager shows the folder structure.

The central area displays the files in the current folder, which you can navigate using the buttons in the toolbar or by typing a location in the filepath area at the top.

You can right-click a file to open a menu with options you can use on that file ([Figure 3-2](#)).

You can also select more than one file at a time for copying or dragging by holding down the Ctrl key while you select files, or you can select a range of files by selecting one file and then holding down the Shift key while you select the last file in the range.

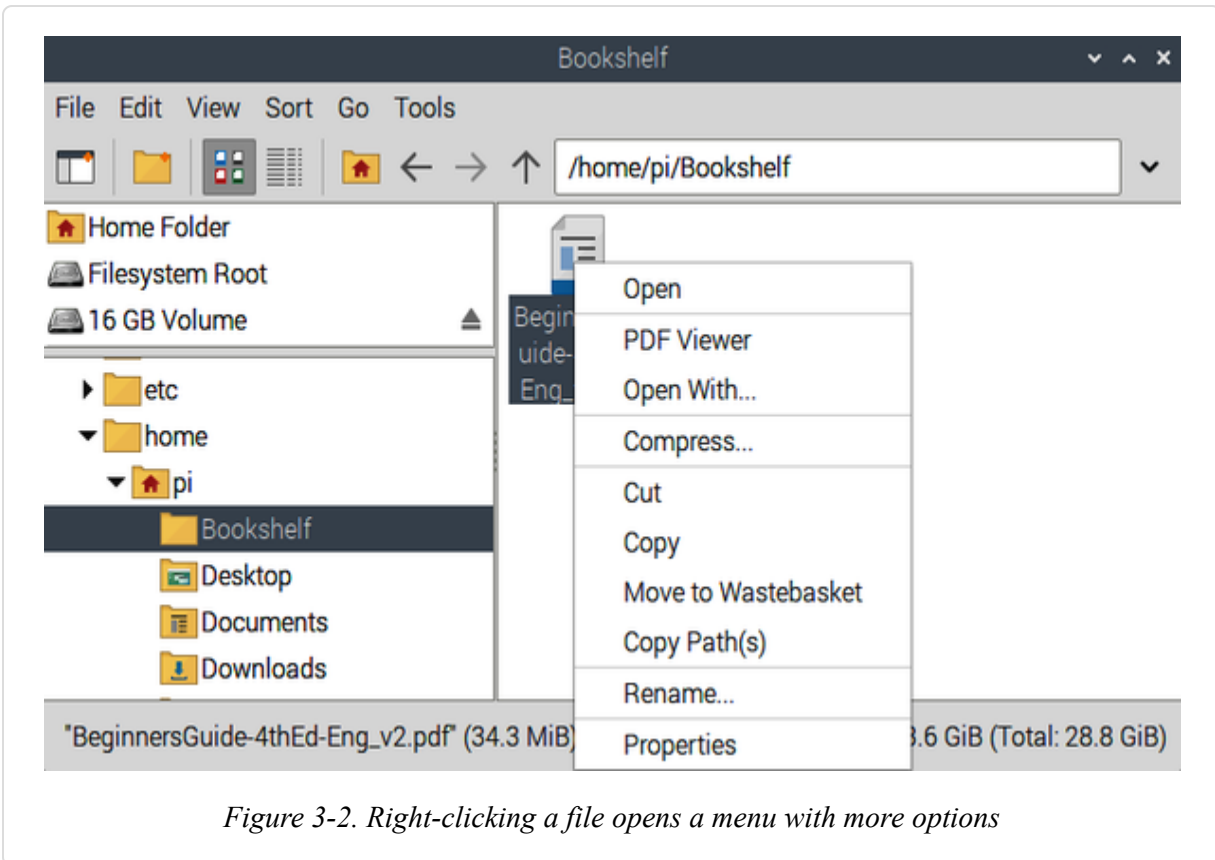


Figure 3-2. Right-clicking a file opens a menu with more options

## See Also

To rename a file or folder, see [Recipe 3.6](#).

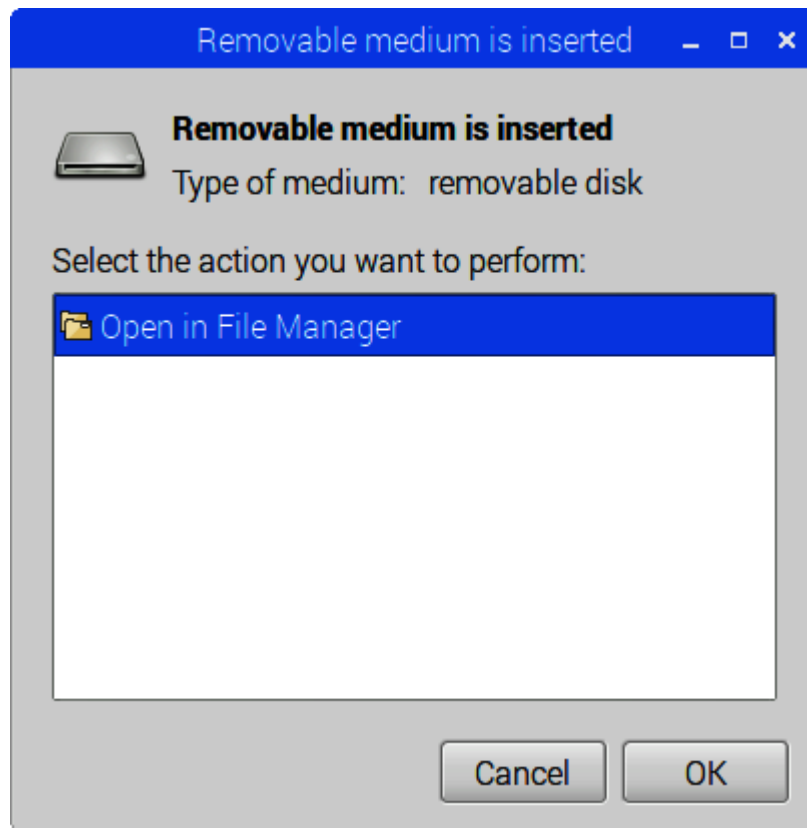
## 3.2 Copying Files onto a USB Flash Drive

### Problem

You want to copy a file from your Raspberry Pi onto a USB flash drive.

### Solution

Insert the USB flash drive into a USB port, and the dialog shown in [Figure 3-3](#) should appear. Select OK to open it in the File Manager.



*Figure 3-3. The removable-media dialog box*

The drive will be mounted in `/media/pi` followed by the name of the flash drive (in my case, UNTITLED). To copy a file from your home folder, drag it onto the folder representing your flash drive, as shown in [Figure 3-4](#).

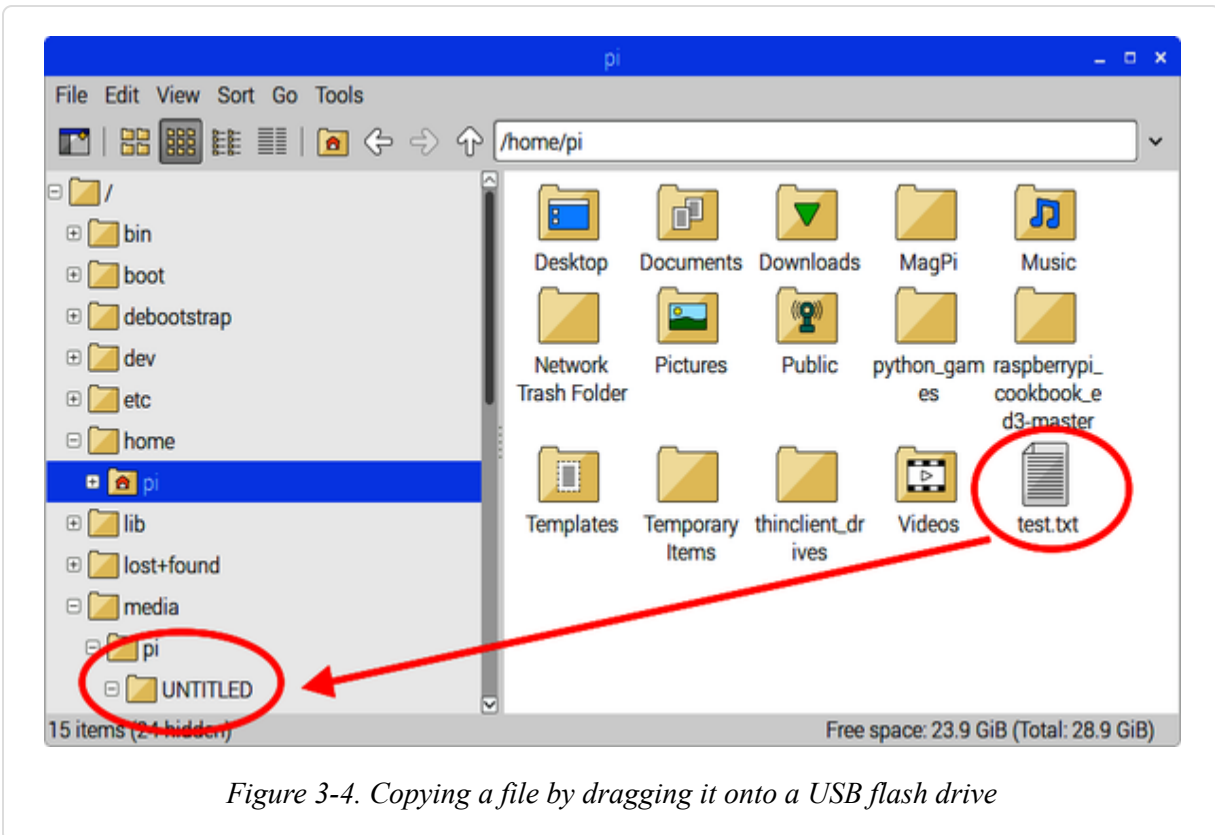


Figure 3-4. Copying a file by dragging it onto a USB flash drive

Windows, macOS, and Linux all have their own disk formats. The USB flash drive should be formatted as FAT32 or exFAT for maximum compatibility with macOS and Windows computers. exFAT supports larger disk sizes than FAT32.

## Discussion

After your USB flash drive is mounted on the Raspberry Pi's filesystem, you can also copy files using the command line. The following example copies the file *test.txt* to the flash drive:

```
$ cd /home/pi
$ cp test.txt /media/pi/UNTITLED/
```

In this example, `cd` is the command to change your current directory, and `cp` is the copy command. These commands are explained more fully in Recipes [3.4](#) and [3.5](#).

## See Also

For general information on using the File Manager, see [Recipe 3.1](#).

To copy files from the command line, see [Recipe 3.4](#).

## 3.3 Starting a Terminal Session

### Problem

When using a Raspberry Pi, you need to issue text commands in a Terminal.

### Solution

At the top of the Raspberry Pi desktop, select the Terminal icon (it looks like a black computer monitor), or, on the Raspberry Menu, in the Accessories group, select the Terminal option ([Figure 3-5](#)).



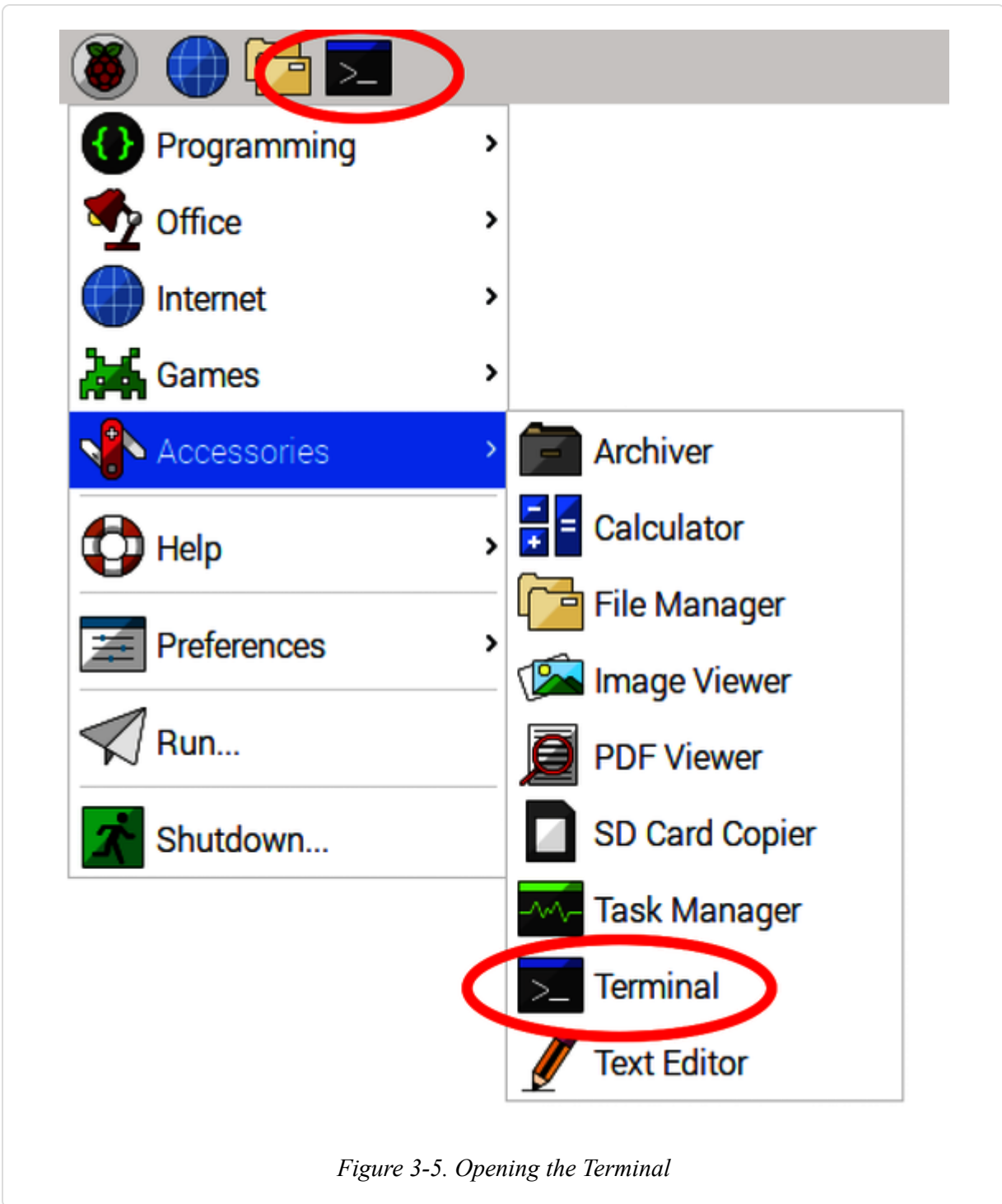


Figure 3-5. Opening the Terminal

## Discussion

When the Terminal starts, it is set to your home directory (*/home/pi*).

You can open as many Terminal sessions as you want. It is often useful to have a couple of sessions open in different directories so that you don't need to constantly switch directories using `cd` ([Recipe 3.4](#)).

When using the Terminal, everything is case sensitive. That is, if you are using a command, you must use the correct case when you're typing. For example, the `ls` command that you will meet in the next recipe must be typed as `ls` and not `LS` or `Ls` or `lS`. Similarly, all filenames are case sensitive, so files named *picture.jpg* and *Picture.jpg* are two different files.

## See Also

In the next section ([Recipe 3.4](#)), you will look at navigating the directory structure using the Terminal.

# 3.4 Navigating the Filesystem Using a Terminal

## Problem

You need to know how to change directories and move about the filesystem using the Terminal.

## Solution

The main command used for navigating the filesystem is `cd` (change directory). After `cd`, you need to specify the directory that you want to change to. This can be either a *relative* path to a directory within your current directory or an *absolute* path to somewhere else on the filesystem.

To see what the current directory is, use the command `pwd` (print working directory).

## Discussion

Try out a few examples. Open a Terminal session, and you should see a prompt like this:

```
pi@raspberrypi: ~ $
```

The prompt that you see after each command (`pi@raspberrypi: ~ $`) is a reminder of your username (`pi`) and your computer name (`raspberrypi`). The `~` character is shorthand for your home directory (`/home/pi` or whatever username you chose when setting up your Raspberry Pi). So at any point you can change your current directory to your home directory as follows:

```
$ cd ~
```

### TIP

Throughout the book, I use a `$` at the beginning of each line where you are expected to type a command. This is called the *prompt*. The response from the command line is not prefixed by anything; it appears just as it does on the Raspberry Pi's screen.

You can confirm that the command did indeed set the directory to *home* by using the `pwd` command:

```
$ pwd
/home/pi
```

If you want to move up one level in the directory structure, you can use the special value `..` (two dots) after the `cd` command, as shown here:

```
$ cd ..
$ pwd
/home
```

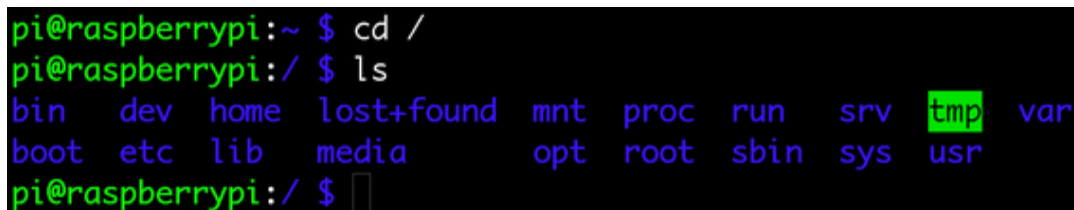
As you might have deduced by now, the path to a particular file or directory is made up of words separated by a `/`. So the very root of the entire filesystem is `/`, and to access the home directory within `/`, you would refer to `/home/`. Then, to find the `pi` directory within that, you would use `/home/pi/`. You can omit the final `/` from a path.

Paths can be absolute (starting with a `/` and specifying the full path from the root), or they can be *relative* to the current working directory, in which case they must not start with a `/` and they assume that the current directory is the starting point. For instance, starting at the root directory (`/`) you could use relative paths to navigate to your home directory (assuming your username is “pi”) like this:

```
$ cd /
$ pwd
/
$ cd home/pi
$ pwd
/home/pi
```

You will have full read and write access to the files in your home directory, but when you move into the places where system files and applications are kept, your access to some files will be restricted to read-only. You can override this ([Recipe 3.12](#)), but some care is required.

Check out the *root* of the directory structure by entering the commands `cd /` and `ls`, as shown in [Figure 3-6](#).

A terminal window showing the execution of 'cd /' and 'ls' commands. The output of 'ls' lists the root directory contents: bin, dev, home, lost+found, mnt, proc, run, srv, tmp, var, boot, etc, lib, media, opt, root, sbin, sys, usr. The 'tmp' directory is highlighted in green in the original image.

```
pi@raspberrypi:~ $ cd /
pi@raspberrypi:/ $ ls
bin  dev  home  lost+found  mnt  proc  run  srv  tmp  var
boot  etc  lib  media      opt  root  sbin  sys  usr
pi@raspberrypi:/ $
```

Figure 3-6. Listing the contents of a directory

The `ls` command (list) shows us all of the files and directories below (`/`) the root directory. You will see a *home* directory listed, which is the directory you have just come from.

Now change into one of those directories using the commands shown in [Figure 3-7](#).



```
pi@raspberrypi:/$ cd lib
pi@raspberrypi:/lib$ ls
arm-linux-gnueabihf  firmware          ld-linux.so.3      lsb                terminfo
console-setup        ifupdown          ld-linux-armhf.so.3  modprobe.d        udev
cpp                  init              libnih-dbus.so.1    modules
crda                  klibc-z-GH4vbo1mvhrmAe8pZWxAgeP0.so  libnih.so.1        resolvconf
dhcpcd                ld-linux-armhf.so.3  libnih.so.1.0.0    systemd
```

*Figure 3-7. Changing directory and listing the contents*

You will see that the files and folders have some color coding. Files are displayed in various colors, whereas folders are dark blue.

Unless you particularly like typing, the Tab key offers a convenient shortcut. If you start typing the name of a file, pressing the Tab key allows the autocomplete feature to attempt to complete the filename. For example, if you're going to change directory to *network*, type the command `cd netw` and then press the Tab key. Because *netw* is enough to uniquely identify the file or directory, pressing the Tab key will autocomplete it.

If what you have typed is not enough to uniquely identify the file or directory, pressing the Tab key another time will display a list of possible options that match what you have typed so far. So if you had stopped at *ne* and pressed the Tab key, you would see something like [Figure 3-8](#).

```
pi@raspberrypi:/lib $ cd /etc/ne
network/ newt/
pi@raspberrypi:/lib $ cd /etc/ne
```

Figure 3-8. Autocompletion using the Tab key

You can provide an extra argument after `ls` to narrow down the things that you want to list. Change directory to `/etc` and then run the following:

```
$ ls f*
fake-hwclock.data  fb.modes  fstab  fuse.conf

fonts:
conf.avail  conf.d  fonts.conf  fonts.dtd

foomatic:
defaultspooler  direct  filter.conf

fstab.d:
pi@raspberrypi /etc $
```

The `*` character is called a *wildcard*. In specifying `f*` after `ls`, we are saying that we want to list everything that begins with an `f`.

Helpfully, the results first list all the files within `/etc` that begin with `f`, and then the contents of all the directories in that folder beginning with `f`.

A common use of wildcards is to list all files with a certain extension (e.g., `ls *.docx`).

A convention in Linux (and many other operating systems) is to prefix with a period any files that should be hidden from the user. Any so-named files or folders will not appear when you type `ls` unless you supply `ls` with the option `-a` (all).

For example:

```

$ cd ~
$ ls -a
.                Desktop                .pulse
..              .dillo                .pulse-
cookie
Adafruit-Raspberry-Pi-Python-Code .dmrc
python_games
.advance        .emulationstation
sales_log
.AppleDB        .fltk                  servo.py
.AppleDesktop  .fontconfig           .stella
.AppleDouble   .gstreamer-0.10
stepper.py.save
Asteroids.zip  .gvfs
switches.txt.save
atari_roms     indiecity
Temporary Items
.bash_history  .local
thermometer.py
.bash_logout   motor.py
.thumbnails
.bashrc        .mozilla               .vnc
.cache         mydocument.doc
.Xauthority
.config        Network Trash Folder
.xsession-errors
.dbus          .profile
.xsession-errors.old

```

As you can see, the majority of the files and folders in your home directory are hidden.

## See Also

To change file permissions, see [Recipe 3.14](#).

## 3.5 Copying a File or Folder

### Problem

You want to copy a file using a Terminal session.

## Solution

Use the `cp` command to copy files and directories.

## Discussion

You can, of course, copy files using the File Manager and its copy and paste menu options ([Recipe 3.1](#)) or keyboard shortcuts.

The simplest example of copying in a Terminal session is to make a copy of a file within your working directory. The `cp` command is followed first by the file to copy and then by the name to be given to the new file.

For example, the following code creates a file called *myfile.txt* and then makes a copy of it with the name *myfile2.txt*; you can find out more about the trick of creating a file using the `>` command in [Recipe 3.9](#):

```
$ echo "hello" > myfile.txt
$ ls
myfile.txt
$ cp myfile.txt myfile2.txt
$ ls
myfile.txt  myfile2.txt
```

Although in this example both filepaths are local to the current working directory, the filepaths can be to anywhere in the filesystem where you have write access. The following example copies the original file to an area named */tmp*, which is a location for temporary files (do not put anything important in that folder):

```
$ cp myfile.txt /tmp
```

Note that in this case, the name to be given to the new file is not specified, just the directory where it is to go. This will create a copy of *myfile.txt* in */tmp* with the same name of *myfile.txt*.

Sometimes, rather than copying just one file, you might want to copy an entire directory full of files and possibly other directories. To copy a



directory and all its contents, you need to use the `-r` option (for recursive):

```
$ cp -r mydirectory mydirectory2
```

Whenever you are copying files or folders, the result of the command will tell you if you do not have permission. If that's the case, you will need to either change the permissions of the folder into which you are copying ([Recipe 3.14](#)) or copy the files with superuser privileges ([Recipe 3.12](#)).

## See Also

You can also rename files rather than copy them; see [Recipe 3.6](#).

For a useful description of the many optional parameters to the `cp` command, see <https://oreil.ly/Cq2SJ>.

## 3.6 Renaming a File or Folder

### Problem

You need to rename a file using a Terminal session.

### Solution

Use the `mv` command to rename files and directories.

### Discussion

The `mv` (move) command is used in a similar way to the `cp` command, except that the file or folder being moved is simply renamed rather than a duplicate being made.

For example, to rename a file from *my\_file.txt* to *my\_file.rtf*, you use the following command:

```
$ mv my_file.txt my_file.rtf
```

Changing a directory name is just as straightforward, and you don't need the recursive `-r` option you used when copying, because changing a directory's name implicitly means that everything within it is contained in a renamed directory.

## See Also

To copy a file or folder, see [Recipe 3.5](#).

## 3.7 Editing a File

### Problem

You want to run an editor from the command line to change a configuration file.

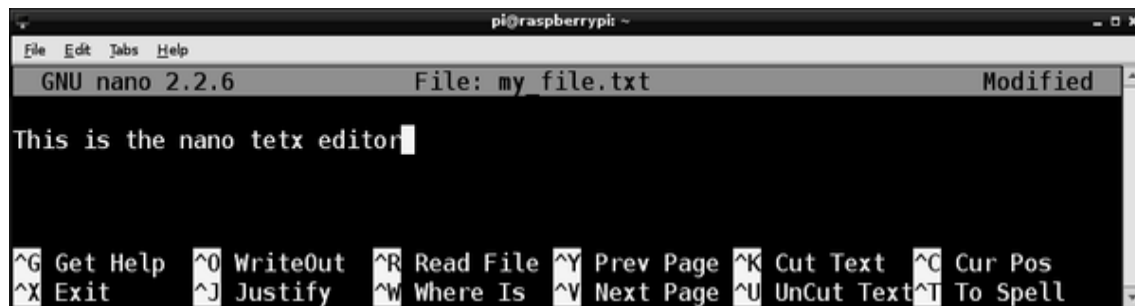
### Solution

Use the nano editor included with most Raspberry Pi distributions.

### Discussion

To use nano, simply type the command **nano** followed by the filename or path to the file that you want to edit. If the file doesn't exist, it will be created when you save it. However, this will happen only if you have write permissions in the directory to which you are trying to write the file.

From your home directory, type the command **nano my\_file.txt** to edit or create the file *my\_file.txt*. [Figure 3-9](#) shows nano in action.



*Figure 3-9. Editing a file with nano*

You cannot use the mouse to position the cursor; you must use the arrow keys instead.

The area at the bottom of the screen lists a number of commands that you can access by holding down the Ctrl key and pressing the letter indicated. Most of these are not that useful. The ones that you are most likely to use are as follows:

#### Ctrl-X

Exit. You will be prompted to save the file before nano exits.

#### Ctrl-V

Next page. Think of it as an arrow pointing downward. This allows you to move through a large file one screen at a time.

#### Ctrl-Y

Previous page.

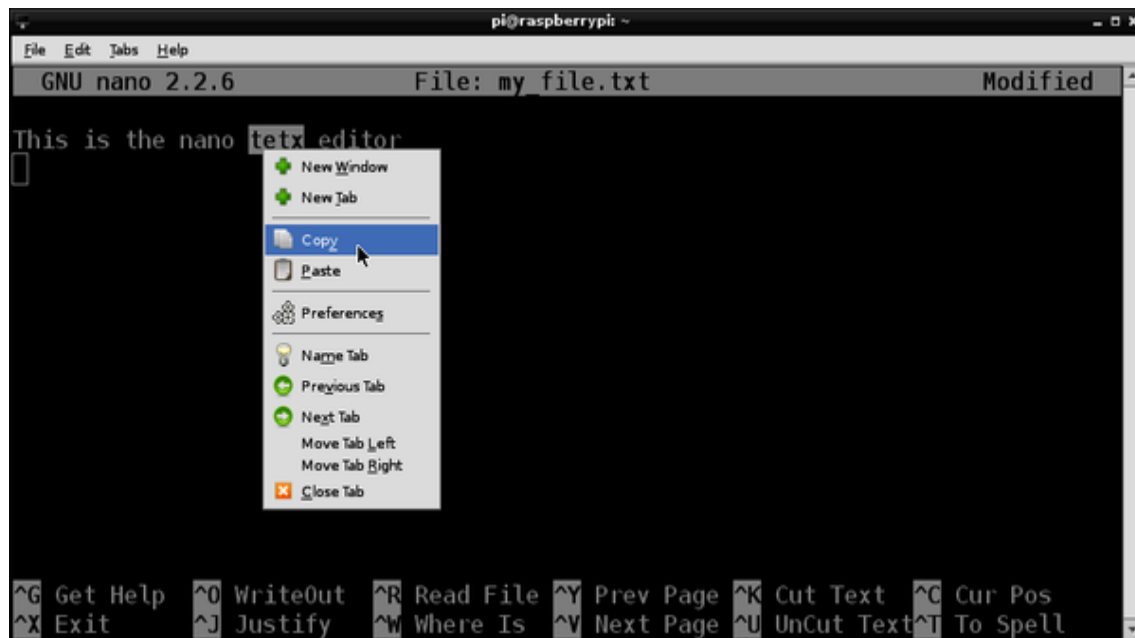
#### Ctrl-W

Where is. This allows you to search for a piece of text.

#### Ctrl-O

Output. This will write (save) the file without exiting the editor.

Some fairly crude copy-and-paste options are also there, but in practice, it's easier to use the normal clipboard from the menu that you access with a right-click, as demonstrated in [Figure 3-10](#).



*Figure 3-10. Using the clipboard in nano*

Using this clipboard also enables you to copy and paste text between other windows, such as your browser.

When you're ready to save your changes to the file and exit nano, use the command `Ctrl-X`. Type `Y` to confirm that you want to save the file. nano then displays the filename as the default name to save the file under; press `Enter` to save and exit.

If you want to abandon the changes you have made, enter `N` in place of `Y`.

## See Also

Editors are very much a matter of personal taste. Many other editors that are available for Linux will work just fine on Raspberry Pi. The **Vim (Vi Improved) editor** has many fans in the Linux world. This is also included in the popular Raspberry Pi distributions. It is not, however, an easy editor for the beginner. You can run it in the same way as nano, but you use the command `vi` instead of `nano`.

## 3.8 Viewing the Contents of a File

### Problem

You want to view the contents of a small file without editing it.

### Solution

Use the `cat` command or the `more` command to view the file.

For example:

```
$ more myfile.txt
This file contains
some text
```

### Discussion

The `cat` command displays the whole contents of the file, even if the contents are longer than will fit on the screen.

The `more` command displays only one screen of text at a time. Press the space bar to display the next screen.

### See Also

You can also use `cat` to concatenate (join together) a number of files ([Recipe 3.32](#)).

Another popular command related to `more` is `less`. `less` is like `more` except that it allows you to move backward in the file as well as forward.

## 3.9 Creating a File Without Using an Editor

### Problem

You want to create a one-line file without having to use an editor.

## Solution

Use the `>` and `echo` commands to redirect what you type on the command line to a file.

For example:

```
$ echo "file contents here" > test.txt
$ more test.txt
file contents here
```

### WARNING

The `>` command overwrites any existing file with the same name, so use it with caution.

## Discussion

If you just want to create an empty file and edit it later, you can use the `touch` command followed by a filename, like this:

```
$ touch test.txt
```

If you use the `touch` command on a file that already exists, it will change the modified timestamp on the file as if you had just edited it.

## See Also

To use the `more` command to view files without using an editor, see [Recipe 3.8](#).

To use `>` to capture other kinds of system output, see [Recipe 3.31](#).

## 3.10 Creating a Directory

### Problem

You want to create a new directory using the Terminal.

## Solution

The `mkdir` command creates a new directory.

## Discussion

To create a directory, use the `mkdir` command. Try out the following example (note that only the commands are shown, not the responses):

```
$ cd ~
$ mkdir my_directory
$ cd my_directory
$ ls
```

You need to have write permissions in the directory within which you are trying to create the new directory.

## See Also

For general information on using the Terminal to navigate the filesystem, see [Recipe 3.4](#).

## 3.11 Deleting a File or Directory

### Problem

You want to delete a file or directory using the Terminal.

### Solution

The `rm` (remove) command will delete a file or directory and its contents. You should use this with extreme caution.

## Discussion

Deleting a single file is simple and safe. The following example will delete the file *my\_file.txt* from the home directory; you can use the `ls` command to make sure it's gone:

```
$ cd ~
$ rm my_file.txt
$ ls
```

You need to have write permissions in the directory within which you are trying to carry out the deletion.

You can also use the `*` wildcard when deleting files. This example deletes all the files that begin with *my\_file.* in the current directory:

```
$ rm my_file.*
```

You could also delete all the files in the directory by typing:

```
$ rm *
```

If you want to recursively delete a directory (that is, not just the directory itself, but all the files and directories that it contains), you can use the `-r` option:

```
$ rm -r mydir
```

### WARNING

When deleting files from a Terminal window, remember that you do not have the safety net of a recycle bin from which deleted files can be retrieved. Also, generally speaking, you won't be given the option to confirm; the files will just immediately be deleted. This can be totally devastating if you combine it with the `sudo` command ([Recipe 3.12](#)).



## See Also

For more information on navigating the file system using the Terminal, see [Recipe 3.4](#).

If you are concerned about accidentally deleting files or folders, you can force the `rm` command to confirm deletions by setting up a command alias ([Recipe 3.36](#)).

## 3.12 Performing Tasks with Superuser Privileges

### Problem

Some commands don't work because you have *insufficient privileges*.

### Solution

You need to issue commands with superuser privileges. The `sudo` (substitute user do) command allows you to perform actions with superuser privileges. Just prefix the command with `sudo`.

#### Be Careful of `sudo`

As was once said in a famous movie franchise, “with great power comes great responsibility.” The `sudo` command allows you to do really dangerous things, like deleting important system files that could render your Raspberry Pi useless without a complete reinstallation of Raspberry Pi OS.

### Discussion

Most tasks that you want to perform on the command line can usually be performed without superuser privileges. The most common exceptions to this are when you're installing new software and editing configuration files.

The `apt` command is the principal way of installing new software into Raspberry Pi OS. You will meet it formally in [Recipe 3.17](#).

Another example requiring superuser privileges is the `reboot` command. If you try to run it as a normal user, you will receive a number of error messages:

```
$ reboot
Failed to set wall message, ignoring: Interactive authentication
required.
Failed to reboot system via logind: Interactive authentication
required.
Failed to open /dev/initctl: Permission denied
Failed to talk to init daemon.
```

If you issue the same command prefixed with `sudo`, the command will work just fine:

```
$ sudo reboot
```

If you have a whole load of commands to run as superuser and don't want to have to prefix each command with `sudo`, you can use the following command:

```
$ sudo sh
#
```

Note how the prompt changes from `$` to `#`. All subsequent commands will be run as superuser. When you want to revert to being a regular user, enter the `exit` command:

```
# exit
$
```

**See Also**

To understand more about file permissions, see [Recipe 3.13](#).

To install software using `apt`, see [Recipe 3.17](#).

## 3.13 Understanding File Permissions

### Problem

You have seen the strange characters that accompany a filename when it is listed. You would like to know what they all mean.

### Solution

To see the permissions and ownership information relating to files and directories, use the `ls` command with the option `-l`.

### Discussion

Run the command `ls -l` (the option letter is a lowercase *L*), and you will see a result like this:

```
$ ls -l
total 16
-rw-r--r-- 1 pi pi    5 Apr 23 15:23 file1.txt
-rw-r--r-- 1 pi pi    5 Apr 23 15:23 file2.txt
-rw-r--r-- 1 pi pi    5 Apr 23 15:23 file3.txt
drwxr-xr-x 2 pi pi 4096 Apr 23 15:23 mydir
```

The first line of response from the `ls` command tells you 16 files are in the directory.

[Figure 3-11](#) shows the different sections of the listing information. The first block contains the permissions. In the second block, the number 1 (labeled “Files”) indicates how many files are involved. This field makes sense only if the listing entry is for a directory; if it is a file, it will mostly just be 1.

The next two entries (both `pi`) are the owner and group of the file. The size entry (the fifth block) indicates the size of the file in bytes. The date

modified will change every time the file is edited or changed, and the final entry is the actual name of the file or directory.

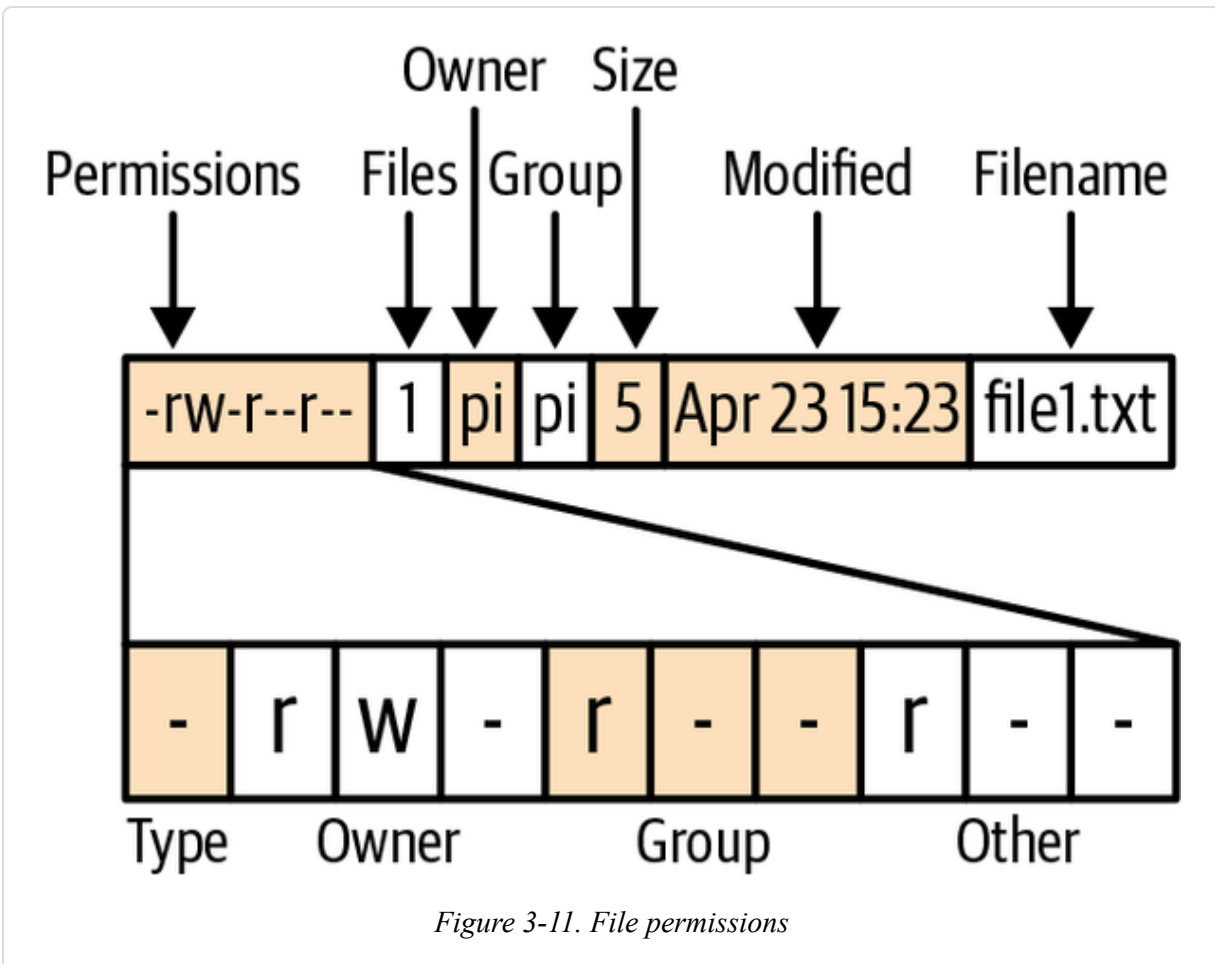


Figure 3-11. File permissions

The permissions block is split into four sections (Type, Owner, Group, and Other). The first section is the type of the file. If this is a directory, it will be the character `d`; if it is a file, the entry will just be a `-`.

The next section comprises three characters that specify the various owner permissions for the file. Each character is a flag that is either on or off. If the owner has read permissions, an `r` will be in the first character position. If the owner has write permissions, a `w` will be in the second slot. The third position, which is `-` in this example, will have an `x` if the file is executable (a program or script) by the owner.

The third section has the same three flags but for any users in the group. Users can be organized into groups. In this case, the file has a user `pi` and a

group ownership of `pi`. If any other users were in the group `pi`, they would have the permissions specified here.

The final section specifies the permissions for users who are neither `pi` nor in the group `pi`.

Because most people will only ever use the Raspberry Pi as the user `pi`, the permissions of most interest are in the first section.

## See Also

To change file permissions, see [Recipe 3.14](#).

# 3.14 Changing File Permissions

## Problem

You need to change the permissions of a file.

## Solution

You can use the command `chmod` to modify file permissions.

## Discussion

Common reasons why you might want to change file permissions include needing to edit a file that is marked as read-only and giving a file execute permissions so that it can run as a program or script.

The `chmod` command allows you to add or remove permissions for a file. There are two syntaxes for doing this: one requires the use of octal (base 8), and the other is text based. We'll use the easier-to-understand text method.

The first parameter to `chmod` is the change to make, and the second is the file or folder to which it should apply. The change parameter takes the form of the permission scope (+, -, = for add, remove, and set, respectively) and then the permission type.

For example, the following command will add execute (*x*) rights for the owner (user) of the file *file2.txt*:

```
$ chmod u+x file2.txt
```

If we now list the directory, we can see that the *x* permission has been added:

```
$ ls -l
total 16
-rw-r--r-- 1 pi pi    5 Apr 23 15:23 file1.txt
-rwxr--r-- 1 pi pi    5 Apr 24 08:08 file2.txt
-rw-r--r-- 1 pi pi    5 Apr 23 15:23 file3.txt
drwxr-xr-x 2 pi pi 4096 Apr 23 15:23 mydir
```

If we wanted to add execute permissions for the group or for other users, we would use *g* and *o*, respectively. The letter *a* adds the permission for everyone.

You will often find examples of setting the file permissions using a number. For example:

```
$ chmod 777 file1.txt
```

Each of the three digits represents 3 bits for the owner, group, and other parts of the file permission. The digits are octal, that is number base 8, so their binary values are as shown in the following table.

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101

Octal	Binary
6	110
7	111

For example, a file permission of `rwxr--r--` would be represented by the number 744.

## See Also

For background on file permissions, see [Recipe 3.13](#).

For a `chmod` octal calculator, see <https://chmod-calculator.com>.

See [Recipe 3.15](#) for changing file ownership.

## 3.15 Changing File Ownership

### Problem

You need to change the ownership of a file.

### Solution

You can use the command `chown` (change owner) to modify the ownership of a file or directory.

### Discussion

As we discovered in [Recipe 3.13](#), any file or directory has both an owner and a group associated with it. Because most users of the Raspberry Pi will just have the single user `pi`, we don't really need to worry about groups.

Occasionally, you will find files on your system that have been installed with a different user than `pi`. If this is the case, you can change the ownership of the file using the `chown` command.

To change the owner of a file, use `chown` followed by the new owner and group, separated by a colon, and then the name of the file.

You will probably find that you need superuser privileges to change ownership, in which case you should prefix the command with `sudo` ([Recipe 3.12](#)). In this example, we change the owner of *file2.txt* from `pi` to `root`:

```
$ sudo chown root:root file2.txt
$ ls -l
total 16
-rw-r--r-- 1 pi pi 5 Apr 23 15:23 file1.txt
-rwxr--r-- 1 root root 5 Apr 24 08:08 file2.txt
-rw-r--r-- 1 pi pi 5 Apr 23 15:23 file3.txt
drwxr-xr-x 2 pi pi 4096 Apr 23 15:23 mydir
```

## See Also

For background on file permissions, see [Recipe 3.13](#).

Also see [Recipe 3.14](#) for changing file permissions.

## 3.16 Making a Screen Capture

### Problem

You want to capture an image of the Raspberry Pi's screen and save it to a file.

### Solution

Use the delightfully named `scrot` (SCREenshOT) screen capture software.

### Discussion

The simplest way to trigger a screen capture is just to enter the command `scrot`. This will immediately take an image of the primary display and



save it in a file named something like `2023-04-25-080116_1024x768_scrot.png` within the current directory.

Sometimes you want a screenshot to show a menu being opened or something that generally disappears when the window loses focus. For such situations, you can specify a delay before the capture takes place by using the `-d` option:

```
$ scrot -d 5
```

The delay is specified in seconds.

If you capture the entire screen, you can crop it later with image editing software, such as GIMP ([Recipe 4.6](#)). However, it is more convenient to just capture a specific area of the screen in the first place, which you can do by using the `-s` option.

To use this option, type the following command and then, with the mouse, drag to define the area of screen that you want to capture:

```
$ scrot -s
```

The filename will include the dimensions in pixels of the image captured.

## See Also

The `scrot` command has a number of other options to control things like using multiple screens and changing the format of the saved file. You can find out more about `scrot` from its manpage by entering the following command:

```
$ man scrot
```

## MANPAGES

Manpages (manual pages) are available for almost all Raspberry Pi OS commands, and you can see them by entering the command name followed by the command itself. However, a command's manpage is not always very accessible, being a thorough reference for the command rather than a simple guide for how to use it. So it's often better just to do an internet search for the command.

## 3.17 Installing Software with apt

### Problem

You want to install software using the command line.

### Solution

The most frequently used tool for installing software from a Terminal session is `apt` (the Advanced Packaging Tool).

The basic format of the command, which you must run as superuser, is as follows:

```
$ sudo apt install <name of software>
```

For example, to install the AbiWord word processing software, you would enter this command:

```
$ sudo apt install abiword
```

### Discussion

The `apt` package manager uses a list of available software. This list is included with the Raspberry Pi operating system distribution but is likely to be out of date. So it is a good idea to always run the following command to update the list before installing new software using `apt`:

```
$ sudo apt update
```

The list and the software packages for installation are all on the internet, so none of this will work unless your Raspberry Pi has an internet connection.

### TIP

If you find that when you update you get an error like `E: Problem with MergeList /var/lib/dpkg/status`, try running these commands, which will remove the offending file and replace it with a new, empty one:

```
$ sudo rm /var/lib/dpkg/status
$ sudo touch /var/lib/dpkg/status
```

The installation process can often take a while because the files must be downloaded and installed. Some installations will also add shortcuts to your desktop or the program groups on your Raspberry Menu.

You can search for software to install using the command `apt search` followed by a search string such as `abiword`. This then displays a list of matching packages that you could install.

## See Also

See [Recipe 3.18](#) for removing programs that you no longer need so that you can free up space.

See also [Recipe 3.21](#) for downloading source code from GitHub.

For installing Python programs with `pip`, see [Recipe 3.19](#).

To install software using a graphical user interface, see [Recipe 4.2](#).

## 3.18 Removing Software Installed with apt

## Problem

Having installed a whole load of programs using `apt`, you now find that you want to remove some of them.

## Solution

The `apt` utility has an option (`remove`) that will remove a package, but it will remove only those packages that have been installed with `apt install`.

For example, if you wanted to remove `AbiWord`, you would use the following command:

```
$ sudo apt remove abiword
```

## Discussion

Removing a package like this doesn't always delete everything because packages often have prerequisite packages that are installed as well. To remove these, you can use the `autoremove` option, as shown here:

```
$ sudo apt autoremove abiword  
$ sudo apt clean
```

The `apt clean` option will do some further tidying up of unused package installation files.

## See Also

See [Recipe 3.17](#) for installing packages using `apt`.

# 3.19 Installing Python Packages with `pip3`

## Problem

You want to use the `pip3` or `pip` (Pip Installs Packages) package manager to install Python libraries.

## Solution

If you have the latest version of Raspberry Pi OS, `pip3` will already be installed, and you can run it from the command line. `pip3` installs Python 3 packages, and on the rare occasion that you might need to install packages for Python 2, just use `pip`.

Here is an example of using `pip3` to install:

```
$ pip3 install pyserial
```

If `pip3` is not installed on your system, you can install it using this command:

```
$ sudo apt install python3-pip
```

Sometimes you will want the software package to be installed for both Python 2 and Python 3, and so you might find yourself running the same commands using both `pip` and `pip3`.

## Discussion

Although many Python libraries can be installed using `apt` (see [Recipe 3.17](#)), some can't, and you must use `pip` instead.

## See Also

To install software using `apt`, see [Recipe 3.17](#).

## 3.20 Fetching Files from the Command Line

## Problem

You want to download a file from the internet without using a web browser.

## Solution

You can use the `wget` command to fetch a file from the internet.

For example, the following command fetches the file *Pifm.tar.gz* from *https://www.icrobotics.co.uk*:

```
$ wget http://www.icrobotics.co.uk/wiki/images/c/c3/Pifm.tar.gz
--2013-06-07 07:35:01--
http://www.icrobotics.co.uk/wiki/images/c/c3/Pifm.tar.gz
Resolving www.icrobotics.co.uk (www.icrobotics.co.uk)...
155.198.3.147
Connecting to www.icrobotics.co.uk
(www.icrobotics.co.uk)|155.198.3.147|
:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5521400 (5.3M) [application/x-gzip]
Saving to: `Pifm.tar.gz'

100%[=====>]
5,521,400    601K/s

2018-06-07 07:35:11 (601 KB/s) - `Pifm.tar.gz' saved
[5521400/5521400]
```

If your URL contains any special characters, it's a good idea to enclose them in double quotes.

## Discussion

You'll find that some instructions for installing software rely on using `wget` to fetch files. It's often more convenient to do this from the command line rather than use a browser, find the file, download it, and then copy it to the place you need it.

The `wget` command takes the URL to download as its argument and downloads it into the current directory. It's typically used to download an

archive file of some type but will also download any web page. So, you could, for example, fetch the Google homepage into a file called *index.xhtml* using the command:

```
$ wget google.com
```

## See Also

For information on installing with `apt`, see [Recipe 3.17](#).

## 3.21 Fetching Source Code with Git

### Problem

Sometimes Python libraries and another software are hosted on the GitHub website or other online Git repository. You need to be able to fetch them onto your Raspberry Pi.

### Solution

To use code in Git repositories, you need to use the `git clone` command to make your own copy of the files.

For example, the following command will download all of the source code examples from this book into a new folder:

```
$ git clone  
https://github.com/simonmonk/raspberrypi_cookbook_ed4.git
```

Along with the URL for the code to clone, there is a web page that you can visit with a browser. If you go to the [GitHub web page for this book](#), you will find a web page similar to the one shown in [Figure 3-12](#).

Clicking the Code button allows you to copy the repository's URL and then paste it after the command `git` in your Terminal session.

## Discussion

There is a difference between Git and GitHub. *Git* is software used to manage code, and *GitHub* is one of many websites hosting code, which was pushed there using Git. In fact, you can actually host your own Git repository on a Raspberry Pi if you want to. However, there are benefits to using a Git-based website such as GitHub or GitLab:

- Your code is stored in the cloud, so if your disk (or SD card) breaks, you won't lose the code.
- The code is publicly visible, so other people can look at it and use it, and if they find things wrong with it they might even offer up fixes for you.
- You can include documentation about your project for all to see in the README file.

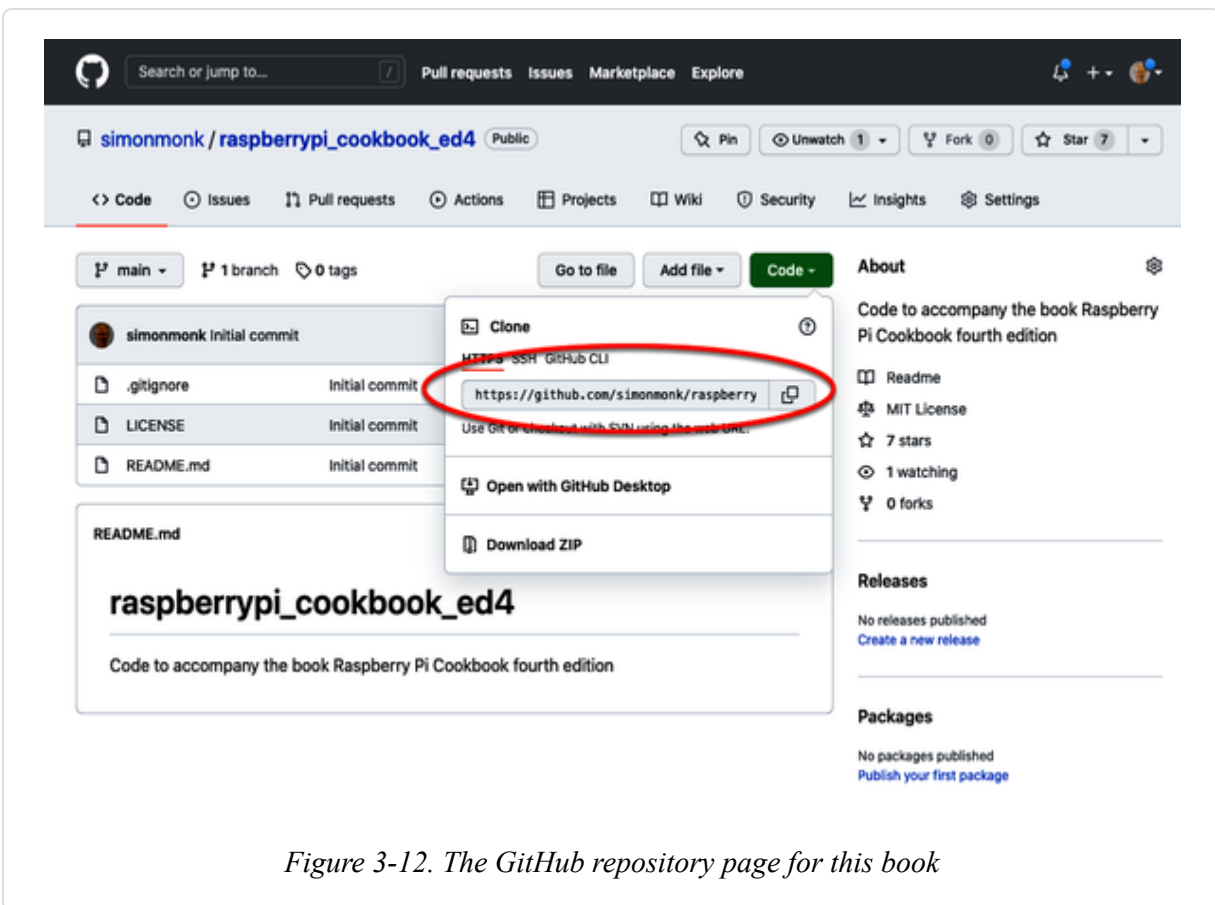


Figure 3-12. The GitHub repository page for this book



If you are working on a Raspberry Pi project that you think others might be interested in, I would recommend using GitHub or GitLab to host your code. There is a little bit to learn, but it's worth the effort.

Another advantage to using Git (whether locally or with a service like GitHub) is that every time you do a chunk of work on a project, you will push that work up to the master copy of your code. This doesn't replace the code that was already there, but is instead stored as a new edition. You can at any time recover earlier versions of the code should you make a mistake.

### WARNING

The terms “master” and “slave” used in the 1-wire and SPI interfaces were written into standards and code many years ago. The origins of these words are clearly problematic. There are moves to replace this terminology with something more modern (and more technically accurate). But for now, unfortunately, we will continue seeing these names until code libraries and the standards they reflect are brought up-to-date.

I host all of the code from my books and other projects on GitHub. These are the steps I take when I make a new repository:

1. Go to your home page on GitHub (you'll need to create an account) and click the + button and select the New Repository option.
2. Give the repository a name and short description.
3. Check the option “Initialize this repository with a README.”
4. Select a license—I select MIT for no better reason than my great respect for that august center of learning, reasoning that if it has a license for sharing my work, it's probably a good one.
5. Click “Create repository.”
6. Back on your computer (Raspberry Pi or other), open a Terminal ([Recipe 3.3](#)).
7. Run the command `git` followed by the URL of the repository. This will create a folder for the project. Any files that you write in this folder will eventually be saved to GitHub when you type the following commands:

```
$ git add .  
$ git commit -m "message about what you changed or added"  
$ git push
```

The first of these commands adds all of the changed or new files to the list of files to be committed. The `commit` command gives you an option to explain what's new in the changes being committed. Finally, the `push` command pushes the changes up to GitHub. At this point, you are prompted for your GitHub username and password. The password will actually be a token that you generate from your GitHub account.

You will find GitHub to be a rich source of Python and other code for use with the Raspberry Pi. This is especially true when it comes to software interfaces to different types of hardware, such as displays and sensors.

## See Also

Learn more about [Git](#) and the Git hosting services [GitHub](#) and [GitLab](#).

For information on downloading this book's program code and other files relating to this book, see [Recipe 3.22](#).

## 3.22 Fetching This Book's Accompanying Code

### Problem

You want to download all the source code and other files relating to this book.

### Solution

You can either *clone* the files from GitHub as described in [Recipe 3.21](#) or, as we will describe here, get the downloads as a single ZIP archive file from GitHub.

A good starting point for getting the book’s downloads is to go to **the book’s web page** using the browser on your Raspberry Pi. Here, in addition to a link to the book’s code hosted on GitHub, you will find errata and other information about the book.

So whether you begin at the website or directly at **the GitHub page**, when you click the Code button, you will see the Download ZIP option (**Figure 3-13**).

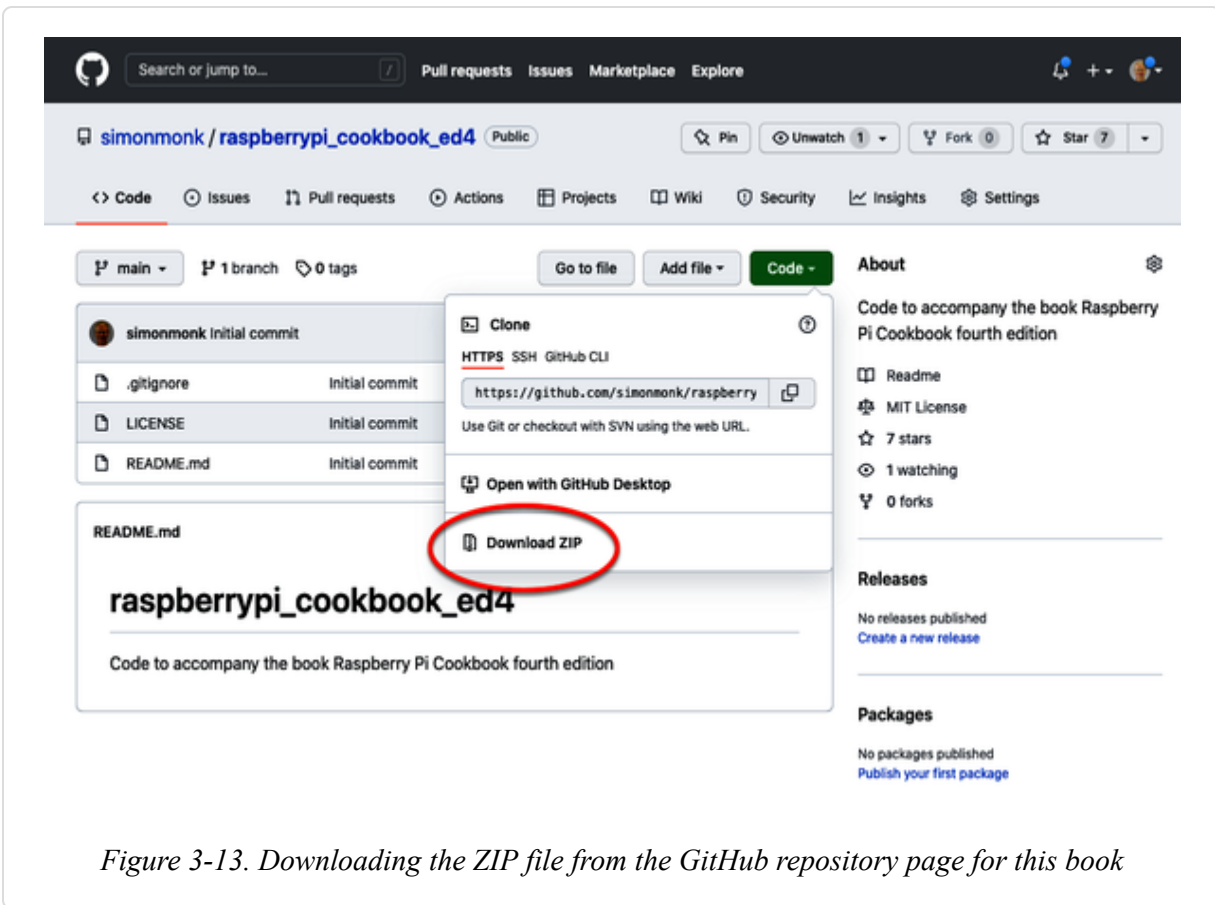


Figure 3-13. Downloading the ZIP file from the GitHub repository page for this book

Click the Download ZIP option. Chromium saves this to your Downloads folder. Click the down arrow next to the downloaded ZIP file and then select the option “Show in folder.”

This opens a File Manager window in the Downloads folder. Find the ZIP file that has just been downloaded.

Double-click the ZIP file to open the Xarchiver tool and then click the “Extract files” icon.

In the dialog that appears, change the path to which the extracted folder is to be saved to */home/pi* and then click Extract.

After the files have been extracted, there will be a new folder in your home directory containing all the downloads for the book.

## Discussion

If you now use the File Manager to see what's in your home directory, you will find a folder called *raspberrypi-cookbook-ed4-master*.

## See Also

For more information on using Git and GitHub, see [Recipe 3.21](#).

## 3.23 Running a Program Automatically on Startup

### Problem

You want a program or script to start automatically as your Raspberry Pi boots.

### Solution

Modify your *rc.local* file to run the program you want.

Edit the file */etc/rc.local* using the following command:

```
$ sudo nano /etc/rc.local
```

Add the following line after the first block of comment lines that begin with #:

```
$ /usr/bin/python /home/pi/my_program.py &
```

It is important to include the `&` on the end of the command line so that it is run in the background; otherwise your Raspberry Pi will not finish booting.

## Discussion

This way of autorunning a program needs a very careful edit of *rc.local*, or you can stop your Raspberry Pi from booting.

## See Also

A safer way of autorunning a program is detailed in [Recipe 3.24](#).

# 3.24 Running a Program Automatically as a Service

## Problem

You want to arrange for a script or program to start automatically every time the Raspberry Pi reboots.

## Solution

Debian Linux, on which most Raspberry Pi distributions are based, uses a dependency-based mechanism for automating the running of commands at startup. This is a little tricky to use and involves creating a configuration file for the script or program that you want to run, which will reside in a folder called *init.d*.

## Discussion

The following example shows you how to run a Python script in your home directory. The script could do anything, but in this case, the script runs a simple Python web server, which is described further in [Recipe 7.17](#).

The steps involved are:

1. Create an *init* script.
2. Make the *init* script executable.
3. Tell the system about the new *init* script.

First, create the *init* script. You need to create this in the folder `/etc/init.d/`. The script can be called anything, but in this example, we call it `my_server`.

Create the new file by using nano with the following command:

```
$ sudo nano /etc/init.d/my_server
```

Paste the following code into the editor window and save the file. This is a lot to type, so if you are reading a paper copy of this book, you can copy and paste the code from [this web page](#); just scroll down until you find this chapter and recipe:

```
### BEGIN INIT INFO
# Provides: my_server
# Required-Start: $remote_fs $syslog $network
# Required-Stop: $remote_fs $syslog $network
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Simple Web Server
# Description: Simple Web Server
### END INIT INFO

#!/bin/sh
# /etc/init.d/my_server

export HOME
case "$1" in
  start)
    echo "Starting My Server"
    sudo /usr/bin/python /home/pi/myserver.py 2>&1 &
    ;;
  stop)
    echo "Stopping My Server"
    PID=`ps auxwww | grep myserver.py | head -1 | awk '{print $2}'`
    kill -9 $PID
    ;;
  *)
```

```
    echo "Usage: /etc/init.d/my_server {start|stop}"
    exit 1
;;
esac
exit 0
```

This is quite a lot of work to automate the running of a script, but most of it is boilerplate code that is the same for every service. To run a different script, just work your way through the script, changing the descriptions and the name of the Python file that you want to run.

The next step is to make this file executable for the owner, which you do using this command:

```
$ sudo chmod o+x /etc/init.d/my_server
```

Now that the program is set up as a service, you can use the following command to test that everything is OK before you set it to autostart as part of the boot sequence:

```
$ /etc/init.d/my_server start
Starting My Server
Bottle v0.11.4 server starting up (using WSGIRefServer())...
Listening on http://192.168.1.16:80/
Hit Ctrl-C to quit.
```

Finally, if that runs OK, use the following command to make the system aware of the new service that you have defined:

```
$ sudo update-rc.d my_server defaults
```

## See Also

For a simpler approach to making a program run automatically, see [Recipe 3.23](#).

For more information on changing file and folder permissions, see [Recipe 3.13](#).

## 3.25 Running a Program Automatically at Regular Intervals

### Problem

You want to run a script once each day or at regular intervals.

### Solution

Use the Linux `crontab` (chronological table) command.

To do this, the Raspberry Pi needs to know the time and date and therefore needs a network connection.

### Discussion

The command `crontab` allows you to schedule events to take place at regular intervals. This can be daily or hourly, and you can even define complicated patterns so different things happen on different days of the week. This is useful for backup tasks that you might want to run in the middle of the night.

You can edit the scheduled events using the following command:

```
$ crontab -e
```

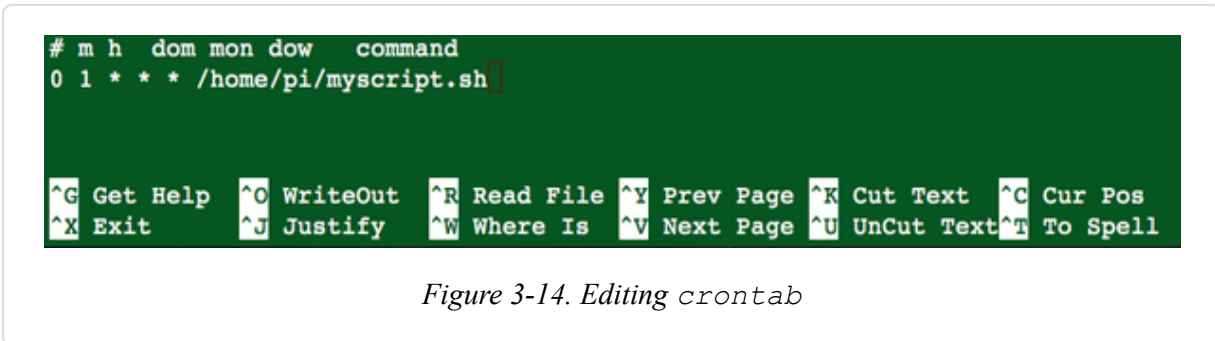
If the script or program that you want to run needs to be run as superuser, prefix all the `crontab` commands with `sudo` ([Recipe 3.12](#)).

The comment line (starting with a `#`) indicates the format of a `crontab` line. The digits are, in order, minute, hour, day of month, month, and day of week and are followed by the command that you want to run.

If there is a `*` in the relevant position, that means *every*; if there is a number instead, the script runs only at that minute/hour/day of the month.



For example, to run *myscript.sh* every day at 1 a.m., you would add the line shown in [Figure 3-14](#).



*Figure 3-14. Editing crontab*

By specifying a range of day numbers, say 1–5 (Monday to Friday), in the day of week column, the script will run only at 1 a.m. on those days, as demonstrated here:

```
0 1 * * 1-5 /home/pi/myscript.sh
```

If your script needs to be run from a particular directory, you can use a semicolon (;) to separate multiple commands, as shown here:

```
0 1 * * * cd /home/pi; python mypythoncode.py
```

## See Also

You can see the full manpage documentation for `crontab` by entering this command:

```
$ man crontab
```

## 3.26 Finding a File

### Problem

You want to find a file that you know is on the system somewhere.

## Solution

Use the Linux `find` command.

## Discussion

Starting with a directory specified in the command, the `find` command will search for a file that you specify and, if it finds the file, display its location.

For example:

```
$ find /home/pi -name gemgem.py
/home/pi/python_games/gemgem.py
```

You can start the search at various points on the tree, even at the root of the entire filesystem (`/`). A search of the entire filesystem will take a lot longer and will also produce error messages. You can redirect these error messages by adding `2>/dev/null` to the end of the line.

To search for the file throughout the entire filesystem, use the following command:

```
$ find / -name gemgem.py 2>/dev/null
/home/pi/python_games/gemgem.py
```

Note that `2>/dev/null` redirects output that would make it difficult to see the file when it was eventually found. You can find out more about redirection in [Recipe 3.31](#).

You can also use wildcards with `find` as follows:

```
$ find /home/pi -name match*
/home/pi/python_games/match4.wav
/home/pi/python_games/match2.wav
/home/pi/python_games/match1.wav
/home/pi/python_games/match3.wav
```

```
/home/pi/python_games/match0.wav  
/home/pi/python_games/match5.wav
```

## See Also

The `find` command has a number of other advanced features for searching. To see the full manpage documentation for `find`, use this command:

```
$ man find
```

## 3.27 Using the Command-Line History

### Problem

You want to be able to repeat commands on the command line without having to type them again.

### Solution

Use the up arrow and down arrow keys to select previous commands from the command history, and use the `history` command with `grep` to find older commands.

### Discussion

You can access the previous command you ran by pressing the up arrow key. Pressing it again will take you to the command before that, and so on. If you overshoot the command you wanted, the down arrow key will take you back in the other direction.

If you want to cancel without running the selected command, use Ctrl-C. Ctrl-C is command line for *stop what you are doing*; in many situations, it will stop a program entirely.

Over time, your command history will grow too large for you to use the arrow keys to find a command that you used ages ago. To find a command from way back, you can use the `history` command:

```
$ history
 1 sudo nano /etc/init.d/my_server
 2 sudo chmod +x /etc/init.d/my_server
 3 /etc/init.d/my_server start
 4 cp /media/4954-5EF7/sales_log/server.py myserver.py
 5 /etc/init.d/my_server start
 6 sudo apt update
 7 sudo apt install bottle
 8 sudo apt install python-bottle
```

This lists all of your command history and is likely to have far too many entries for you to find the command you want. To remedy this, you can use the `|` character to *pipe* (see [Recipe 3.33](#)) the `history` command into the `grep` command, which will display only results matching a search string. So, for example, to find all the `apt` ([Recipe 3.17](#)) commands that you've issued, you can use the line:

```
$ history | grep apt
 6 sudo apt update
 7 sudo apt install bottle
 8 sudo apt install python-bottle
55 history | grep apt
```

Each history item has a number next to it, so if you find the line you were looking for, you can run it using `!` followed by the history number, as shown here:

```
$ !6
sudo apt update
.....
```

## See Also

To find files rather than commands, see [Recipe 3.26](#).

## 3.28 Monitoring Processor Activity

### Problem

The Raspberry Pi can run a bit slow sometimes, so you want to see what's hogging the processor.

### Solution

Use the Task Manager utility, which you'll find on the Raspberry Menu, in the Accessories group (Figure 3-15).

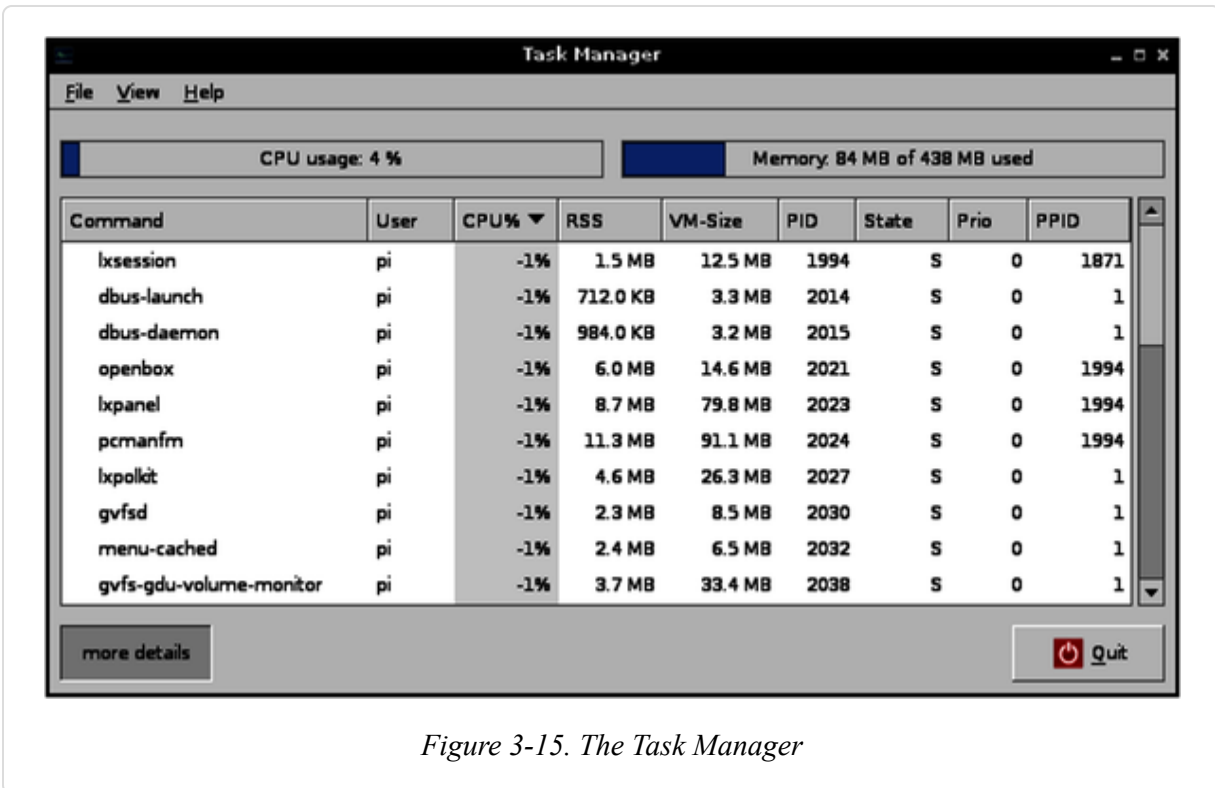


Figure 3-15. The Task Manager

The Task Manager allows you to see at a glance how much CPU and memory are being used. You can also right-click a process and select the option to kill it from the pop-up menu that appears.

The bar graphs toward the top of the window display the total CPU usage and memory usage. The processes are listed below that, and you can see the CPU share each is taking.

## Discussion

If you prefer to do this type of thing from the command line, use the Linux `top` command to display very similar data about processor and memory usage and which processes are using the most resources (Figure 3-16). You can then use the `kill` command to kill a process. You will need to do this as superuser.

In this case, you can see that the `top` process is a Python program that uses 97% of CPU. The first column shows its process ID (2447). To kill this process, enter this command:

```
$ kill 2447
```

It is quite possible to kill some vital operating system process this way, but if you do, powering off your Pi and turning it back on again will restore things to normal.

```

pi@raspberrypi: ~ -- ssh -- 92x26
top - 06:50:41 up 38 min, 1 user, load average: 0.22, 0.12, 0.07
Tasks: 69 total, 2 running, 67 sleeping, 0 stopped, 0 zombie
%Cpu(s): 99.7 us, 0.3 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 448776 total, 151356 used, 297420 free, 17192 buffers
KiB Swap: 102396 total, 0 used, 102396 free, 92684 cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 2447 pi         20   0  9748 5216 2052 R   97.0   1.2   0:11.07 python
 2206 lightdm    20   0 92132 11m 9248 S    1.3   2.6   0:54.98 lightdm-gtk-gre
 2192 root       20   0 44972 11m 3172 S    0.7   2.7   0:25.25 Xorg
 2448 pi         20   0  4656 1352 1028 R    0.7   0.3   0:00.09 top
 2396 pi         20   0  9800 1640 1008 S    0.3   0.4   0:00.32 sshd
   1 root       20   0  2144  728  620 S    0.0   0.2   0:01.89 init
   2 root       20   0    0    0    0 S    0.0   0.0   0:00.00 kthreadd
   3 root       20   0    0    0    0 S    0.0   0.0   0:00.02 ksoftirqd/0
   5 root       0  -20    0    0    0 S    0.0   0.0   0:00.00 kworker/0:0H
   6 root       20   0    0    0    0 S    0.0   0.0   0:00.33 kworker/u:0
   7 root       0  -20    0    0    0 S    0.0   0.0   0:00.00 kworker/u:0H
   8 root       0  -20    0    0    0 S    0.0   0.0   0:00.00 khelper
   9 root       20   0    0    0    0 S    0.0   0.0   0:00.00 kdevtmpfs
  10 root       0  -20    0    0    0 S    0.0   0.0   0:00.00 netns
  12 root       20   0    0    0    0 S    0.0   0.0   0:00.00 bdi-default
  13 root       0  -20    0    0    0 S    0.0   0.0   0:00.00 kblockd
  14 root       20   0    0    0    0 S    0.0   0.0   0:00.35 khubd
  15 root       0  -20    0    0    0 S    0.0   0.0   0:00.00 rpciod
  16 root       20   0    0    0    0 S    0.0   0.0   0:00.00 khungtaskd

```

Figure 3-16. Using the `top` command to see resource usage

Sometimes you might have a process running that is not immediately visible when you use `top`. If this is the case, you can search all the processes running by using the `ps` command and piping (Recipe 3.33) the results to the `grep` command (Recipe 3.27), which will search the results and highlight items of interest.

For example, to find the process ID for our CPU-hogging Python process, we could run the following command:

```

$ ps -ef | grep "python"
pi         2447  2397  99 07:01 pts/0        00:00:02 python speed.py
pi         2456  2397   0 07:01 pts/0        00:00:00 grep  --color=auto
python

```

In this case, the process ID for the Python program `speed.py` is 2447. The second entry in the list is the process for the `ps` command itself.

## THE KILLALL COMMAND

A variation on the `kill` command is the `killall` command. Use this with caution because it kills all processes that match its argument. So, for example, the following command will kill all Python programs running on the Raspberry Pi:

```
$ sudo killall python
```

If you want even more information, try the `htop` command as an alternative to `top`.

### See Also

See also the manpages for `top`, `ps`, `grep`, `kill`, and `killall`. You can view these by typing the `man` command followed by the name of the command for which you want information, as shown here:

```
$ man top
```

## 3.29 Working with File Archives

### Problem

You have downloaded a compressed file and want to uncompress it.

### Solution

Depending on the file type, you will need to use the `tar` command or the `gunzip` command.

### Discussion

If the file that you want to uncompress just has the extension `.gz`, you can unzip it using the command:



```
$ gunzip myfile.gz
```

You also often find files (called *tarballs*) that contain a directory that has been archived with the Linux `tar` utility and then compressed with `gzip` into a file with a name like *myfile.tar.gz*.

You can extract the original files and folders out of a tarball by using the `tar` command:

```
$ tar -xzf myfile.tar.gz
```

If the file is a ZIP archive, you can use the File Manager and Xarchiver tools, as shown in [Recipe 3.22](#).

## See Also

You can find out more about `tar` from its manpage, which you can access with the command `man tar`.

## 3.30 Listing Connected USB Devices

### Problem

You've plugged in a USB device and want to make sure Linux recognizes it.

### Solution

Use the `lsusb` (like `ls` but for USB devices) command. This lists all of the devices attached to the USB ports on your Raspberry Pi:

```
$ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
```

```
Bus 001 Device 004: ID 15d9:0a41 Trust International B.V. MI-  
2540D  
[Optical mouse]
```

## Discussion

This command informs you as to whether a device is connected, but it will not guarantee that the device is working correctly. There might be drivers to install or configuration changes to make for the hardware.

## 3.31 Redirecting Output from the Command Line to a File

### Problem

You want to quickly create a file with some text or record a directory listing into a file.

### Solution

Use the `>` command to redirect output that would otherwise appear in your Terminal after you run the command.

For example, to copy a directory listing into a file called *myfiles.txt*, do the following:

```
$ ls > myfiles.txt  
$ more myfiles.txt  
Desktop  
indiecity  
master.zip  
mcpi
```

## Discussion

You can use the `>` command on any Linux command that produces output, even if you are running, say, a Python program.

You can also use the opposite (<) command to redirect user input, although this is not nearly as useful as >.

## See Also

To use the `cat` command to join together a number of files, see [Recipe 3.32](#).

## 3.32 Concatenating Files

### Problem

You have a number of text files, and you want to join them into one big file.

### Solution

Use the `cat` command to *concatenate* a number of files into one output file.

For example:

```
$ cat file1.txt file2.txt file3.txt > full_file.txt
```

### Discussion

Joining files is the real purpose of the `cat` command. You can supply as many filenames as you like, and they will all be written to the file that you specify. If you do not redirect the output, it will just appear in your Terminal window. If they are big files, this process might take some time!

## See Also

See also [Recipe 3.8](#), in which `cat` is used to display the contents of a file.

## 3.33 Using Pipes

### Problem

You want to use the output of one Linux command as the input to another command.

### Solution

Use the `pipe` command, which is the *bar* symbol (`|`) on your keyboard, to pipe the output of one command to another.

For example:

```
$ ls -l *.py | grep Jun
-rw-r--r-- 1 pi pi 226 Jun  7 06:49 speed.py
```

This example will find all the files with the extension `py` that also have `Jun` in their directory listing, indicating that they were last modified in June.

### Discussion

At first sight, this looks rather like output redirection using `>` ([Recipe 3.31](#)). The difference is that `>` will not work if the target is another program. It will work only for redirecting to a file.

You can chain together as many programs as you like, as shown here, although this isn't something you will do often:

```
$ command1 | command2 | command3
```

### See Also

See [Recipe 3.27](#) to search your command history using `pipe` and `grep`, and [Recipe 3.28](#) for an example of using `grep` to find a process.

## 3.34 Hiding Output to the Terminal

### Problem

You want to run a command, but you don't want the output filling up your screen.

### Solution

Redirect the output to */dev/null* using `>`.

For example:

```
$ ls > /dev/null
```

The `dev` directory contains operating system devices, including things like serial ports. Within this directory, a special device (the null device) is defined that simply discards everything sent to it.

### Discussion

This example illustrates the syntax but is otherwise pretty useless. A more common use is when you're running a program and the developer has left a lot of trace messages in its code, which you don't really want to see. The following example hides superfluous output from the `find` command (see [Recipe 3.36](#)):

```
$ find / -name gemgem.py 2>/dev/null  
/home/pi/python_games/gemgem.py
```

### See Also

For more information about redirecting standard output, see [Recipe 3.31](#).

## 3.35 Running Programs in the Background

## Problem

You want to run a program while also working on some other task.

## Solution

Run the program or command in the background using the `&` command.

For example:

```
$ python speed.py &
[1] 2528
$ ls
```

Rather than wait until the program has finished running, the command line displays the process ID (the second number) and immediately allows you to continue with whatever other commands you want to run. You can then use this process ID to kill the background process ([Recipe 3.28](#)).

To bring the background process back to the foreground, use the `fg` command:

```
$ fg
python speed.py
```

This reports the command or program that is running and then waits for it to finish.

## Discussion

Output from the background process will still appear in the Terminal.

An alternative to putting processes in the background is to just open more than one Terminal window.

## See Also

For information on managing processes, see [Recipe 3.28](#).

## 3.36 Creating Command Aliases

### Problem

You want to create aliases (shortcuts) to commands that you use frequently.

### Solution

Edit the file `~/.bashrc` using nano ([Recipe 3.7](#)), and then move to the end of the file and add as many lines as you want, like this:

```
alias L='ls -a'
```

This creates an alias called *L* that, when entered, will be interpreted as the command `ls -a`.

Save and exit the file using Ctrl-X and Y, and then type the following command to update the Terminal with the new alias:

```
$ source .bashrc
```

### Discussion

Many Linux users set up an alias for `rm`, like the following, so that it confirms deletions:

```
$ alias rm='rm -i'
```

This is not a bad idea, as long as you do not forget when you use someone else's system who doesn't have this alias set up!

### See Also

For more information about `rm`, see [Recipe 3.11](#).

## 3.37 Setting the Date and Time

### Problem

You want to manually set the date and time on your Raspberry Pi because it does not have an internet connection.

### Solution

Use the Linux `date` command.

The date and time format is `MMDDhhmmYYYY`, in which `MM` is the month number; `DD` is the day of the month; `hh` and `mm` are the hours and minutes, respectively; and `YYYY` is the year.

For example:

```
$ sudo date 010203042019
Wed  2 Jan 03:04:00 GMT 2019
```

### Discussion

If the Raspberry Pi is connected to the internet, as it boots up it will automatically set its own time using an internet time server.

You can also use `date` to display the local time by entering `date` on its own:

```
$ date
Fri 19 Jul 10:59:08 BST 2019
```

## 3.38 Finding Out How Much Room You Have on the SD Card

### Problem



You want to know how much free space there is on the SD card.

## Solution

Use the Linux `df` (disk filesystem) command:

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          3.6G  1.7G  1.9G  48% /
/dev/root       3.6G  1.7G  1.9G  48% /
devtmpfs        180M    0  180M   0% /dev
tmpfs           38M   236K   38M   1% /run
tmpfs           5.0M    0   5.0M   0% /run/lock
tmpfs           75M    0   75M   0% /run/shm
/dev/mmcblk0p1  56M   19M   38M  34% /boot
```

The `-h` option shows the sizes using the KB, MB, and GB symbols (shortened to K, M, and G) rather than the number of bytes.

## Discussion

Looking at the first line of the results, you can see that 3.6 GB of storage is on the SD card, of which 1.7 GB is used.

When you run out of disk space, you are likely to get unexpected bad behavior, such as error messages saying that a file could not be written.

## See Also

You can find the manpage for `df` with the command `man df`.

## 3.39 Finding Out What Operating System Version You Are Running

### Problem

You want to know exactly what version of Raspberry Pi OS you are running.

## Solution

Enter the following command into a Terminal or Secure Shell (SSH) session:

```
$ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 11 (bullseye)"
NAME="Raspbian GNU/Linux"
VERSION_ID="11"
VERSION="11 (bullseye)"
VERSION_CODENAME=bullseye
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
```

## Discussion

As you can see from the result in the previous example, the first line tells us all we need to know. In this case, my Raspberry Pi is running Raspbian (Raspberry Pi OS) version 11, which also goes under the nickname *bullseye*.

It can be useful to know what version of Raspberry Pi OS you are running if you are having problems with a certain piece of software. Often the first question that you will be asked by support is *What version of Raspberry Pi OS are you running?*

You might need to know what version of the Linux kernel you have on your Raspberry Pi. You can find this using the following command:

```
$ uname -a
Linux raspberrypi 5.15.32-v7l+ #1538 SMP Thu Mar 31 19:39:41 BST
2022
    armv7l GNU/Linux
```

Here you can see that the author's Raspberry Pi uses v5.15 of the kernel.

## **See Also**

To see how much room you have left on your SD card or other boot disk, see [Recipe 3.38](#).

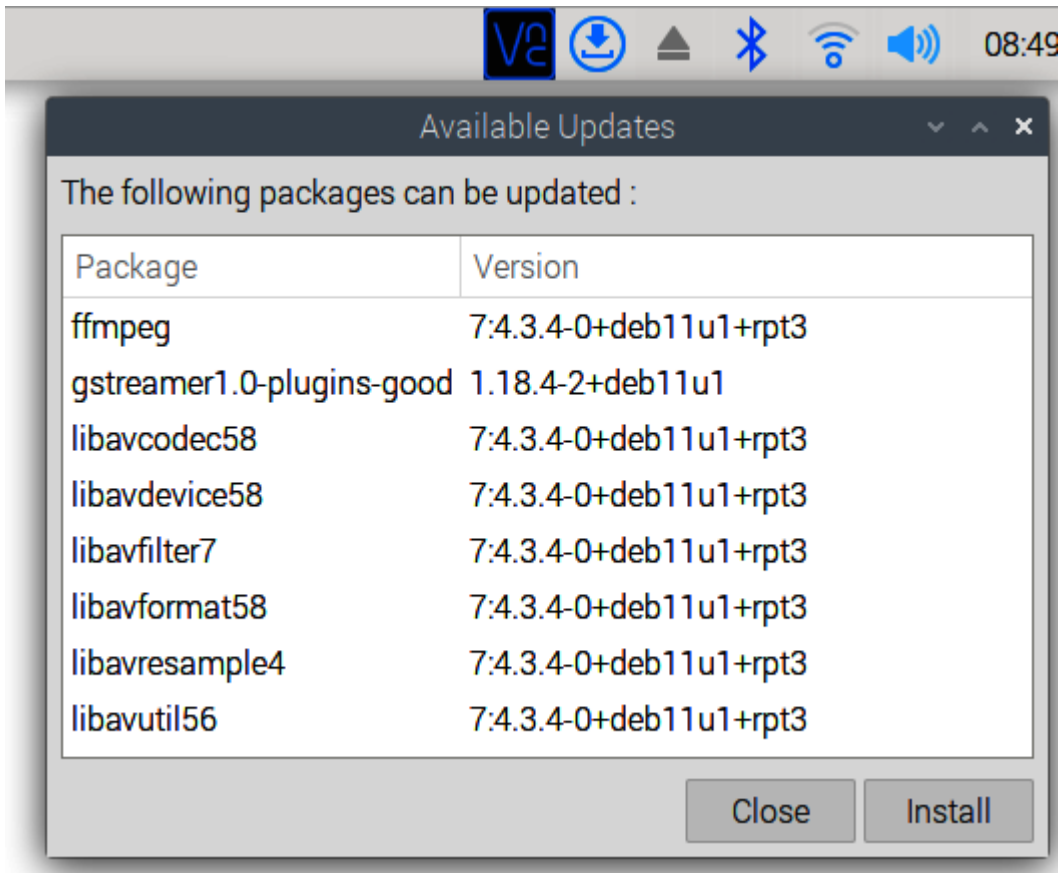
## **3.40 Updating Raspberry Pi OS**

### **Problem**

You want to update your Raspberry Pi to the latest version of Raspberry Pi OS.

### **Solution**

If you have a fairly new version of Raspberry Pi OS (Bullseye onwards), then an icon should be in the top right of the screen that looks like an arrow pointing down toward a tray ([Figure 3-17](#)). When you click on this, you are given the option to see available updates and install them.



*Figure 3-17. Installing updates from the desktop*

If you prefer, you can install from the command line. Open a command line using the Terminal ([Recipe 3.3](#)) and enter the following command to update your system to the latest version:

```
$ sudo apt update  
$ sudo apt full-upgrade
```

This will take some time, especially if there is a lot to upgrade. Most important, if you have any precious files on your system, I recommend copying them onto a USB flash drive ([Recipe 3.2](#)) before upgrading.

## Discussion

The first of these commands doesn't actually update Raspberry Pi OS; it just updates the `apt` package manager to make it aware of the latest versions of the packages that comprise your operating system and related software.

The command `full-upgrade` upgrades the operating system itself. During the process, you will be warned how much disk space will be required, so you should use [Recipe 3.38](#) to check that you have enough room before pressing Y to go ahead with the upgrade.

Keeping your distribution up to date is important for a number of reasons. First, one of the main reasons that the operating system is changed is to fix bugs. So problems during the installation of software often vanish after a system update. Second, if you expose your Raspberry Pi to the internet, new versions of Raspberry Pi OS often patch security vulnerabilities.

## **See Also**

To start again with a completely fresh installation of Raspberry Pi OS, see [Recipe 1.6](#).

# Chapter 4. Using Ready-Made Software

---

## 4.0 Introduction

This chapter contains a number of recipes for using ready-made software on the Raspberry Pi.

Some of the recipes in this chapter are concerned with converting the Raspberry Pi into a single-use appliance, while others use specific pieces of software on a Raspberry Pi.

## 4.1 Making a Media Center

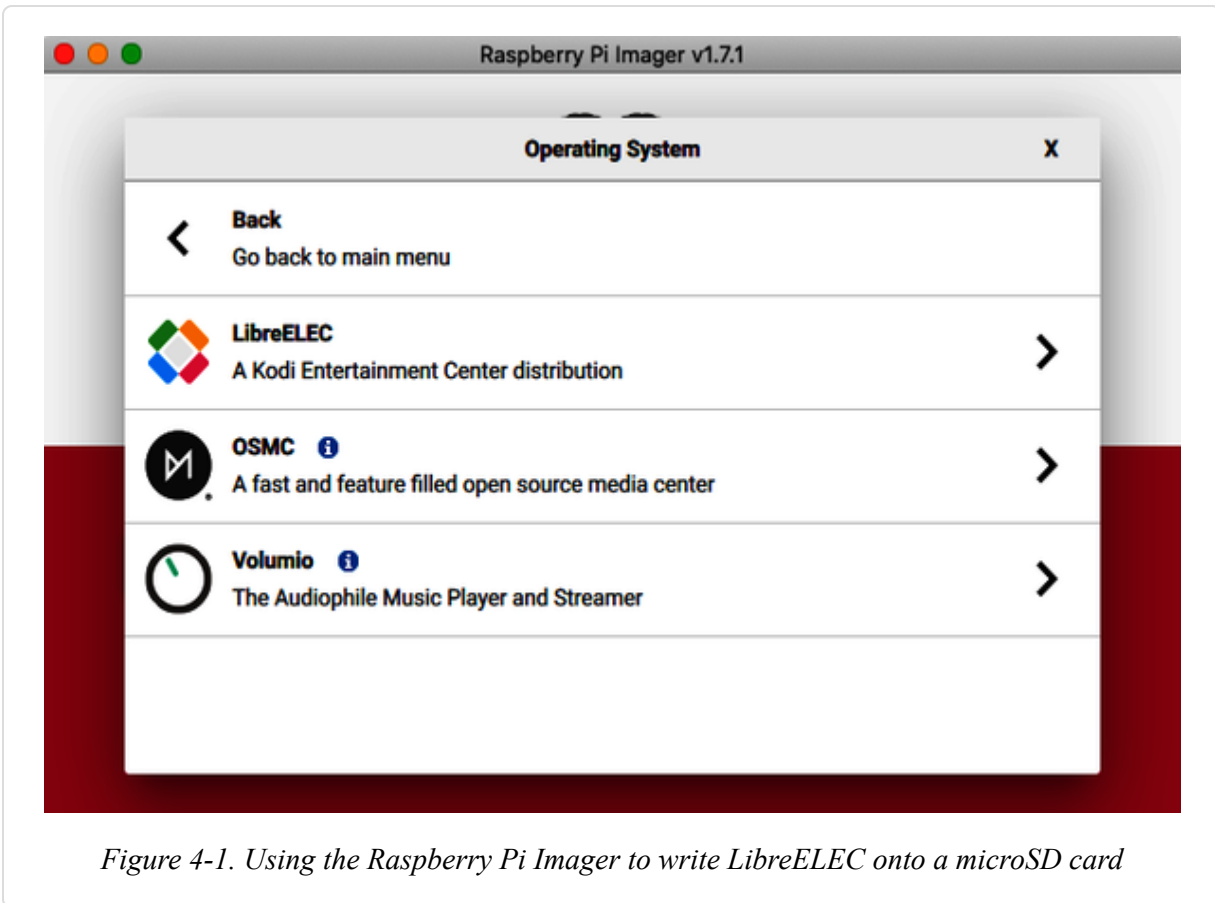
### Problem

You want to convert your Raspberry Pi into a super-duper media center.

### Solution

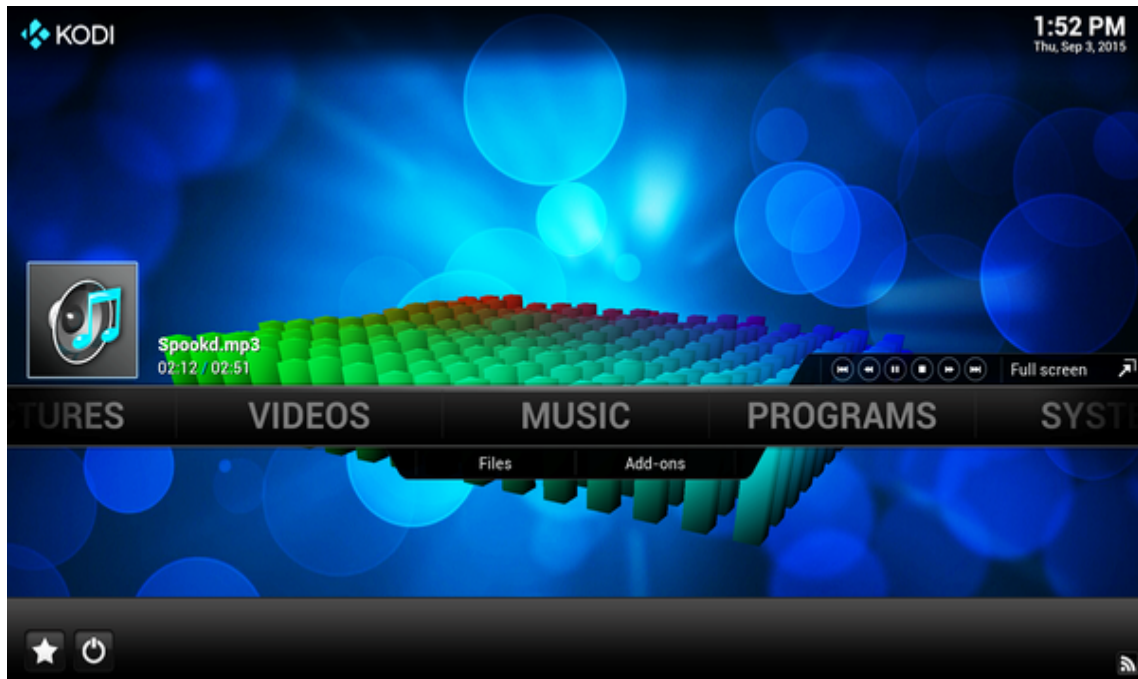
To use your Raspberry Pi as a media center, you should go for the superior performance of the Raspberry Pi 4 B, as video playing is very processor intensive.

You can set up your Raspberry Pi as a media center while using the Raspberry Pi Imager ([Recipe 1.6](#)) to write to a microSD card. Instead of selecting Raspberry Pi OS as the distribution to install, select LibreELEC from the Media Player OS section of the Operating System button ([Figure 4-1](#)).



*Figure 4-1. Using the Raspberry Pi Imager to write LibreELEC onto a microSD card*

LibreELEC is a distribution that optimizes your Raspberry Pi as a media center. It includes the Kodi media center software, which is based on the XBMC open source project that was originally developed to convert Xbox game consoles into media centers. The code has since been ported to many platforms, including the Raspberry Pi (Figure 4-2).



*Figure 4-2. Raspberry Pi as a media center*

Raspberry Pi is perfectly capable of playing full HD video as well as streamed music, MP3 files, and internet radio.

## **Discussion**

**Kodi** is a powerful piece of software with many features and is very intuitive to set up. Perhaps the simplest way to check whether it is working is to put some music and/or video files onto a USB flash drive or external USB hard disk and connect it to the Raspberry Pi. You should be able to play them from Kodi.

Since the Raspberry Pi is likely to be sitting near your TV, you might find that your TV has a USB port that can provide enough current to run the Raspberry Pi. If this is the case, you won't need a separate power supply.

A wireless keyboard and mouse are a good idea because, if you buy them as a pair, they will use a single USB port for the dongle, which avoids the need for wires trailing all over the place. You can also buy mini keyboards with built-in trackpads that are useful in this situation.



A wired network connection is generally higher performance and better than a WiFi connection, but it is not always convenient to have the Pi near an Ethernet socket. If this is the case, you can set up XBMC to use WiFi.

Setting up Kodi is very intuitive, and you can find full instructions on using the software at <http://kodi.wiki>.

## See Also

A popular alternative to LibreELEC, which can also be installed from the Raspberry Pi Imager, is [OSMC](#).

You can add an [infrared \(IR\) remote to Raspberry Pi to control Kodi](#).

## 4.2 Installing Recommended Software

### Problem

You want to install some commonly used software on your Raspberry Pi.

### Solution

Use the Recommended Software tool ([Figure 4-3](#)), which you will find in the Preferences section of the Raspberry Menu.

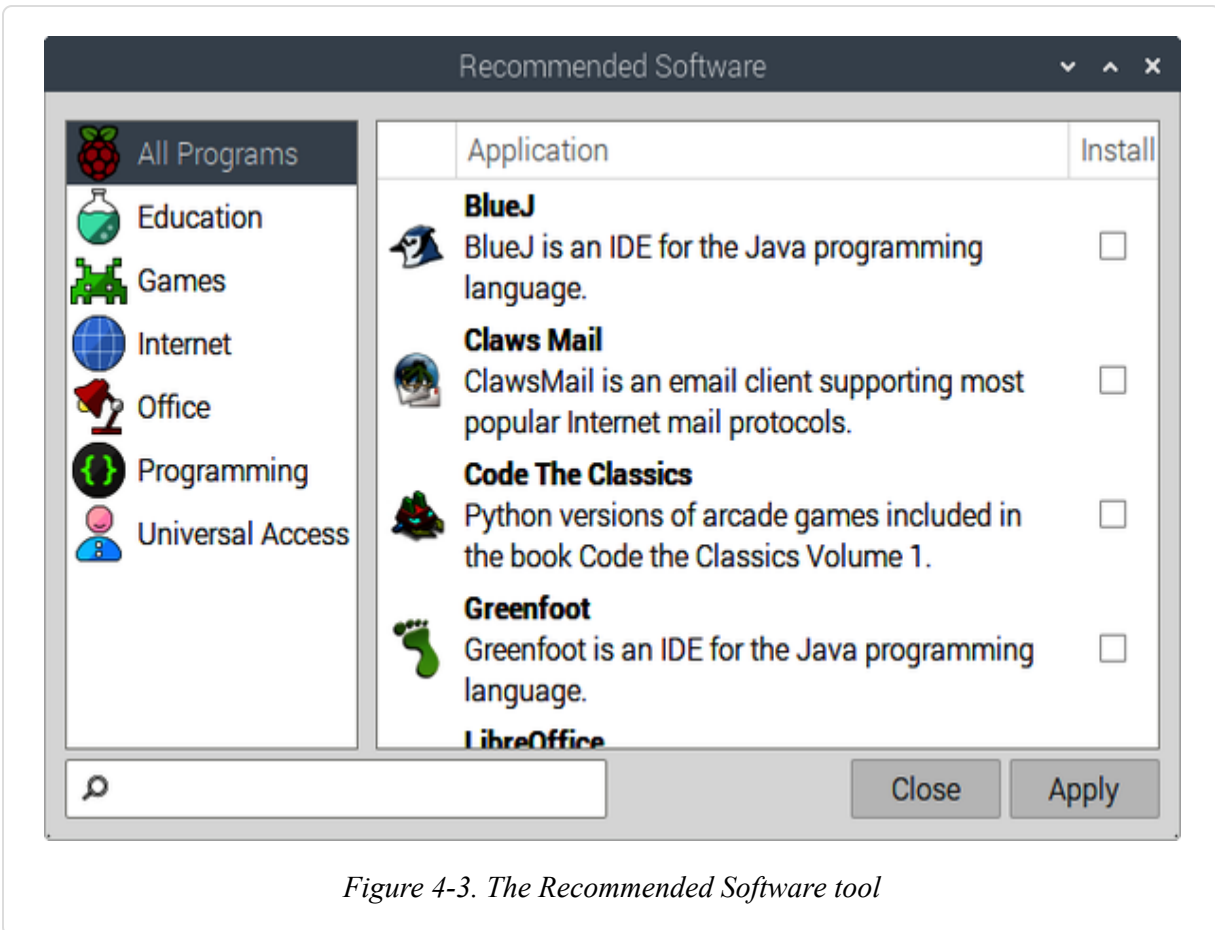


Figure 4-3. The Recommended Software tool

This tool includes many pieces of software that in the past were pre-installed on Raspberry Pi OS. So, it has many of the most common and useful pieces of Raspberry Pi software in it. Use it to browse to the software that you want to install, select the checkbox for the desired programs, and click Apply.

The software will then be downloaded and installed. When the installer has finished, the new software will appear on your Raspberry Menu.

## Discussion

If you can't find the software that you want using the Recommended Software tool, you can widen the net and use the similar Add/Remove Software tool, also in the Preferences section of the Raspberry Menu (Figure 4-4).

This tool has thousands of software packages and programs for you to install; sometimes it is easier to type something in the search area, rather than browse through all the packages.

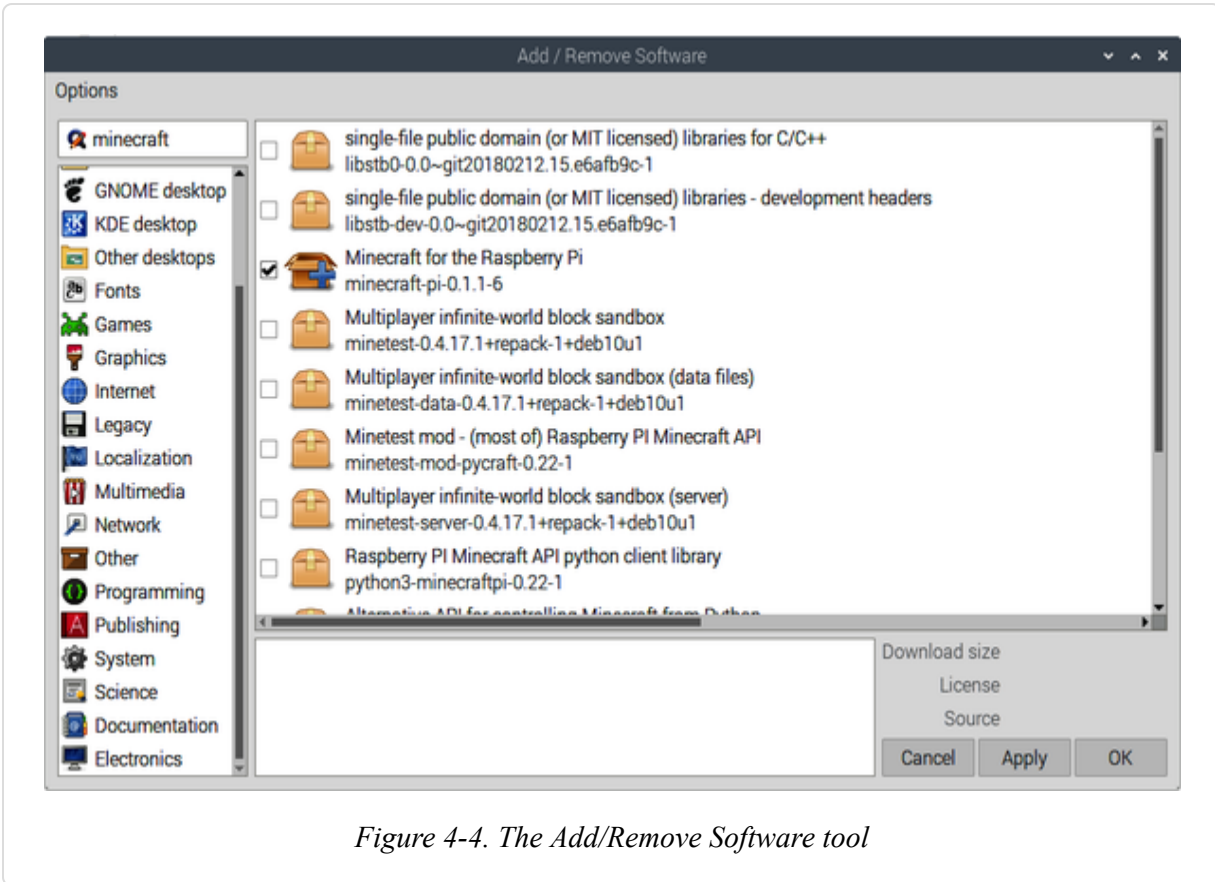


Figure 4-4. The Add/Remove Software tool

## See Also

To install software using the `apt` command line tool, see [Recipe 3.17](#).

## 4.3 Using Office Software

### Problem

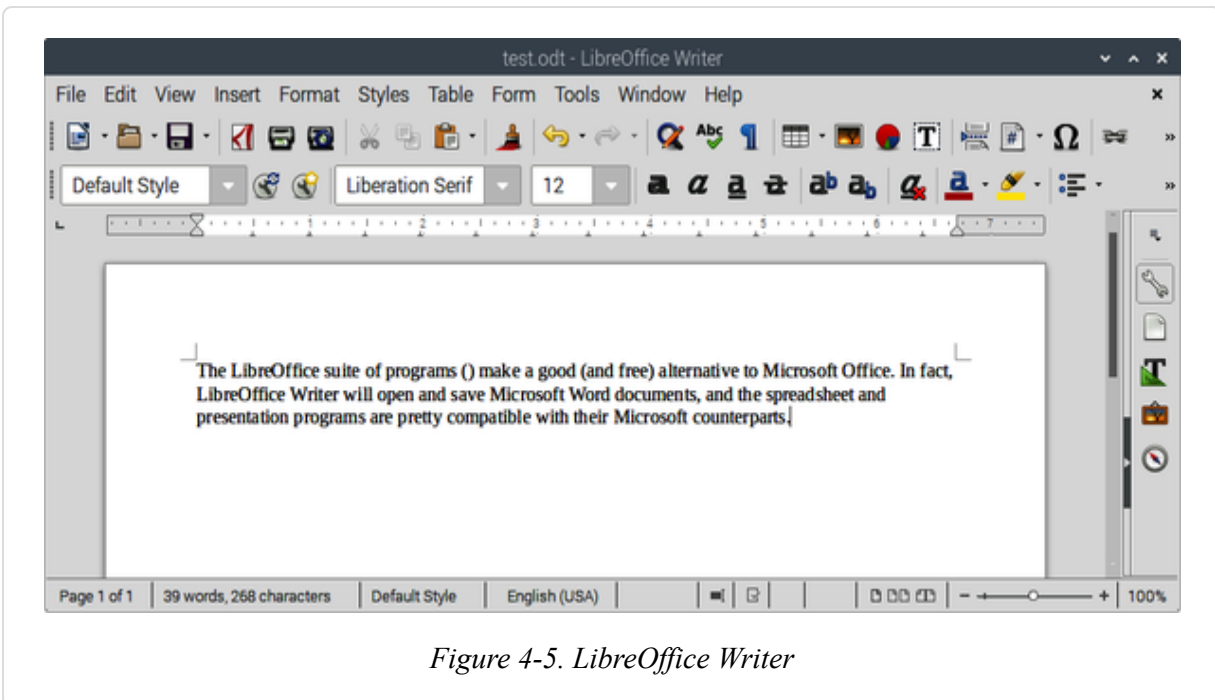
You need to open word processor, presentation, and spreadsheet documents on a Raspberry Pi.

## Solution

Install LibreOffice using the Recommended Software tool ([Recipe 4.2](#)).

## Discussion

The LibreOffice suite of programs ([Figure 4-5](#)) makes a good (and free) alternative to Microsoft Office. It includes a word processor, spreadsheet, presentation, and drawing software. In fact, the LibreOffice Writer word processor will open and save Microsoft Word documents, and the spreadsheet and presentation programs are pretty compatible with their Microsoft counterparts.



A Raspberry Pi 4 or 400 will run office applications much better than an older Raspberry Pi.

These days it's often more convenient to keep your documents in the cloud and edit them in a browser. The most common examples of these services are Microsoft 365 and Google Docs. Both require you to sign up for an account, but the Chromium browser is perfectly capable of using these services ([Figure 4-6](#)).

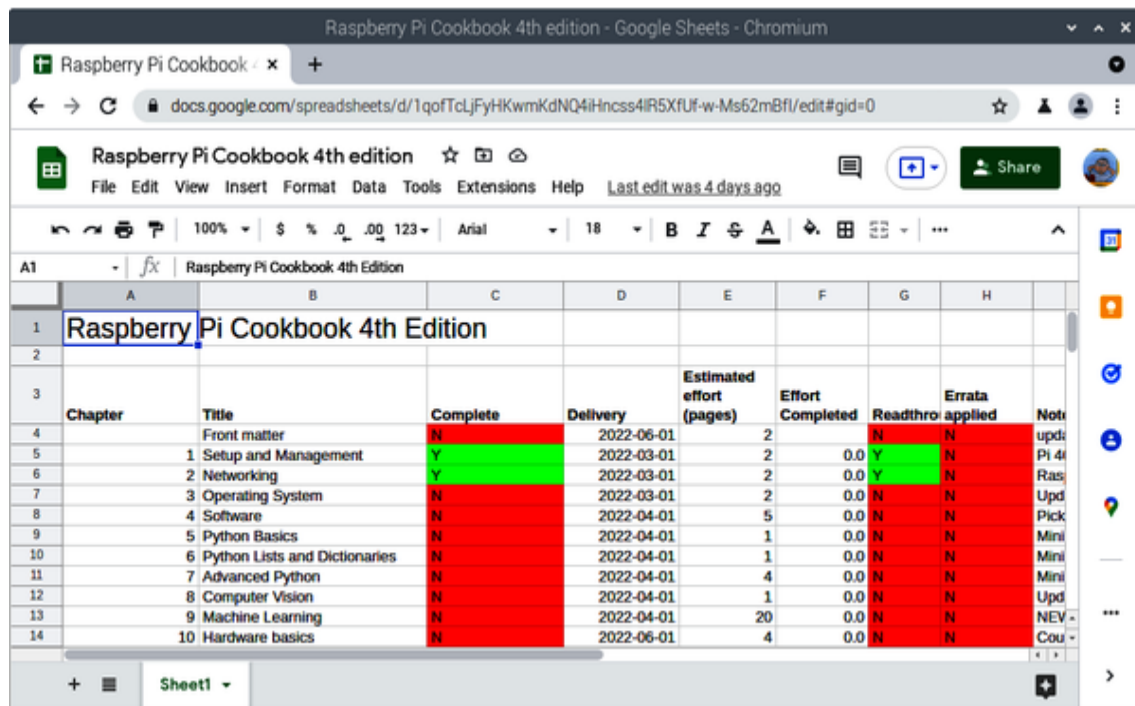


Figure 4-6. Using a Google Docs spreadsheet in the Chromium browser

## See Also

Visit <https://www.libreoffice.org> for more information on the LibreOffice suite of software.

If you just want to edit an unformatted text file, you can use the nano editor (Recipe 3.7) or VisualStudio Code (Recipe 4.10).

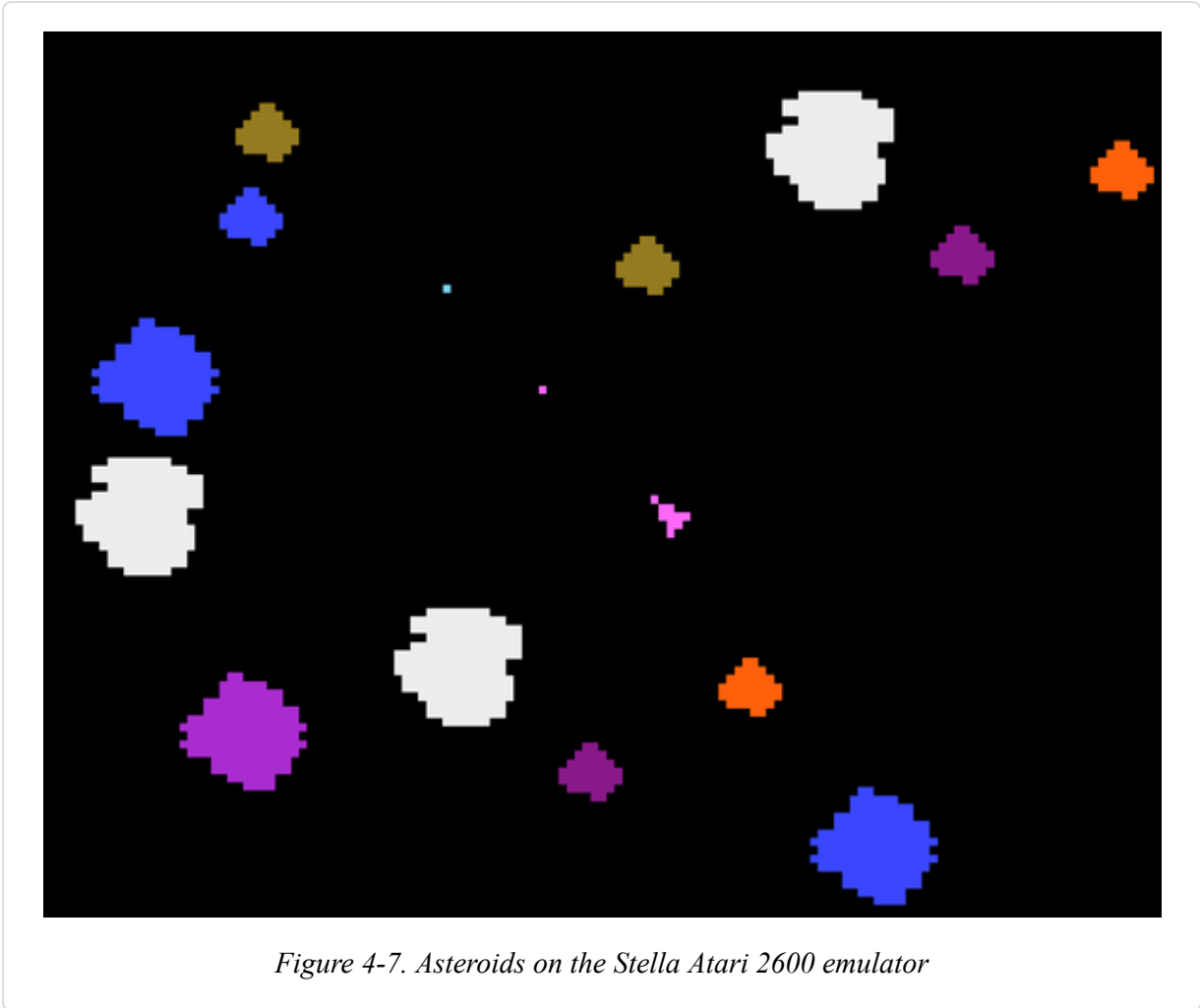
## 4.4 Running a Vintage Game Console Emulator

### Problem

You want to turn your Raspberry Pi into a vintage game console.

### Solution

If you fancy rediscovering your misspent youth and playing Asteroids on an emulator for the Atari 2600 ([Figure 4-7](#)), the RetroPie project will appeal to you.



*Figure 4-7. Asteroids on the Stella Atari 2600 emulator*

Many wonderful projects have been built that create custom consoles and game tables complete with retro game controllers.

Although you can install RetroPie on top of Raspberry Pi OS, the easiest way to use it is to write it onto a microSD card using the Raspberry Pi Imager ([Recipe 1.6](#)).

## WARNING

It is worth noting that even though these games are ancient, they are still owned by someone. The ROM image files that you need to play the games on an emulator, although easy to find on the internet, are not necessarily yours to take. So please stick to the law.

## Discussion

The emulator uses a surprisingly large amount of the Raspberry Pi's meager resources, so you might find that you need to use a Raspberry Pi 4, 3, or 2. In an internet search, you can find many people who have taken this basic setup and added a retro USB controller, like the widely available and quite low-cost controllers, and built the Pi and a monitor into a big arcade-style housing. You can also buy a kit called the Picade from Pimoroni to make a lovely arcade machine ([Figure 4-8](#)).



*Figure 4-8. The Pimoroni Picade kit*

## **See Also**

Full RetroPie documentation is available on [the RetroPie site](#).



## 4.5 Turning Your Raspberry Pi into a Radio Transmitter

### Problem

You want to convert your Raspberry Pi into a low-powered FM transmitter that will send a radio signal to a normal FM radio receiver ([Figure 4-9](#)).

### Solution

Back in the early days of the Raspberry Pi, some clever folks at Imperial College, London, created some C code that allows you to do just this. The download even plays the *Star Wars* theme as a sample. This project will still work if you have an [original Raspberry Pi 1](#).

The project lives on for newer Raspberry Pis as an altogether more advanced project called *rpitx*.

All you need is a short length of wire attached to general-purpose input/output (GPIO) pin 4. A 10 cm female-to-male header lead will work just fine for this. In fact, it should work with a radio sitting right next to your Pi without any kind of antenna—such is the strength of the transmission.

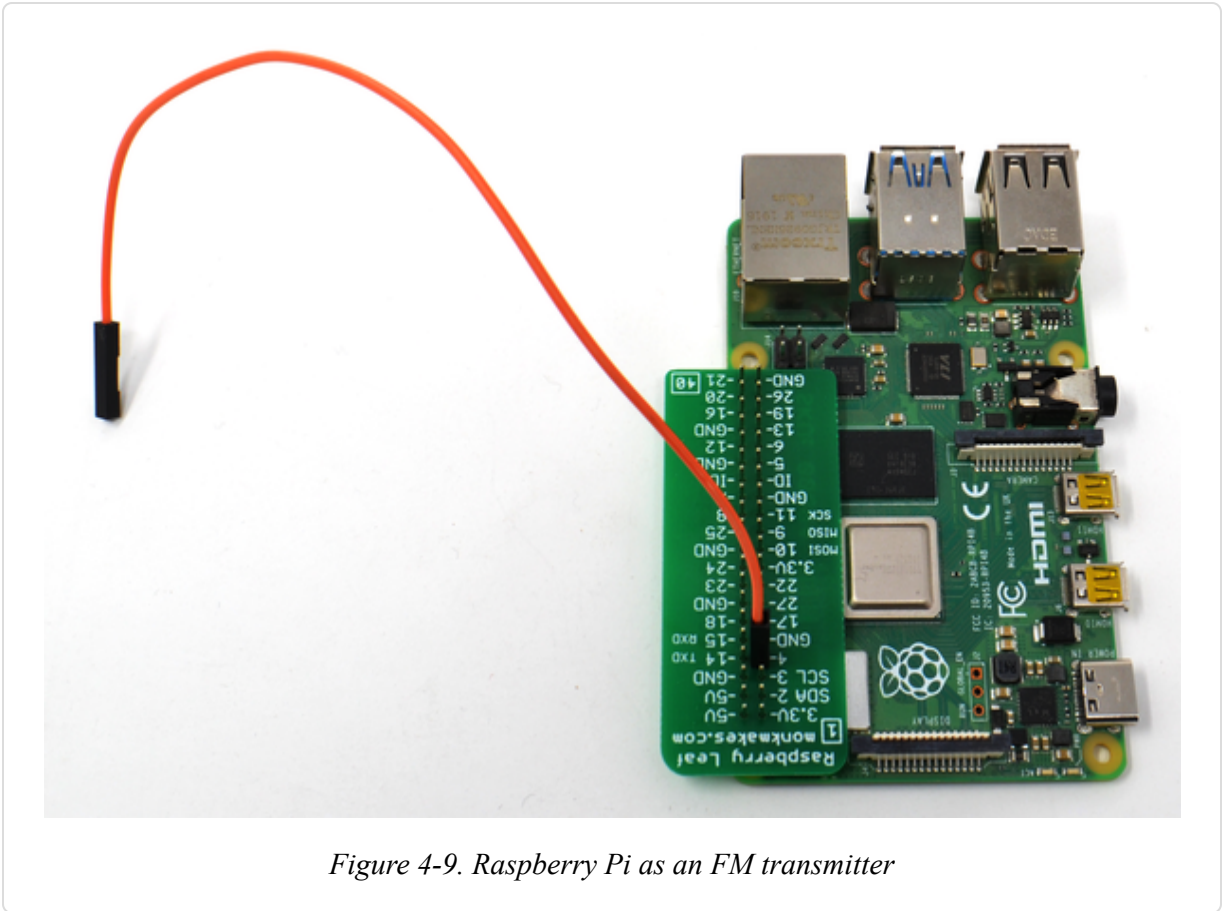


Figure 4-9. Raspberry Pi as an FM transmitter

The first step is to install the rpitx software using the following commands. Note that this installation will change some things about how your Raspberry Pi is configured, including the frequency at which the GPU (graphics process) works. So if this is your main Raspberry Pi, make sure you first back up anything precious. Here's the code you need:

```
$ git clone https://github.com/F5OEO/rpitx
$ cd rpitx
$ ./install.sh
```

You will now need to do something else for a good 15 minutes or so while the software installs. You might see what look like error messages and warnings, but these are normal. At the end of the installation, the installer script will ask:

```
In order to run properly, rpitx need to modify /boot/config.txt.  
Are you  
    sure (y/n)
```

Press Y, and the script will then confirm the changes it has made with the following message:

```
Set GPU to 250Mhz in order to be stable
```

If you need to reverse this change, edit */boot/config.txt* by removing the last line that says `gpu_freq=250`, then reboot.

Next, find yourself an FM radio receiver and tune it to 103.0 MHz. If this frequency is already occupied by some other transmission, pick another frequency and make note of it.

Now run the following command (changing the frequency parameter from 103.0 if you had to change frequency):

```
sudo ./pifmrds -freq "103.0" -audio src/pifmrds/stereo_44100.wav
```

If all is well, you should hear the voice of the developer talking about left and right channels.

## Discussion

You need to know that this project may not be legal in your country. The power output is higher than that of FM transmitters used with MP3 players.

Were you to put a Raspberry Pi in your vehicle, this would be a great way to output sound through the vehicle's audio system.

## See Also

To learn more about the rpitx project, see <https://oreil.ly/TrlO1>.

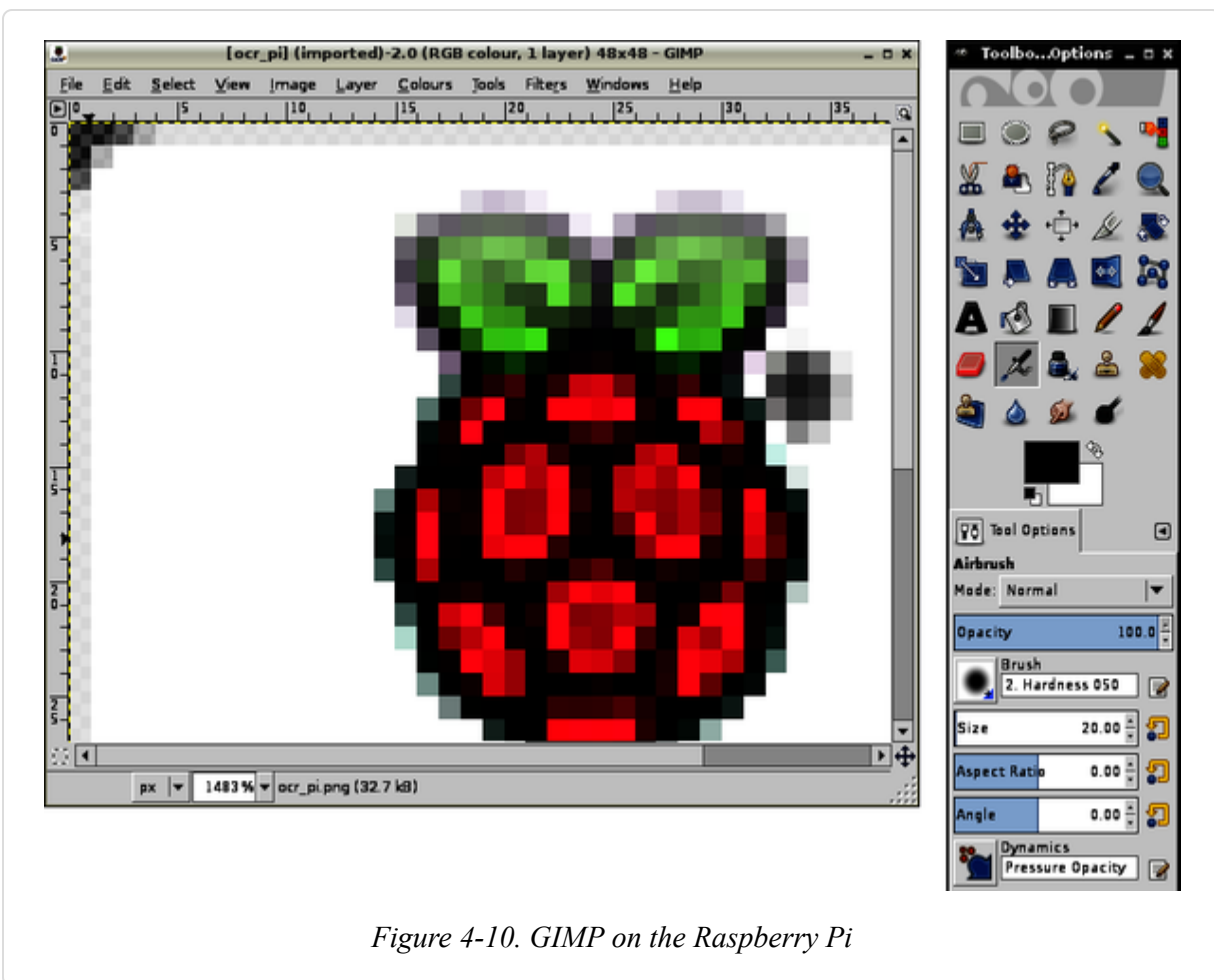
## 4.6 Editing Bitmap Images

### Problem

You want to manipulate a photograph or other image.

### Solution

Install and run the GNU Image Manipulation Program (GIMP; see [Figure 4-10](#)).



*Figure 4-10. GIMP on the Raspberry Pi*

GIMP is available to install from the Add/Remove Software tool (see [Recipe 4.2](#)). When you search for GIMP, a lot of results will come back for various GIMP utilities, so look for the package called “GNU Image Manipulation Program.”

If you prefer to install GIMP from the command line, open a Terminal session and type the following command:

```
$ sudo apt install gimp
```

Once GIMP is installed, you'll find an entry for GNU Image Manipulation Program in your Raspberry Menu under the Graphics heading.

## Discussion

Despite being hungry for memory and processor power, GIMP is usable even on a Raspberry Pi 2 B, but there will be much less waiting around if you use a Raspberry Pi 4 or 400.

## See Also

Find out more from the [GIMP website](#).

GIMP has a lot of features and is a very sophisticated image-editing program, so it does take a little learning. You'll find an online manual for the software at the GIMP website.

For more information on installing with `apt`, see [Recipe 3.17](#).

For editing vector images, see [Recipe 4.7](#).

## 4.7 Editing Vector Images

### Problem

You want to create or edit high-quality vector drawings such as Scalable Vector Graphics (SVG).

### Solution

Inkscape is one of the packages available from the Add/Remove Software tool (see [Recipe 4.2](#)). Open the tool up and search for “Inkscape.”

If you prefer to install Inkscape from the command line, you can do so with the following commands:

```
$ sudo apt update
$ sudo apt install inkscape
```

Once Inkscape is installed, its icon will appear in the Graphics section of your Raspberry Menu.

## Discussion

Inkscape ([Figure 4-11](#)) is the most-used open source vector image editor. Vector drawing packages differ from bitmap image editors like GIMP ([Recipe 4.6](#)) in that the image is made up of shapes, lines, text, etc. that are stored as such, rather than being converted into pixels. This means that you can come back and edit those things (perhaps the position of a line), which is not possible in a bitmap editor.

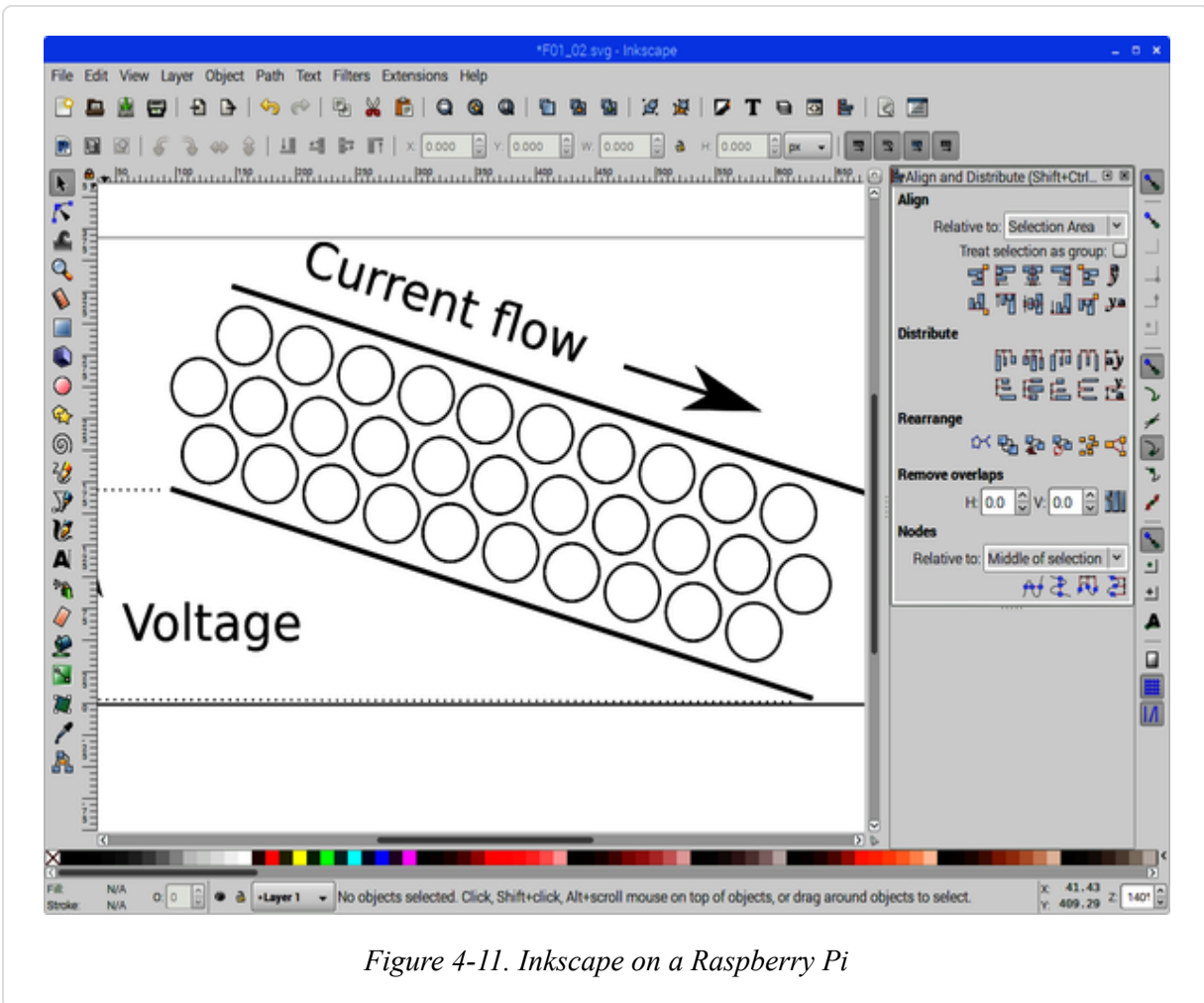


Figure 4-11. Inkscape on a Raspberry Pi

Inkscape is a very powerful piece of software with many features that can take some time to master, so don't be disheartened if it won't do what you want at first. You will probably need to run through a few tutorials.

Inkscape is another program best run with the extra power of a Raspberry Pi 4 or 400.

## See Also

For documentation on Inkscape, visit [Inkscape.org](https://inkscape.org).

For editing bitmap images such as photographs, see [Recipe 4.6](#).

## 4.8 Using Bookshelf

## Problem

You want to read Raspberry Pi books and magazines for free.

## Solution

Use the pre-installed Bookshelf application. You will find this application in the Help section of the Raspberry Menu. Open it and you will see (Figure 4-12) past issues of *The MagPi*, *Wireframe*, and *HackSpace* magazines, as well as books from the Raspberry Pi Press.



Figure 4-12. The Bookshelf application

## Discussion

This is a really great resource. You will find lots of articles in *The MagPi* magazine to get started with various aspects of the Raspberry Pi. It is also a great source of inspiration for projects.

## See Also



Lots of interesting Raspberry Pi resources are available at the [Raspberry Pi Foundation](#).

You can also download back issues of *The MagPi* magazine in PDF or subscribe to the paper edition.

## 4.9 Playing Internet Radio

### Problem

You want to be able to play internet radio on your Raspberry Pi.

### Solution

The VLC media player should be pre-installed with Raspberry Pi OS. You will find it under Sound & Video in the Raspberry Menu. If it's not there, you can install it using the Preferred Software tool ([Recipe 4.2](#)).

If you prefer to install the VLC media player from the command line, you can do so by running the following command:

```
sudo apt install vlc
```

Run the program and then, on the Media menu, select the Open Network Stream option. This opens a dialog box (see [Figure 4-13](#)) in which you can enter the URL of the internet radio station that you want to play. You will need to plug headphones or amplified speakers into the audio socket on the Raspberry Pi.

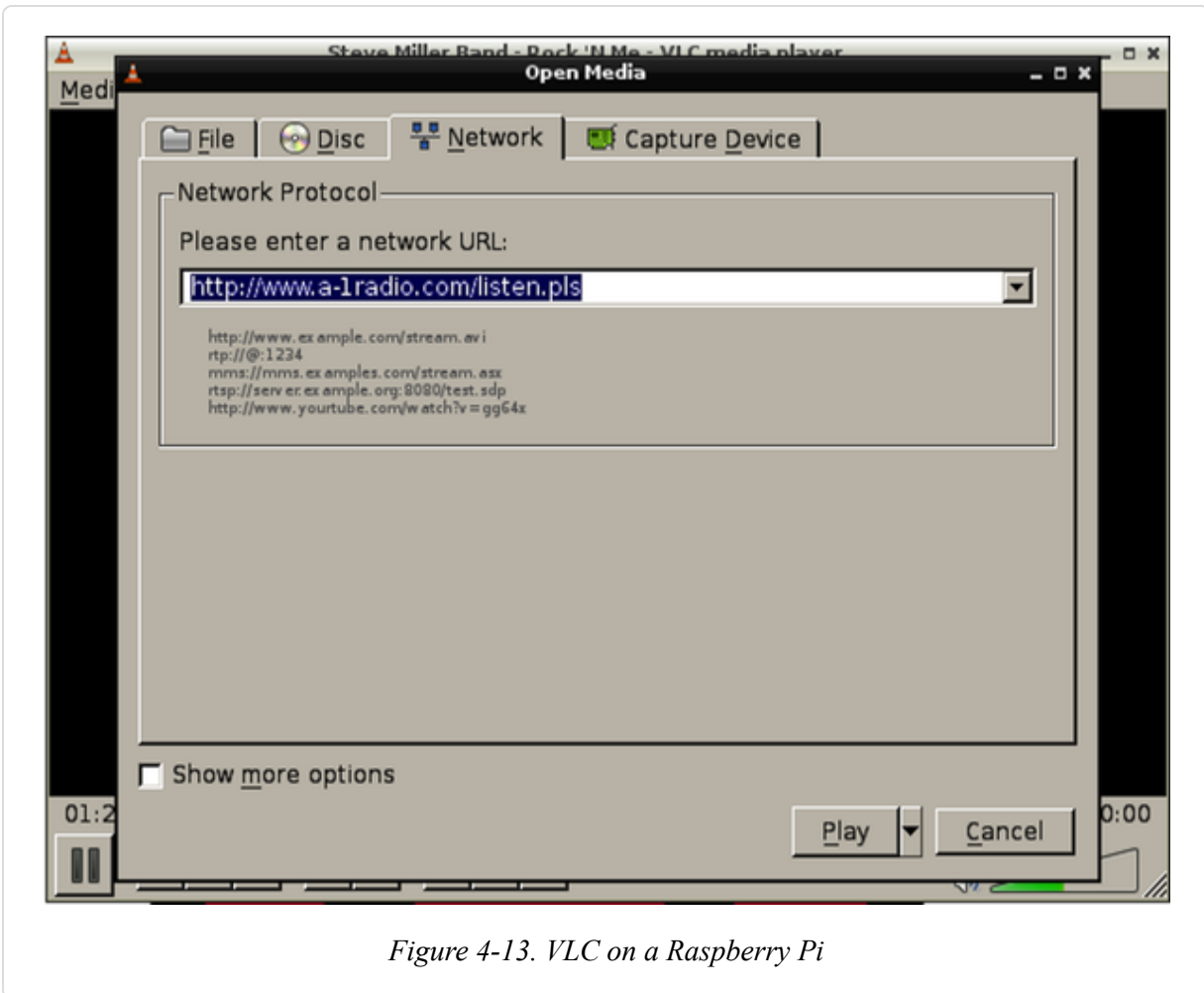


Figure 4-13. VLC on a Raspberry Pi

## Discussion

You also can run VLC from the command line as follows:

```
$ vlc http://www.a-1radio.com/listen.pls -I dummy
```

VLC will probably produce a series of error messages but then play the audio just fine. The `-I dummy` option prevents a VLC window opening.

## See Also

This recipe borrows heavily from this [tutorial](#), in which Jan Holst Jensen takes things a step further and adds radio-style controls to the project.

For UK readers, you can find a list of the [BBC radio stream URLs online](#).

## 4.10 Using Visual Studio Code

### Problem

You want to use a light-weight programming editor.

### Solution

Install Microsoft's Visual Studio Code, or VS Code, as we shall refer to it from now on.

You will find VS Code in the Programming section of the Recommended Software tool ([Recipe 4.2](#)).

If you prefer to install it from the command line, you can use the following command:

```
$ sudo apt install code
```

### Discussion

VS Code is much loved by programmers. It sits in a happy place, halfway between a simple text editor and a fully featured integrated development environment (IDE). It's very easy to use and offers helpful suggestions as you write your code, without you having to learn the ins and outs of a more complex IDE. It offers support for many programming languages and does nice things like color-code your programs to make them easier to read.

If you are a seasoned programmer, you may prefer to edit your Raspberry Pi Python code using Visual Studio Code rather than a beginner's Python IDE like Thonny or Mu ([Recipes 5.3](#) and [5.4](#)). This is particularly true if your project consists of multiple files. [Figure 4-14](#) shows the file explorer area on the left, where you can see all the files in your project. Clicking on one makes it available in the editor area, and open files each have their own tab.

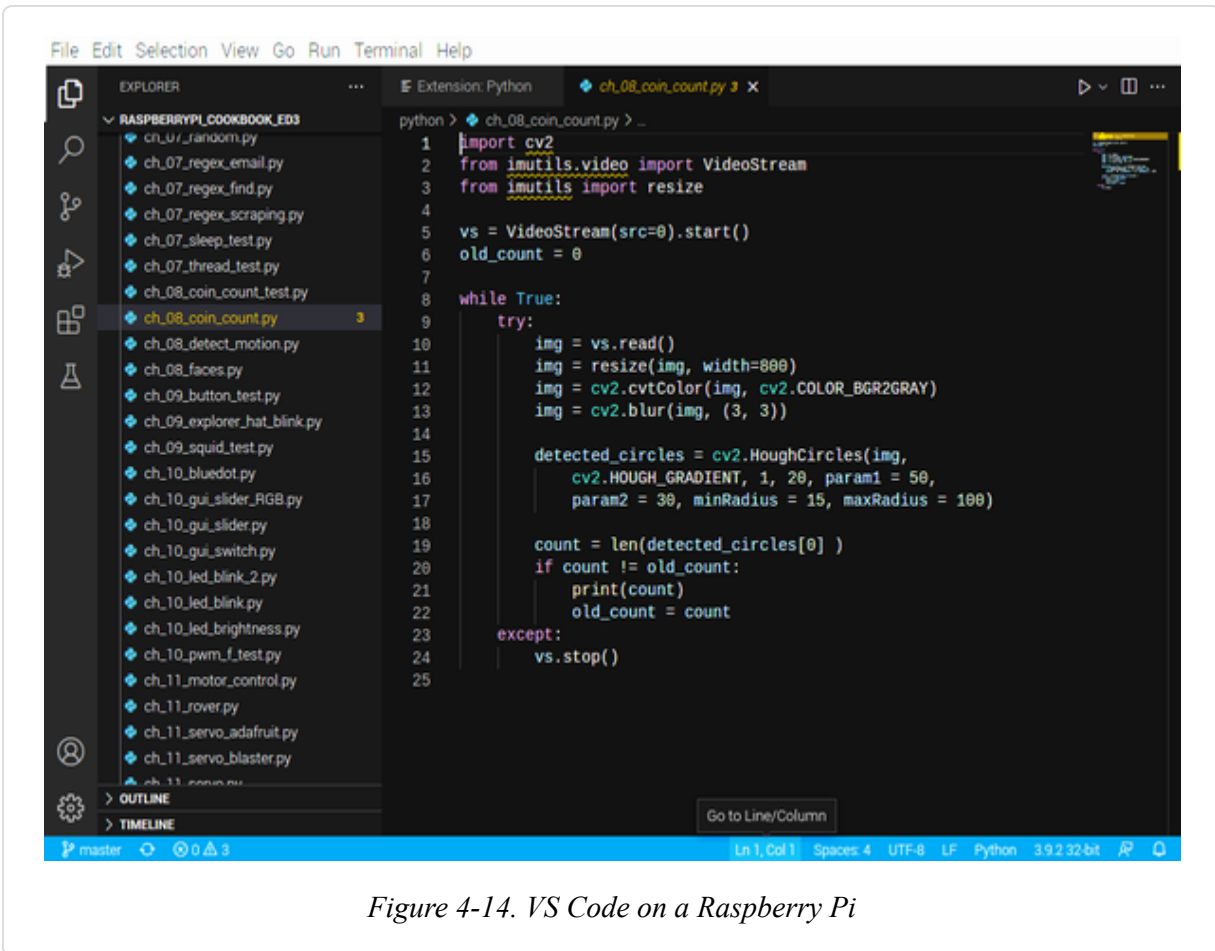


Figure 4-14. VS Code on a Raspberry Pi

As you can see, the editor highlights the language syntax, color-coding it to make it easier to read and spot mistakes.

Visual Studio Code is a useful tool to have around, whether programming in Python (see [Chapter 5](#)) or some other language.

## See Also

The simple text editor nano is an easy way to edit code (see [Recipe 3.7](#)).

Thonny is a Python editor aimed at beginners (see [Recipe 5.3](#)).

## 4.11 Controlling a Laser Cutter

### Problem

You want to control your low-cost K40 laser cutter from your Raspberry Pi.

## Solution

If you have one of the popular and low-cost Chinese laser cutters of the K40 variety ([Figure 4-15](#)), you can use the K40 Whisperer software to control it. Thus, instead of having to use a relatively expensive Windows computer to control your laser cutter, you can use a much lower cost Raspberry Pi.



*Figure 4-15. A K40 laser cutter*

This software is written in Python, but also relies on the Inkscape vector drawing package. So if you have not already done so, install Inkscape by following [Recipe 4.7](#).

Next, download the [source code for the K40 Whisperer software](#). Select the latest download from the K40 Whisperer Source column. Double-click the downloaded ZIP file and extract it into your home directory (*/home/pi*).

Inside the unzipped directory, you will find a file called *README\_Linux.txt*. The instructions here are based on that file. The *README* is for Linux in general rather than being Raspberry Pi-specific.

First, run the following two commands to create a user group for the software. The first adds a new user group specifically for the laser cutter, and the second adds the user `pi` to that group. If you created a different username when setting up your Raspberry Pi, use that in place of `pi`:

```
$ sudo groupadd lasercutter
$ sudo usermod -a -G lasercutter pi
```

Make sure that your laser cutter is connected to your Raspberry Pi with its USB lead and the laser cutter is switched on. You now need to run the following commands to find out the USB vendor and product ID for the laser cutter. These will probably be `1a86` and `5512`, respectively, but it's worth checking in case the laser cutter's manufacturer changed them. To do this, run the `lsusb` command:

```
$ lsusb
Bus 001 Device 004: ID 1a86:5512 QinHeng Electronics CH341 in
EPP/MEM/I2C mode,
  EPP/I2C adapter
Bus 001 Device 005: ID 0424:7800 Microchip Technology, Inc.
(formerly SMSC)
Bus 001 Device 003: ID 0424:2514 Microchip Technology, Inc.
(formerly SMSC)
  USB 2.0 Hub
Bus 001 Device 002: ID 0424:2514 Microchip Technology, Inc.
(formerly SMSC)
  USB 2.0 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

This will list all the USB devices connected to your Raspberry Pi; look for the entry that corresponds to your laser cutter. In this case, it's the only

Chinese name there (QinHeng). But you can confirm this if you need to by unplugging the laser cutter and running the `lsusb` command again and seeing which entry disappears.

On the line for this entry you can see the text `1a86:5512`. The part before the `:` is the vendor ID and `5512` is the product ID. You need to know these because you are going to use them in a configuration file.

Create and edit a new file in nano using the command:

```
$ sudo nano /etc/udev/rules.d/97-ctc-lasercutter.rules
```

You can find information on editing files in [Recipe 3.7](#). Paste the following text into the file. If your vendor and product IDs are different from mine, change the appropriate parts of the text to match your IDs:

```
SUBSYSTEM=="usb", ATTRS{idVendor}=="[1a86]", ATTRS{idProduct}=="[5512]",  
    ENV{DEVTYPE}=="usb_device", MODE="0664", GROUP="lasercutter"
```

Reboot your Raspberry Pi and then continue the installation by running the following command to install some more modules that K40 Whisperer needs:

```
$ sudo apt install libxml2-dev libxslt-dev  
$ sudo apt install libusb-1.0-0  
$ sudo apt install libusb-1.0-0-dev
```

Make sure that you are in the directory where you downloaded the K40 Whisperer and run the following command:

```
$ pip3 install -r requirements.txt
```

You are now ready to run the program using the following command. This will open a window. Click the File menu and then Options to configure the

software (Figure 4-16):

```
$ python3 k40_whisperer.py
```

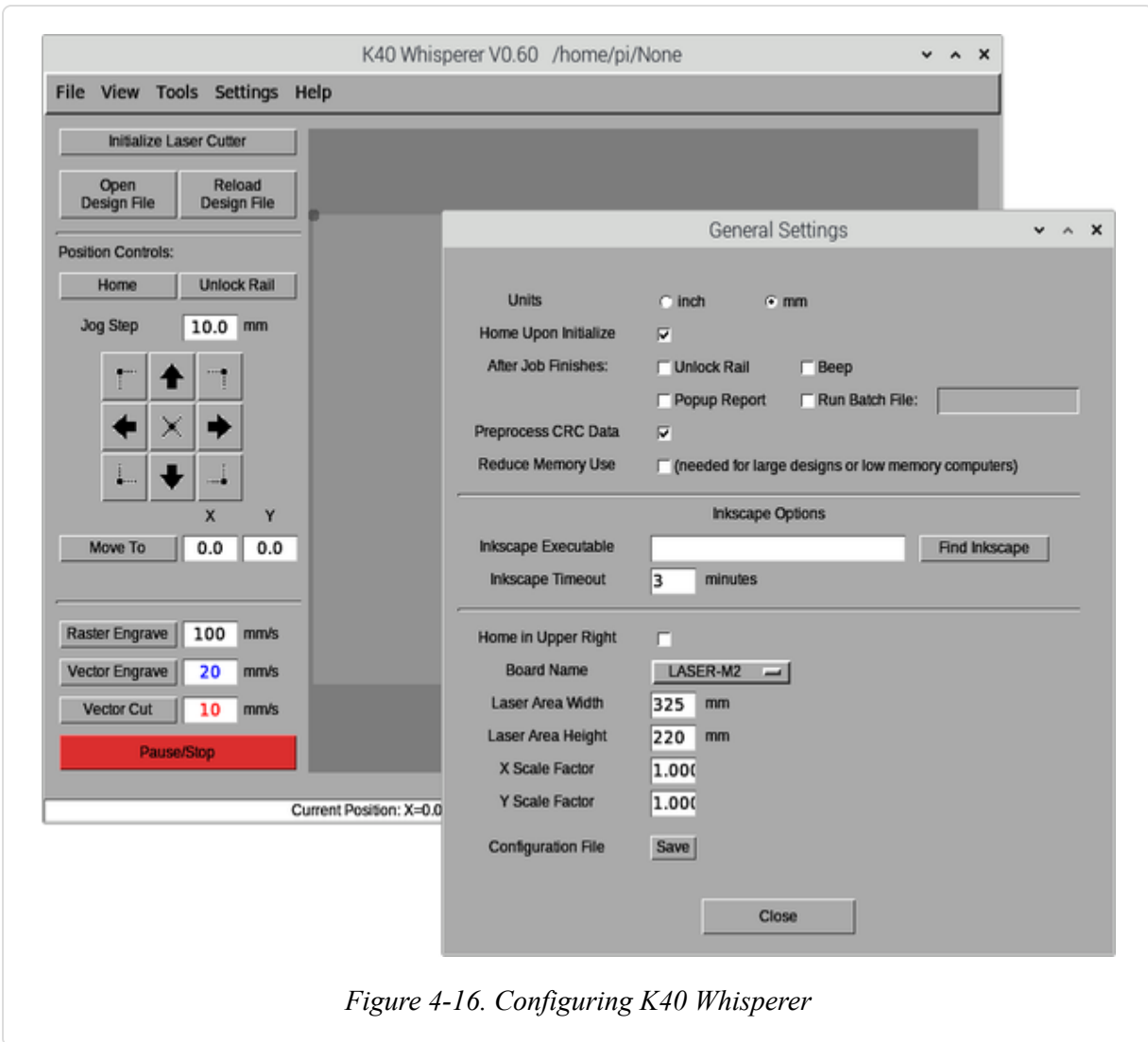


Figure 4-16. Configuring K40 Whisperer

Consult the manual that came with your laser cutter to find the version of the control board used in it. If you can't find this, you might need to try a few different options for the Board Name setting. You do not need to change any of the other settings.

With the Settings window closed, you can load an SVG file to cut (Figure 4-17).



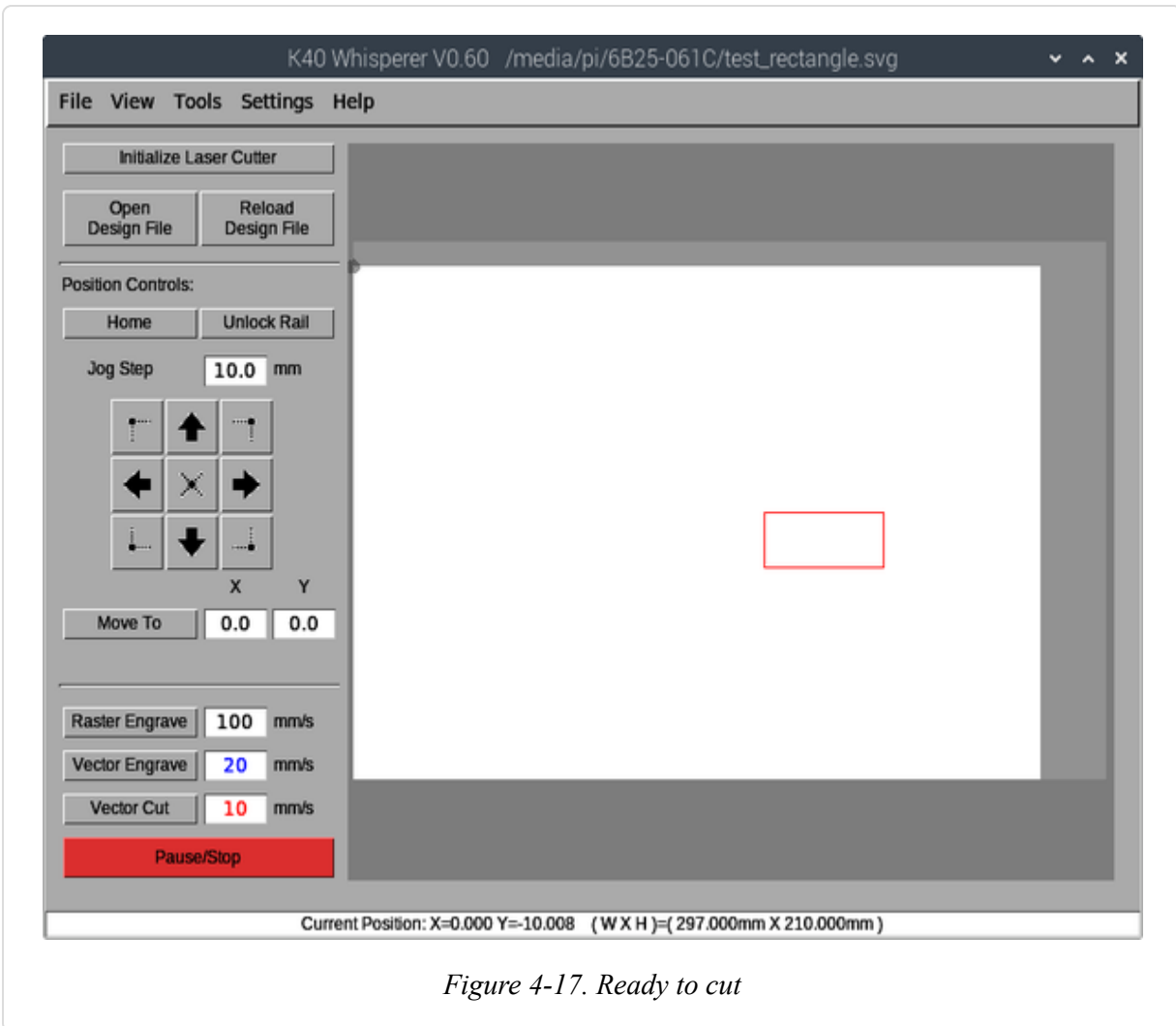


Figure 4-17. Ready to cut

## Discussion

Laser cutters often come with their own proprietary software to control the cutter that is generally run on a Microsoft Windows computer. It may even have a hardware USB dongle designed to tie that piece of software to your computer.

The K40 Whisperer software cuts and engraves directly from SVG drawings. You just draw outlines in different colors with a line width of 0.1mm. Red lines mean cut, blue lines mean vector engrave (lightly burn an outline), and black lines mean raster engrave (scan back and forth to burn out an area of the surface).

## See Also

For full information on K40 Whisperer, see <https://oreil.ly/5RJth>.

To create the SVG drawings for your laser cutting and etching, you can use Inkscape ([Recipe 4.7](#)).

A Raspberry Pi also makes a great controller for a 3D printer using [OctoPrint](#).

# Chapter 5. Python Basics

---

## 5.0 Introduction

Although many languages can be used to program the Raspberry Pi, Python is the most popular. In fact, the “Pi” in Raspberry Pi is inspired by the word “python.”

In this chapter, you’ll find a host of recipes to help you start programming with Raspberry Pi.

## 5.1 Deciding Between Python 2 and Python 3

### Problem

You need to use Python but are unsure which version to use.

### Solution

Use Python 3 until you face a problem that is best solved by reverting to version 2.

To install Python 2 on Raspberry Pi OS, run the following commands:

```
$ sudo apt update
$ sudo apt install python2
```

You can then run Python 2 using the command `python2`.

### Discussion

Although Python’s most recent version, Python 3, has been around for years, you’ll find that a lot of people still use Python 2. Python 3 (the default for Raspberry Pi OS) is run by using either of the commands

`python` or `python3`. The examples in this book are written for Python 3 unless otherwise stated. Most will run on both Python 2 and Python 3 without modification.

This reluctance on the part of the Python community to ditch the older version is largely because Python 3 introduced some changes that broke compatibility with version 2. As a result, some of the huge body of third-party libraries developed for Python 2 won't work under Python 3.

My strategy is to write in Python 3 whenever possible, reverting Python 2 only when I need to because of compatibility problems.

## See Also

For a good summary of the Python 2 versus Python 3 debate, see the [Python wiki](#).

## 5.2 Choosing a Python Editor

### Problem

Many options are available when it comes to Python editors, and you want to know which one you should start with.

### Solution

Most people will start with either Thonny ([Recipe 5.3](#)) or Mu ([Recipe 5.4](#)). Both are good editors and both are easy to use.

Thonny is pre-installed with Raspberry Pi OS, whereas Mu has to be installed. You will find lots of resources for both editors, and Thonny is recommended over Mu in most material from the Raspberry Pi Foundation.

### Discussion

It doesn't matter too much which editor you start with. You might like to try out both Thonny and Mu and see which you like best.

As you get more advanced in your programming experience, you should try VS Code ([Recipe 4.10](#)) as a more professional tool.

## See Also

VS Code is described in [Recipe 4.10](#) and Thonny in [Recipe 5.3](#).

## 5.3 Editing Python Programs with Thonny

### Problem

You want to edit your Python programs using Thonny.

### Solution

Thonny should be pre-installed with Raspberry Pi OS. But if it's not in the Programming section of your Raspberry Menu, you can install it using the Recommended Software tool ([Recipe 4.2](#)).

Start by opening Thonny from the Programming section of your Raspberry Menu and enter the following lines of text, as also shown in [Figure 5-1](#):

```
for i in range(1, 10):  
    print(i)
```

When you start the second line of the program, the print statement should indent automatically.

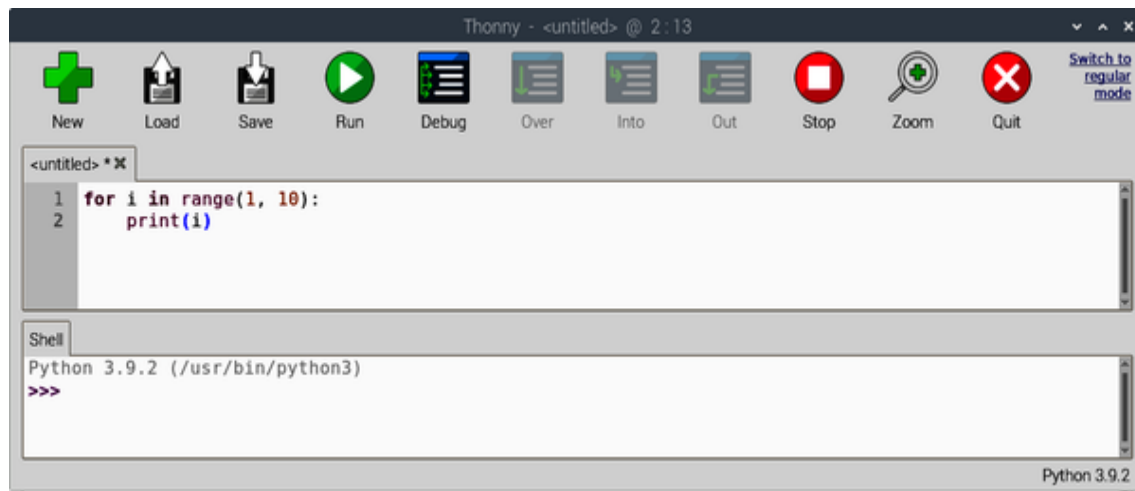


Figure 5-1. The Thonny editor

You can run the program to see what it does by clicking the Run button. Before it runs the program, Thonny will prompt you to save the file. Give it a name such as *count.py*. The program will then run and you'll see the output at the bottom of the Thonny window (Figure 5-2).



Figure 5-2. Running a program in Thonny

Although you can call your Python programs anything you like, the convention is that you give them the file extension `.py`.

## Discussion

We haven't started on Python programming yet, but from this example, you can see how a program is a matter of giving instructions to the computer in the Python programming language. In this case, the instructions are to print out a series of numbers.

## See Also

A popular alternative to Thonny is Mu ([Recipe 5.4](#)).

Visit the official [Thonny web page](#) for more information.

As well as using Mu to edit and run Python files, you can also edit Python files in nano ([Recipe 3.7](#)) and then run them from a Terminal session ([Recipe 5.6](#)).

## 5.4 Editing Python Programs with Mu

### Problem

You want to edit your Python programs using Mu.

### Solution

Mu is not pre-installed in the latest versions of Raspberry Pi OS. To install it, use the Recommended Software tool ([Recipe 4.2](#)). Once it's installed, you will find it in the Programming section of the Raspberry Menu ([Figure 5-3](#)).

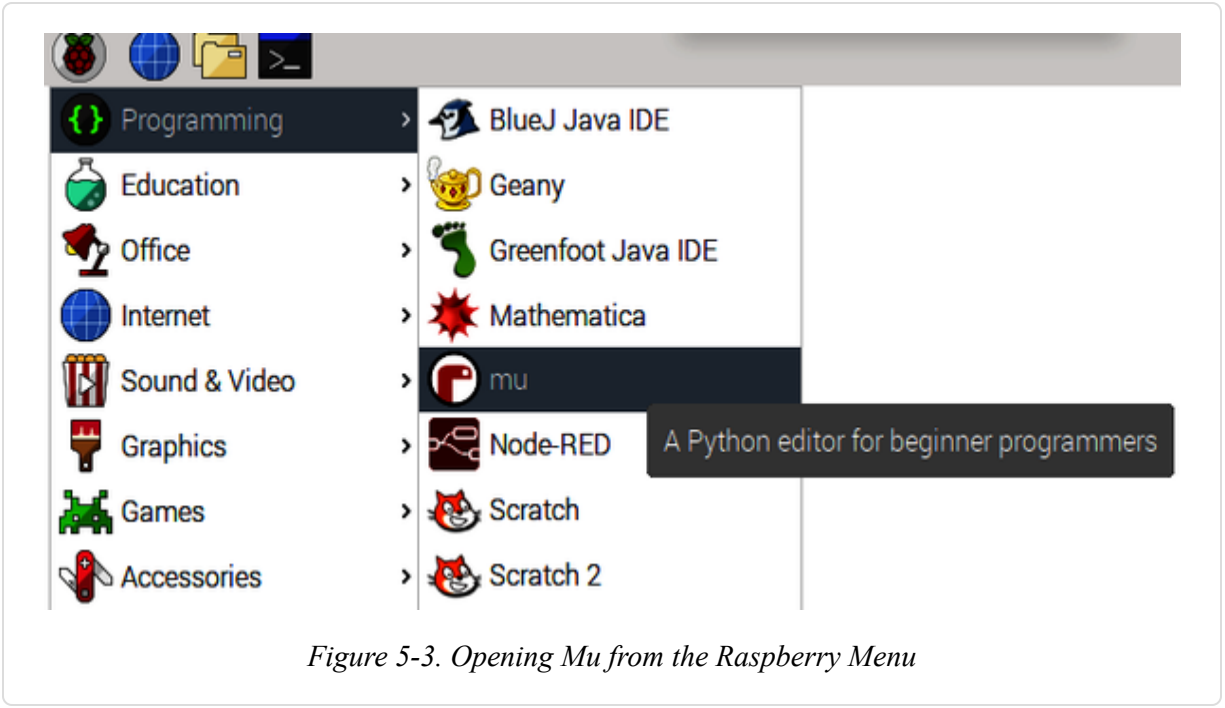


Figure 5-3. Opening Mu from the Raspberry Menu

When you first start Mu, you are prompted to select a mode (Figure 5-4).



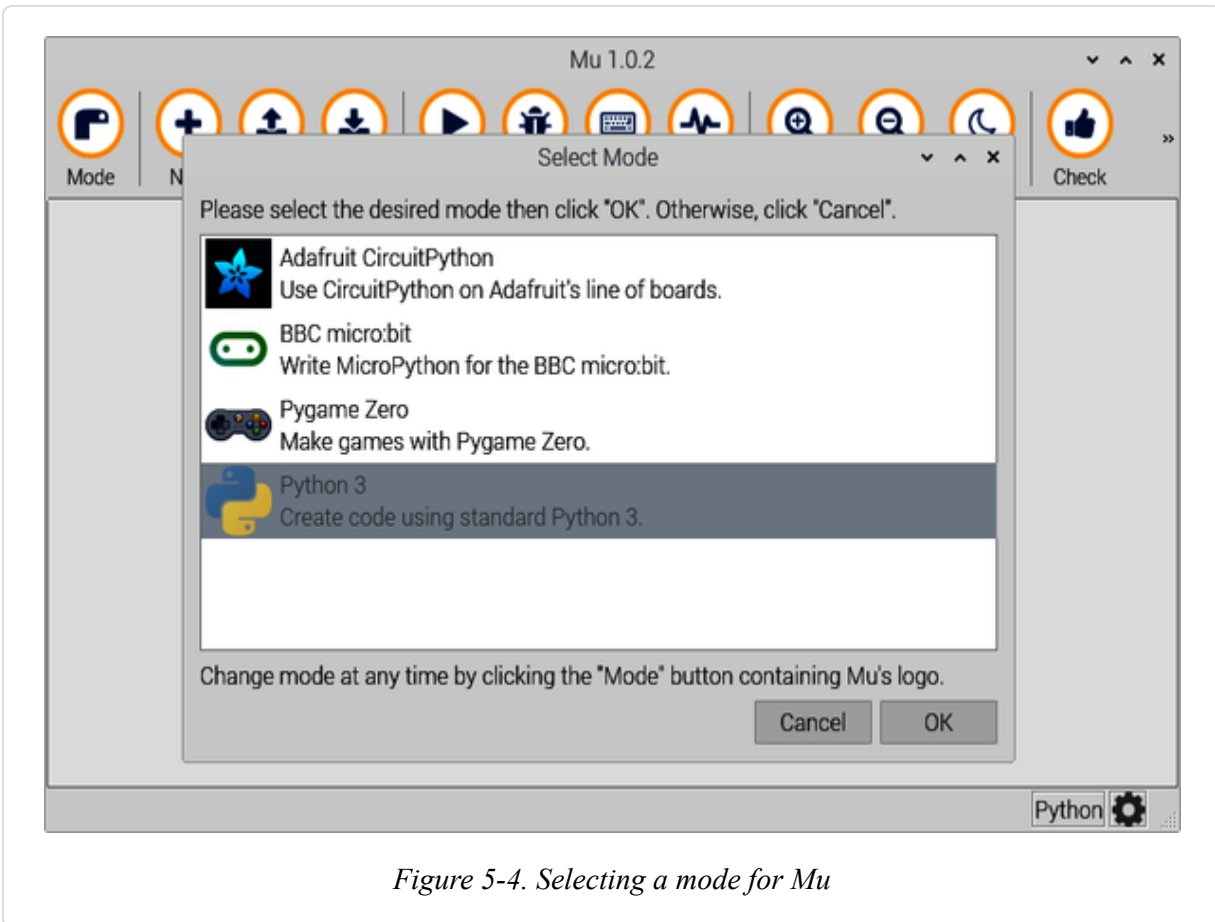


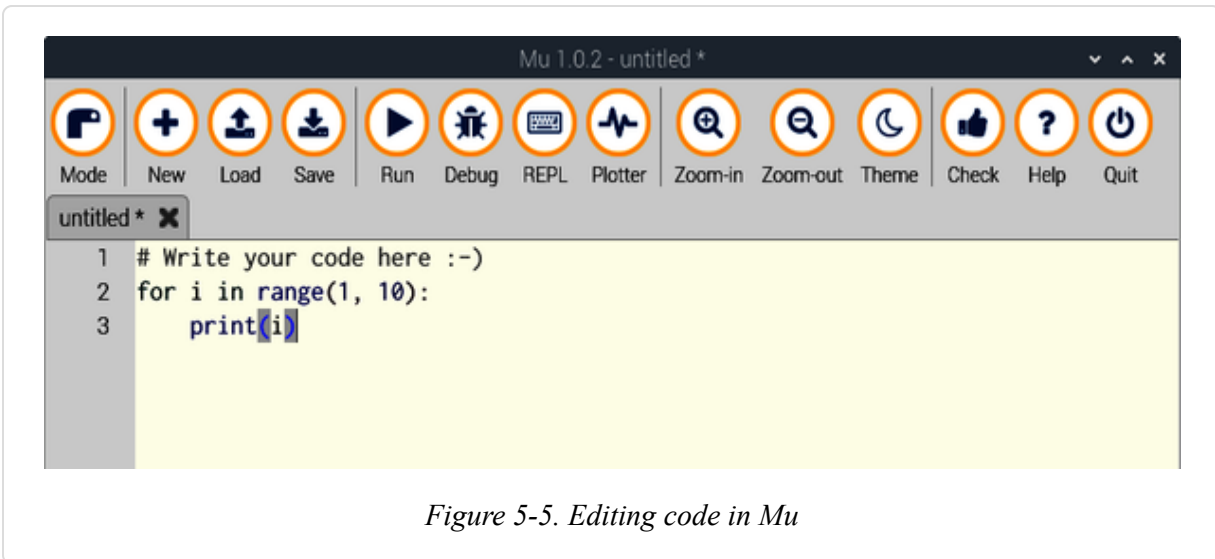
Figure 5-4. Selecting a mode for Mu

Select the Python 3 mode and click OK. This opens the Mu editor, so it's ready for you to start writing some Python.

Let's try it out. Carefully type the following test into the editor area under the comment "Write your code here :-)":

```
for i in range(1, 10):  
    print(i)
```

This short program will count to 9. Don't worry how it works for now; all will be explained in [Recipe 5.23](#). Note that when you get to the end of the first line and press Enter, the second line with the print statement should indent automatically ([Figure 5-5](#)).



*Figure 5-5. Editing code in Mu*

Before we can run the program we need to save it to a file, so at the top of the Mu window, click the Save button and then name the file `count.py` (Figure 5-6).

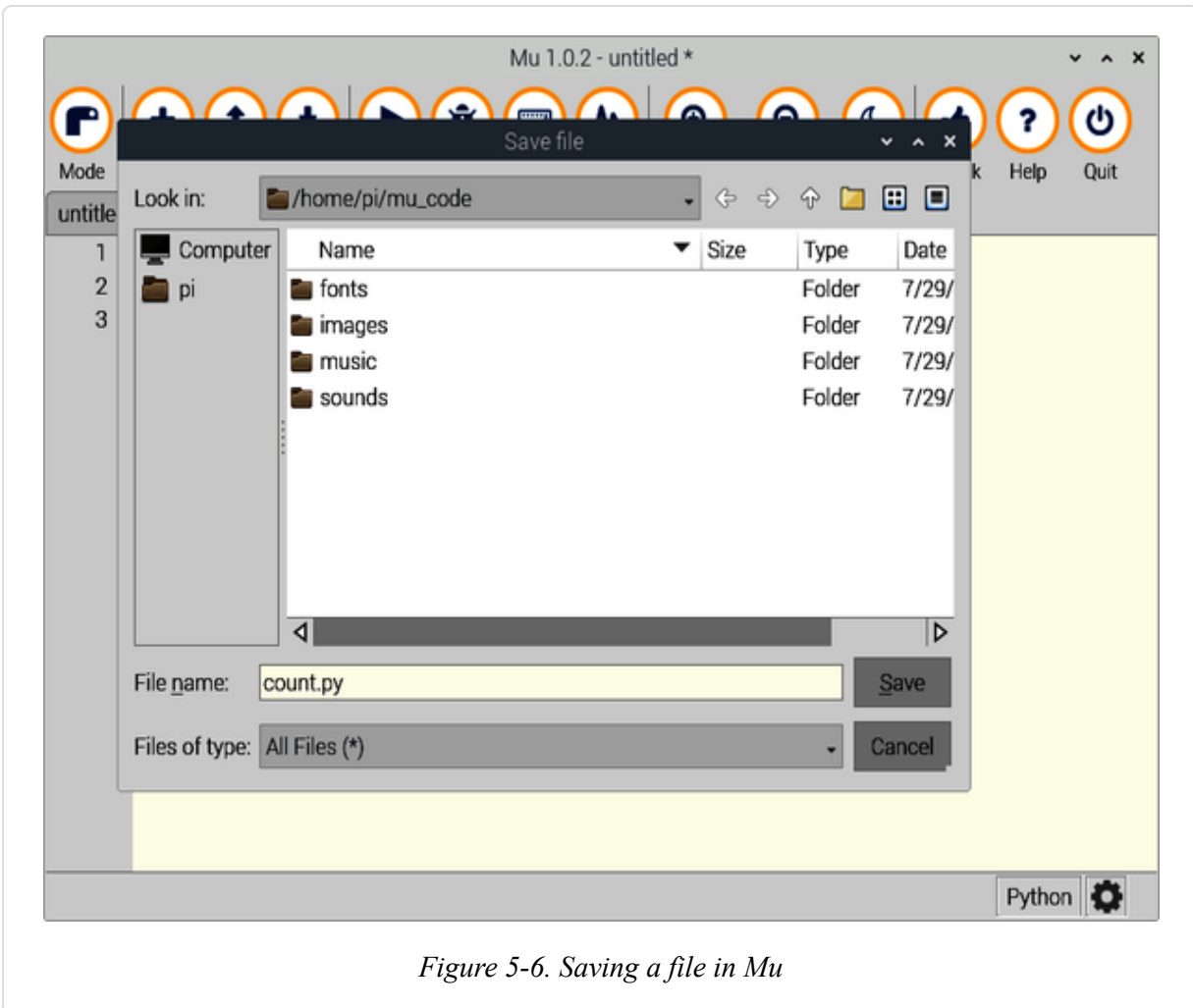


Figure 5-6. Saving a file in Mu

Now that the file is saved, run the program by clicking the Run button at the top of the Mu window. This causes the Mu editor screen to split, with the bottom half showing the result of running the program (Figure 5-7).

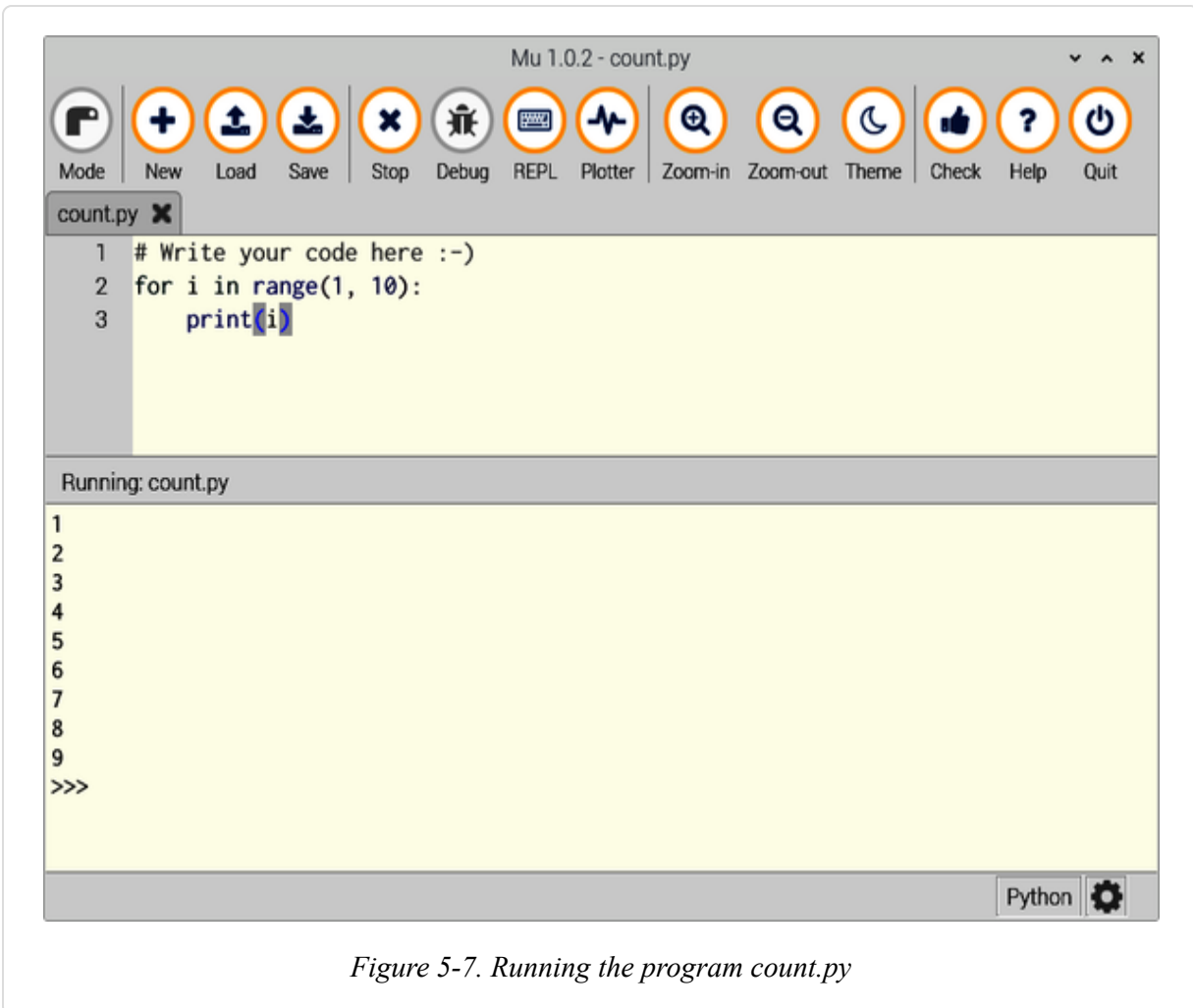


Figure 5-7. Running the program *count.py*

If you have already followed [Recipe 3.22](#) and downloaded the files accompanying this book, you can open these directly in Mu using the Open button and then navigating to the folder `~/raspberrypi_cookbook_ed4/python`, as shown in [Figure 5-8](#). Note that Mu is Python 3, and a few Python programs for this book work only with Python 2, so check the text of the recipe that the code belongs to if you have problems running it from Mu.

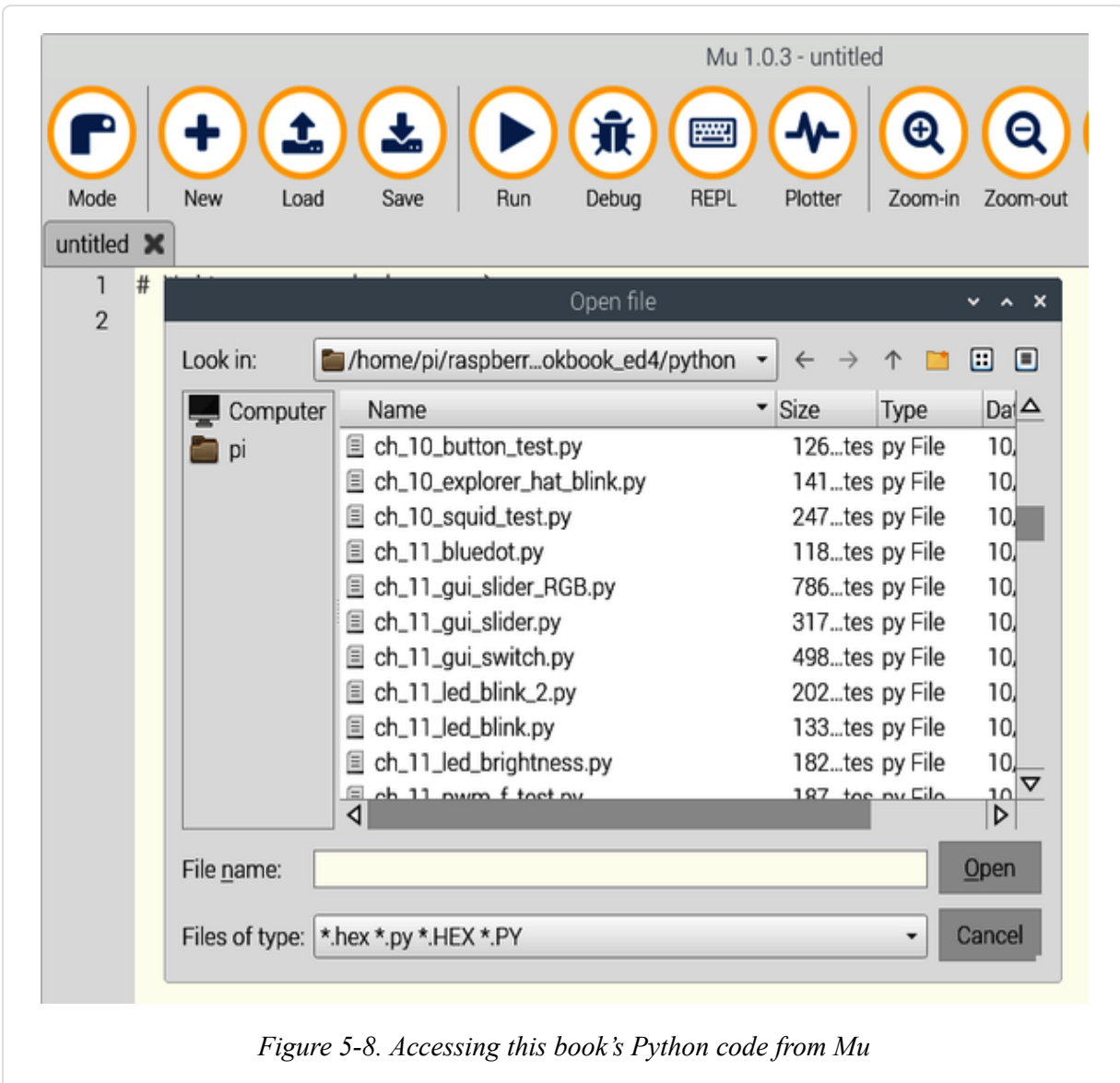


Figure 5-8. Accessing this book's Python code from Mu

## Discussion

Python is unusual for a programming language in that indentation is a fundamental part of the language. Whereas many C-based languages use `{` and `}` to delimit a block of code, Python uses the indentation level. So in the preceding example, Python knows that `print` is to be invoked repeatedly as part of the `for` loop because it is indented four spaces from the left.

When you're starting out in Python, it's not uncommon to see an error such as "IndentationError: unexpected indent," which means that something is

not indented correctly somewhere. If everything appears to line up, double-check that none of the indents contain tab characters because Python treats tabs differently.

You can use either spaces or tabs but you can't mix them in the same block, and it is very bad practice to mix them in the same program (even if Python allows you to do so).

In selecting Python 3 for our editing mode ([Figure 5-4](#)), we ignored the other mode options. The Adafruit CircuitPython mode allows you to use your Raspberry Pi to program Adafruit's range of CircuitPython boards, and the BBC micro:bit mode allows you to write MicroPython programs for a BBC micro:bit board. Both of these activities are about using other boards that are not covered in this book; however, it's good to know that options are available for using the Raspberry Pi with these microcontroller boards.

## See Also

The other popular Python editor for beginners is Thonny ([Recipe 5.3](#)).

As well as using Mu to edit and run Python files, you can also edit files in nano ([Recipe 3.7](#)) and then run them from a Terminal session ([Recipe 5.6](#)).

## 5.5 Using the Python Console

### Problem

You want to enter Python commands without writing an entire program. It can be useful to do this to experiment with some features of Python.

### Solution

Use the Python console, either within Thonny or in a Terminal session. The Python console provides a command line a little like that of Raspberry Pi OS ([Recipe 3.3](#)), but instead of entering operating system commands, you can enter Python commands. If you are using Mu ([Recipe 5.4](#)), you can

access the Python console by clicking the REPL (Read Eval Print Loop) button at the top of the Mu window (Figure 5-9).

Ignoring everything except the bottom of Figure 5-9, you can see a command prompt where you can type Python commands. In this case, I have typed the following after the `In [1] :` prompt:

```
2 + 2
```

and reassuringly received the answer:

```
4
```

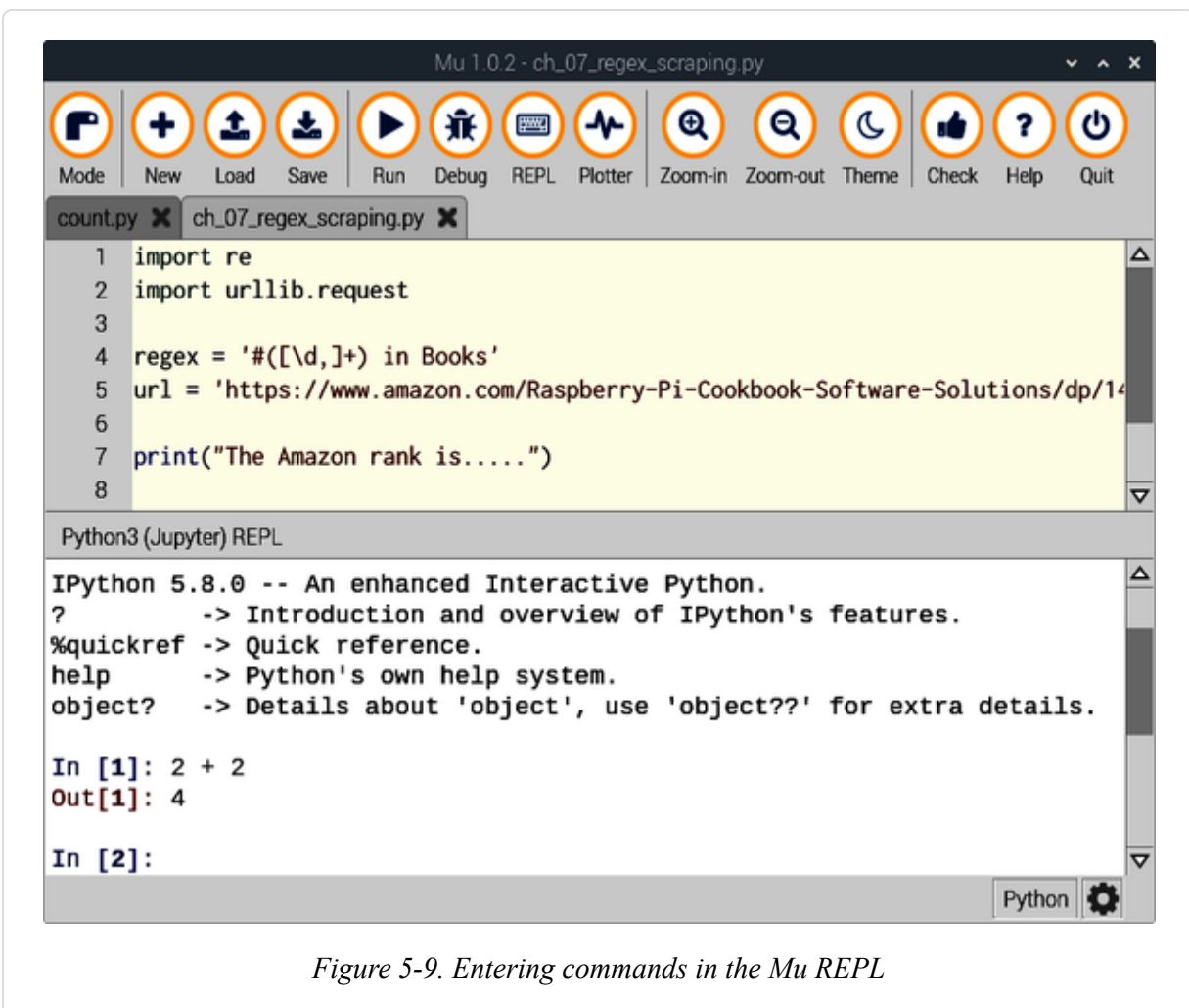


Figure 5-9. Entering commands in the Mu REPL

## Discussion

An alternative to using Mu to run individual Python commands is to start a Python 3 console in a Terminal window by typing the command **python3**. The `>>>` prompt indicates that you can type Python commands. If you need to type multiline commands, then the console will automatically provide a continuation line indicated by three dots. You still need to indent such lines by four spaces, as shown in the following session:

```
>>> from time import sleep
>>> while True:
...     print("hello")
...     sleep(1)
...
hello
hello
```

You need to press Enter twice after your last command for the console to recognize the end of the indented block and run the code.

The Python console also provides a command history so that you can move back and forth through your previous commands using the up and down arrow keys.

When you are finished with the Python console and want to return to the command line, type **exit()**.

### HELP IS AT HAND

The console can be a useful place to test out lines of code while you are developing a program, without having to run the whole program. You can get help on many Python features by typing `help(thing)` into the console, where *thing* is the thing you want help on.

For example, try typing `help(print)` into the Python console.

## See Also

If you have more than a couple of lines that you want to type in, chances are you would be better off using Thonny or Mu (Recipe [5.3](#) or [5.4](#)) to edit and



run a file.

## 5.6 Running Python Programs from the Terminal

### Problem

Running programs from within Thonny ([Recipe 5.3](#)) or Mu ([Recipe 5.4](#)) is fine, but sometimes you want to run a Python program from a Terminal window.

### Solution

Use the `python` or `python3` command in a Terminal, followed by the filename containing the program you want to run. (Raspberry Pi OS comes only with Python 3, so both the `python` or `python3` commands will run a Python 3 program. Which command you use is up to you.)

### Discussion

To run a Python 3 program from the command line, use a command like this:

```
$ python3 myprogram.py
```

If you want to run the program using Python 2, change the command `python3` to `python2` after installing Python 2 as described in [Recipe 5.1](#). In both cases the Python program that you want to run should be in a file with the extension `.py`.

You can run most Python programs as a normal user; however, some you'll need to run as superuser. If this is the case for your program, prefix the command with `sudo`: but be aware that if someone malicious wrote that

program, giving it such privileges could allow it to get up to all sorts of mischief:

```
$ sudo python3 myprogram.py
```

In the earlier examples, you need to include `python3` in the command to run the program, but you can optionally add a line to the start of a Python program so that Linux knows it is a Python program. This special line is called a *shebang* (a contraction of the names of two symbols, “hash” and exclamation mark, or “bang”). The following single-line example program illustrates this:

```
#!/usr/bin/python3
print("I'm a program, I can run all by myself")
```

Before you can run this directly from the command line, you must give the file write permissions by using the following command (see [Recipe 3.14](#)); this example assumes the file is called *test.py*:

```
$ chmod +x test.py
```

The parameter `+x` means to add execute permissions to the file.

Now you can run the Python program *test.py* using the single command:

```
$ ./test.py
I'm a program, I can run all by myself
$
```

The `./` at the start of the line is needed for the command line to find the file.

If you run a Python program with the `-i` option, the program will run and a console will open. This can be useful for debugging because you will have access to the program’s variables within the console, as if you had just typed the program directly into the console.

## See Also

[Recipe 3.25](#) shows you how to run a Python program as a timed event.

To automatically run a program at startup, see [Recipe 3.23](#).

# 5.7 Assigning Names to Values (Variables)

## Problem

You want to give a value a name.

## Solution

You assign a value to a name using `=`.

## Discussion

In Python, you don't have to declare the type of a variable; you can just assign it a value using the assignment operator (`=`), as shown in the following examples:

```
a = 123
b = 12.34
c = "Hello"
d = 'Hello'
e = True
```

You can define character-string constants using either single or double quotes. The logical constants in Python are `True` and `False`, and they are case sensitive.

By convention, variable names begin with a lowercase letter, and if the variable name consists of more than one word, the words are joined together with an underscore character. A variable name cannot start with a digit, but may include digits after the first character.

It is always a good idea to give your variables descriptive names so that when you come back to your program after a break, you can work out how it works.

Some examples of valid variable names are `x`, `total`, and `number_of_chars`.

## See Also

You also can assign a variable a value that is a list ([Recipe 6.1](#)) or a dictionary ([Recipe 6.13](#)).

For more information on arithmetic with variables, see [Recipe 5.10](#).

## 5.8 Displaying Output

### Problem

You want to see the value of a variable.

### Solution

Use the `print` command. You can try the following example in the Python console ([Recipe 5.5](#)):

```
>>> x = 10
>>> print(x)
10
>>>
```

Note that the `print` command starts a new line to print on.

### Discussion

In Python 2, you could use the `print` command without parentheses around the value. However, this is not true in Python 3, so for compatibility

with both versions of Python, always use parentheses around the value you are printing.

## See Also

To read user input, see [Recipe 5.9](#).

To better format what is printed, see Recipes [7.1](#) and [7.2](#).

## 5.9 Reading User Input

### Problem

You want to prompt the user to enter a value.

### Solution

Use the `input` (Python 3) command. You can try the following example in the Python 3 console ([Recipe 5.5](#)):

```
>>> x = input("Enter Value:")
Enter Value:23
>>> print(x)
23
>>>
```

### Discussion

In Python 3, `input` behaves quite differently than it did in Python 2. In Python 3, `input` returns a string, even if what was typed was a number. This was not the case in Python 2, where if the text looked like a number, it was converted to a number.

## See Also

Find more information on `input` in Python 2 at <https://oreil.ly/EhqMt>.

## 5.10 Using Arithmetic Operators

### Problem

You want to do arithmetic in Python.

### Solution

Use the +, -, \*, and / operators.

### Discussion

The most common operators for arithmetic are +, -, \*, and /, which are, respectively, add, subtract, multiply, and divide.

You can also group together parts of the expression with parentheses, as shown in the following example, which, given a temperature in degrees Celsius, converts it to degrees Fahrenheit:

```
>>> tempC = input("Enter temp in C: ")
Enter temp in C: 20
>>> tempF = (int(tempC) * 9) / 5 + 32
>>> print(tempF)
68.0
>>>
```

The `int` function converts the string of characters from `input` into an integer number.

Other arithmetic operators include `%` (modulo remainder) and `**` (raise to the power of). For example, to raise 2 to the power of 8, you would write the following:

```
>>> 2 ** 8
256
```

The order in which the arithmetic operators are evaluated can make a difference to the calculation. For example, which of these is the result of  $2 + 3 * 2$ :

- $(2 + 3) * 2 = 10$
- $2 + (3 * 2) = 8$

The answer for Python 3 is 8, because  $*$  is always done before  $+$ .

A mnemonic for remembering the order of precedence for operations is BODMAS:

- Brackets
- Orders (raising to power  $**$ )
- Division
- Multiplication
- Adding
- Subtracting

Something that you will often want to do when programming is to increase the value in a variable by a certain amount. If you have a variable called  $x$  that contains a number, you can add 1 to it using:

```
x = x + 1
```

Because this is such a common thing to want to do, there are shortcuts for operators like  $+$ ,  $-$ ,  $*$ , and  $/$  that both apply the operator and assign the new value. So to add 1 to  $x$ , you can also write:

```
x += 1
```

## See Also

See [Recipe 5.9](#) on using the `input` command, and [Recipe 5.14](#) on converting the string value from `input` to a number.

The `Math` library has many useful [math functions](#) that you can use.

## 5.11 Creating Strings

### Problem

You want to create a string variable—that is, a variable that contains text.

### Solution

Use the assignment operator (=) and a string constant to create a new string. You can use either double or single quotation marks around the string, but they must match. For example:

```
>>> s = "abc def"
>>> print(s)
abc def
>>>
```

### Discussion

If double or single quotes are *within* a string, then choose the other type of quotes as the beginning and ending markers of the string. For example:

```
>>> s = "Isn't it warm?"
>>> print(s)
Isn't it warm?
>>>
```

Sometimes you'll need to include special characters such as tab or newline inside your string. This requires the use of what are called *escape characters*. To include a tab, use `\t`, and for a newline, use `\n`. For example:

```
>>> s = "name\tage\nMatt\t14"
>>> print(s)
name    age
Matt    14
>>>
```



## See Also

For a full list of escape characters, see the [Python Reference Manual](#).

## 5.12 Concatenating (Joining) Strings

### Problem

You want to join a number of strings together.

### Solution

Use the + (concatenate) operator.

For example:

```
>>> s1 = "abc"
>>> s2 = "def"
>>> s = s1 + s2
>>> print(s)
abcdef
>>>
```

### Discussion

In many languages, you can have a chain of values to concatenate, some of which are strings and some of which are other types, such as numbers, and the numbers will automatically be converted into strings during the concatenation. This is not the case in Python, and if you try the following command, you will get an error:

```
>>> "abc" + 23
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

In Python, you must convert each component that you want to concatenate into a string before concatenating, as shown in this example:

```
>>> "abc" + str(23)
'abc23'
>>>
```

## See Also

See [Recipe 5.13](#) for more information about converting numbers into strings using the `str` function.

## 5.13 Converting Numbers into Strings

### Problem

You want to convert a number into a string.

### Solution

Use the `str` Python function.

For example:

```
>>> str(123)
'123'
>>>
```

### Discussion

A common reason for wanting to convert a number into a string is so you can then concatenate it with another string ([Recipe 5.12](#)).

## See Also

For the reverse operation of turning a string into a number, see [Recipe 5.14](#).

## 5.14 Converting Strings into Numbers

### Problem

You want to convert a string into a number.

### Solution

Use the `int` or `float` Python function.

For example, to convert the string `-123` into a number, you could use the following:

```
>>> int("-123")
-123
>>>
```

This will work on both positive and negative whole numbers.

To convert a floating-point number, use `float` instead of `int`:

```
>>> float("123.45")
123.45
>>>
```

### Discussion

Both `int` and `float` will handle leading zeros correctly and are tolerant of any spaces or other whitespace characters around the number.

You can also use `int` to convert a string representing a number in a number base other than the default of 10 by supplying the number base as the second argument. The following example converts the string representation of binary 1001 into a number:

```
>>> int("1001", 2)
9
>>>
```

This second example converts the hexadecimal number `AFF0` into an integer:

```
>>> int("AFF0", 16)
45040
>>>
```

## See Also

For the reverse operation of turning a number into a string, see [Recipe 5.13](#).

## 5.15 Finding the Length of a String

### Problem

You need to know how many characters there are in a string.

### Solution

Use the `len` Python function.

### Discussion

For example, to find the length of the string `abcdef`, you would use:

```
>>> len("abcdef")
6
>>>
```

## See Also

The `len` command also works on lists ([Recipe 6.3](#)).

## 5.16 Finding the Position of One String Within Another

### Problem

You need to find the position of one string within another string.

### Solution

Use the `find` Python function.

For example, to find the starting position of the string `def` within the string `abcdefghi`, you would use:

```
>>> s = "abcdefghi"
>>> s.find("def")
3
>>>
```

The character positions start at 0 (not 1), so a position of 3 means the fourth character in the string.

### Discussion

If the string you're looking for doesn't exist in the string being searched, `find` returns the value `-1`.

### See Also

The `replace` function is used to find and then replace all occurrences of a string ([Recipe 5.18](#)).

## 5.17 Extracting Part of a String

### Problem

You want to cut out a section of a string between certain character positions.

### Solution

Use the Python `[ : ]` *slice* notation.

For example, to cut out a section from the second character to the fifth character of the string `abcdefghi`, you would use the following:

```
>>> s = "abcdefghi"
>>> s[1:5]
'bcde'
>>>
```

The character positions start at 0 (not 1), so a position of 1 means the second character in the string, and 5 means the sixth character; however, the character range is exclusive (characters end at index position 4 not 5) at the high end. Thus, in this example, the letter *f* is not included, even though it is character 5.

### Discussion

The `[ : ]` notation is quite powerful. You can omit either argument, in which case the start or end of the string is assumed as appropriate. For example:

```
>>> s = "abcdefghi"
>>> s[:5]
'abcde'
>>>
```

and:

```
>>> s = "abcdefghi"
>>> s[3:]
'defghi'
>>>
```

You can also use negative indices to count back from the end of the string. This can be useful in situations such as when you want to find the three-letter extension of a file, as in the following example:

```
>>> "myfile.txt"[-3:]
'txt'
```

## See Also

[Recipe 5.12](#) describes joining strings together rather than splitting them.

[Recipe 6.11](#) uses the same syntax but with lists rather than strings.

Another and more powerful way to manipulate strings is described in [Recipe 7.23](#).

## 5.18 Replacing One String of Characters with Another Within a String

### Problem

You want to replace all occurrences of a string within another string.

### Solution

Use the `replace` function.

For example, to replace all occurrences of `X` with `times`, you would use the following:

```
>>> s = "It was the best of X. It was the worst of X"
>>> s.replace("X", "times")
```

```
'It was the best of times. It was the worst of times'  
>>>
```

The function `replace` takes two parameters. The first is the string to find, and the second is what it should be replaced with.

## Discussion

The string you're searching for must match exactly; that is, the search is case sensitive and will include spaces.

## See Also

See [Recipe 5.16](#) for searching for a string without performing a replacement.

Another and more powerful way to manipulate strings is described in [Recipe 7.23](#).

## 5.19 Converting a String to Uppercase or Lowercase

### Problem

You want to convert all the characters in a string to uppercase or lowercase letters.

### Solution

Use the `upper` or `lower` function as appropriate.

For example, to convert `aBcDe` to uppercase, you would use the following:

```
>>> "aBcDe".upper()  
'ABCDE'  
>>>
```



To convert it to lowercase, use this:

```
>>> "aBcDe".lower()
'abcde'
>>>
```

Note that even though `upper` and `lower` do not take any parameters, they still need a `()` on the end.

## Discussion

Like most functions that manipulate a string in some way, `upper` and `lower` do not actually modify the string but rather return a modified copy of the string.

For example, the following code returns a copy of the string `s`, but note how the original string is unchanged:

```
>>> s = "aBcDe"
>>> s.upper()
'ABCDE'
>>> s
'aBcDe'
>>>
```

To change the value of `s` to be all uppercase, do the following:

```
>>> s = "aBcDe"
>>> s = s.upper()
>>> s
'ABCDE'
>>>
```

## See Also

See [Recipe 5.18](#) for replacing text within strings.

## 5.20 Running Commands Conditionally (if)

### Problem

You want to run some Python commands only when some condition is true.

### Solution

Use the Python `if` command.

The following example will print the message `x is big` only if `x` has a value greater than 100:

```
>>> x = 101
>>> if x > 100:
...     print("x is big")
...
x is big
```

### Discussion

After the `if` keyword, there is a *condition*. This condition often, but not always, compares two values and gives an answer that is either `True` or `False`. If it is `True`, the subsequent indented lines will all be executed.

It is quite common to want to do one thing if a condition is `True` and something different if it is `False`. In this case, the `else` command is used with `if`, as shown in this example:

```
x = 101
if x > 100:
    print("x is big")
else:
    print("x is small")

print("This will always print")
```

You can also chain together a long series of `elif` (else if) conditions. If any one of the conditions succeeds, that block of code is executed, and none of the other conditions that follow it are tried.

For example:

```
x = 90
if x > 100:
    print("x is big")
elif x < 10:
    print("x is small")
else:
    print("x is medium")
```

This example will print `x is medium`.

## See Also

See [Recipe 5.21](#) for more information on different types of comparisons you can make.

## 5.21 Comparing Values

### Problem

You want to compare the values of two quantities.

### Solution

Use one of the comparison operators: `<`, `>`, `<=`, `>=`, `==`, or `!=`.

### Discussion

You used the `<` (less than) and `>` (greater than) operators in [Recipe 5.20](#). Here's the full set of comparison operators:

`<` Less than

---

>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Exactly equal to
!=	Not equal to

Some people prefer to use the <> operator in place of !=. Both work the same.

You can test these commands using the Python console ([Recipe 5.5](#)), as shown in the following exchange:

```
>>> 1 != 2
True
>>> 1 != 1
False
>>> 10 >= 10
True
>>> 10 >= 11
False
>>> 10 == 10
True
>>>
```

A common mistake is to use = (set a value) instead of == (double equals) in comparisons. This can be difficult to spot because if one half of the comparison is a variable, it is perfectly legal syntax and will run, but it will not produce the result you were expecting.

As well as comparing numbers, you can also compare strings by using these comparison operators, as shown here:

```
>>> 'aa' < 'ab'
True
>>> 'aaa' < 'aa'
False
```

The strings are compared lexicographically—that is, in the order that you would find them in a dictionary.

This is not quite correct because, for each letter, the uppercase version of the letter is considered less than the lowercase equivalent. Each letter has a value, called its **ASCII code**, and an uppercase letter has a lower numeric value than the lowercase version of the same letter.

For example:

```
>>> 'B' > 'a'
False
>>> 'B' > 'A'
True
```

## See Also

For more information using `if` command and logical operators, see Recipes [5.20](#) and [5.22](#).

Another and more powerful way to manipulate strings is described in [Recipe 7.23](#).

## 5.22 Using Logical Operators

### Problem

You need to specify a complex condition in an `if` statement.

### Solution

Use one of the logical operators: `and`, `or`, or `not`.

### Discussion

As an example, you might want to check whether a variable `x` has a value between 10 and 20. For that, you would use the `and` operator:

```
>>> x = 17
>>> if x >= 10 and x <= 20:
...     print('x is in the middle')
...
x is in the middle
```

You can combine as many `and` and `or` statements as you need, and you can also use brackets to group them if the expressions become complicated.

## See Also

For more information on the `if` command and comparing values, see Recipes [5.20](#) and [5.21](#).

## 5.23 Repeating Instructions an Exact Number of Times

### Problem

You need to repeat some program code an exact number of times.

### Solution

Use the Python `for` command and iterate over a range.

For example, to repeat a command 10 times, use the following:

```
>>> for i in range(1, 11):
...     print(i)
...
1
2
3
4
5
6
7
8
9
```

```
10
>>>
```

## Discussion

The second parameter in the `range` command is *exclusive*; that is, to count up to 10, you must specify a value of 11.

## See Also

If the condition for stopping the loop is more complicated than simply repeating the command a certain number of times, see [Recipe 5.24](#).

If you are trying to repeat commands for each element of a list or dictionary, see [Recipes 6.7](#) or [6.16](#), respectively.

## 5.24 Repeating Instructions Until Some Condition Changes

### Problem

You need to repeat some program code until something changes.

### Solution

Use the Python `while` statement. The `while` statement repeats its nested commands until its condition becomes false. The following example will stay in the loop until the user enters **X** for exit:

```
>>> answer = ''
>>> while answer != 'X':
...     answer = input('Enter command:')
...
Enter command:A
Enter command:B
Enter command:X
>>>
```

## Discussion

Note that `answer` is given an initial value of an empty string, before the `while` loop starts. If you did not do this, the `while` line would cause an error, because at that point, `answer` would be undefined and not have a value to compare.

## See Also

If you just want to repeat some commands a certain number of times, see [Recipe 5.23](#).

If you are trying to repeat commands for each element of a list or dictionary, see [Recipes 6.7](#) or [6.16](#), respectively.

## 5.25 Breaking Out of a Loop

### Problem

You are in a loop and need to exit the loop if some condition occurs.

### Solution

Use the Python `break` statement to exit either a `while` or a `for` loop.

The following example behaves in exactly the same way as the example code in [Recipe 5.24](#):

```
>>> while True:
...     answer = input('Enter command:')
...     if answer == 'X':
...         break
...
Enter command:A
Enter command:B
Enter command:X
>>>
```



The line `while True:` looks a bit odd at first. This just means repeat forever, or until you jump out of the loop by some other means.

## Discussion

This example uses the `input` command as it works in Python 3. To run the example in Python 2, substitute the command `raw_input` for `input`.

This example behaves in exactly the same way as the example in [Recipe 5.24](#). However, in this case, the condition for the `while` loop is just `True`, so the loop will never end unless we use `break` to exit the loop when the user enters **X**.

## See Also

You can also leave a `while` loop by using its condition; see [Recipe 5.24](#).

# 5.26 Defining a Function in Python

## Problem

You want to avoid repeating the same code over and over in a program.

## Solution

Create a function that groups together lines of code, allowing it to be called from multiple places.

The following example illustrates how to create and then call a function in Python:

```
def count_to_10():
    for i in range(1, 11):
        print(i)
```

This example uses the `def` command to define a function called `count_to_10`, which will print out the numbers 1 to 10 whenever it is called:

```
>>> count_to_10()
1
2
3
4
5
6
7
8
9
10
>>>
```

## Discussion

The conventions for naming functions are the same as for variables in [Recipe 5.7](#); that is, they should start with a lowercase letter, and if the name consists of more than one word, the words should be separated by underscores.

The example function is a little inflexible because it can only count to 10. If we wanted to make it more flexible—for example, so it could count up to any number—we could include the maximum number as a *parameter* to the function, as this example illustrates:

```
def count_to_n(n):
    for i in range(1, n + 1):
        print(i)

count_to_n(5)
>>> count_to_10()
1
2
3
4
5
>>>
```

The parameter `n` is included inside the parentheses and then used inside the `range` command, but not before `1` is added to it.

Using a parameter for the number you want to count up to means that if you usually count to 10 but sometimes count to a different number, you will always have to specify the number. You can, however, specify a default value for a parameter, and hence have the best of both worlds, as shown in this example:

```
def count_to_n(n=10):
    for i in range(1, n + 1):
        print(i)

count_to_n()
```

This will now count to 10 unless a different number is specified when you call the function.

If your function needs more than one parameter, perhaps to count between two numbers, the parameters are separated by commas:

```
def count(from_num=1, to_num=10):
    for i in range(from_num, to_num + 1):
        print(i)

count()
1
2
3
4
5
6
7
8
9
10
>>>
count(5)
1
2
3
4
5
```

```
>>>
count(5, 10)
5
6
7
8
9
10
>>>
```

If you want some parameters to have default values and others not, the ones with defaults have to come after the ones without. That is, `count(from_num, to_num=10)` is allowed but `count(from_num=1, to_num)` is not allowed.

All these examples are functions that do not return any value; they just do something. If you need a function to return a value, you need to use the `return` command.

The following function takes a string as an argument and adds the word *please* to the end of the string:

```
def make_polite(sentence):
    return sentence + " please"

print(make_polite("Pass the cheese"))
```

When a function returns a value, you can assign the result to a variable, or, as in this example, you can print out the result.

## See Also

To return more than one value from a function, see [Recipe 7.3](#).

# Chapter 6. Python Lists and Dictionaries

---

## 6.0 Introduction

In [Chapter 5](#), we looked at the basics of the Python language. In this chapter, we will look at two key Python data structures: *lists* and *dictionaries*.

## 6.1 Creating a List

### Problem

You want to use a variable to hold a series of values rather than just one value.

### Solution

Use a list. In Python, a list is a collection of values stored in a specific order so that you can access them by position.

You create a list by using the [ and ] characters to contain its initial contents:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>>
```

Unlike the more rigid arrays in languages like C, you don't need to specify the size of a list in Python when you declare it. You can also change the number of elements in the list any time you like.

## Discussion

As this example illustrates, the items in a list do not need to be the same type, although they often are. In fact its quite common for the elements of a list to themselves be lists.

To create an empty list that you can add items to later, you can use:

```
>>> a = []
>>>
```

## 6.2 Accessing Elements of a List

### Problem

You want to find individual elements of a list or change them.

### Solution

Use the `[]` notation to access elements of a list by their position in the list. For example, to access the element at position 1 in a list:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> a[1]
'Fred'
```

### Discussion

The list positions (indices) start at 0 for the first element (not at 1).

As well as using the `[]` notation to read values out of a list, you can also use it to change values at a certain position, as shown here:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> a[1] = 777
>>> a
[34, 777, 12, False, 72.3]
```

If you try to change (or, for that matter, read) an element of a list using an index that is too large, you will get an “index out of range” error:

```
>>> a[50] = 777
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
>>>
```

A handy quirk of Python lists is that you can also access elements starting from the end (right) of the list using negative indices. The index position of -1 is the last element of the list, and -2 the next to the last, and so on. An example of this is:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> a[-1]
72.3
>>> a[-2]
False
>>>
```

## 6.3 Finding the Length of a List

### Problem

You need to know how many elements there are in a list.

### Solution

Use the `len` Python function.

For example:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> len(a)
5
```

The function `len` has been in the Python language from version 1 and is rather contrary to the more object-oriented, class-based versions 2 and 3 of Python. For these, it would make more sense for you to be able to write something like:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> a.length() # This example won't work
```

But you can't—that's just how it is with Python.

## Discussion

The `len` command also works on strings ([Recipe 5.15](#)).

## 6.4 Adding Elements to a List

### Problem

You need to add an item to a list.

### Solution

Use the `append`, `insert`, or `extend` Python functions.

To add a single item to the end of a list, use `append`, as shown here:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> a.append("new")
>>> a
[34, 'Fred', 12, False, 72.3, 'new']
```

### Discussion

Sometimes you don't want to add the new elements to the end of a list, but instead want to insert them at a certain position in the list. For this, use the



`insert` command. The first argument is the index where the item should be inserted, and the second argument is the item to be inserted:

```
>>> a.insert(2, "new2")
>>> a
[34, 'Fred', 'new2', 12, False, 72.3]
```

Note how all the elements after the newly inserted element are shifted up one position.

Both `append` and `insert` add only one element to a list. The `extend` function adds all the elements of one list to the end of another:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> b = [74, 75]
>>> a.extend(b)
>>> a
[34, 'Fred', 12, False, 72.3, 74, 75]
```

An alternative to using `append` is to use `+=` as the following example illustrates:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> b = [74, 75]
>>> a += b
>>> a
[34, 'Fred', 12, False, 72.3, 74, 75]
```

## 6.5 Removing Elements from a List

### Problem

You need to remove an item from a list.

### Solution

Use the `pop` Python function.

The command `pop` with no parameters removes the last element of a list:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> a.pop()
72.3
>>> a
[34, 'Fred', 12, False]
```

## Discussion

Notice that `pop` returns the value removed from the list.

To remove an item in a position other than the last element, use `pop` with a parameter indicating the position from which the item should be removed:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> a.pop(0)
34
```

If you use an index position that is beyond the end of the list, you will get an “Index out of range” error.

If you want to remove something from a list by its value rather than its position in the list, you can use `remove`, as the following example illustrates:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> a.remove(12)
a
[34, 'Fred', False, 72.3]
>>>
```

## 6.6 Creating a List by Parsing a String

### Problem

You need to convert a string of words separated by some character into an array of strings, with each string in the array being one of the words.

## Solution

Use the `split` Python string function.

The command `split` with no parameters separates out the words of a string into individual elements of an array:

```
>>> "abc def ghi".split()
['abc', 'def', 'ghi']
```

If you supply `split` with a parameter, it will split the string using the parameter as a separator.

For example:

```
>>> "abc--de--ghi".split('--')
['abc', 'de', 'ghi']
```

## Discussion

This command can be very useful when you are, for example, importing data from a file. The `split` command can optionally take an argument that is the string to use as a delimiter when you are splitting the string. So if you were to use commas as a separator, you could split the string as follows:

```
>>> "abc,def,ghi".split(',')
['abc', 'def', 'ghi']
```

If you want to go the other way and convert your list of strings into a single string, you can use the `join` command as shown:

```
>>> a = ['abc', 'def', 'ghi']
```

```
>>> "".join(a)
'abcdefghi'
```

## See Also

Another and more powerful way to manipulate strings is described in [Recipe 7.23](#).

## 6.7 Iterating Over a List

### Problem

You need to apply some lines of code to each item of a list in turn.

### Solution

Use the `for` Python command:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> for x in a:
...     print(x)
...
34
Fred
12
False
72.3
>>>
```

### Discussion

The `for` keyword is immediately followed by a variable name (in this case, `x`). This is called the *loop variable*; it will be set to each element of the list specified after `in`.

The indented lines that follow will be executed one time for each element in the list. Each time through the loop, `x` will be given the value of the element

in the list at that position. You can then use `x` to print out the value, as shown in the example.

## See Also

Comprehensions are another way to manipulate lists (see [Recipe 6.12](#)).

## 6.8 Enumerating a List

### Problem

You need to run some lines of code for each item in a list in turn, but you also need to know the index position of each item.

### Solution

Use the `for` Python language along with the `enumerate` command:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> for (i, x) in enumerate(a):
...     print(i, x)
...
(0, 34)
(1, 'Fred')
(2, 12)
(3, False)
(4, 72.3)
>>>
```

### Discussion

It's quite common to need to know the position of something in a list while enumerating each of the values. An alternative method is to simply count with an index variable and then access the value using the `[]` syntax:

```
>>> a = [34, 'Fred', 12, False, 72.3]
>>> for i in range(len(a)):
...     print(i, a[i])
...
(0, 34)
(1, 'Fred')
(2, 12)
(3, False)
(4, 72.3)
>>>
```

```
...     print(i, a[i])
...
(0, 34)
(1, 'Fred')
(2, 12)
(3, False)
(4, 72.3)
>>>
```

## See Also

Comprehensions are another way to manipulate lists (see [Recipe 6.12](#)).

See [Recipe 6.7](#) to iterate over a list without needing to know each item's index position.

## 6.9 Testing if Something Is in a List

### Problem

You want to know if a list contains a certain element.

### Solution

Use the `in` keyword as the following example illustrates:

```
>>> x = [12, 66, 32, 6, 99]
>>> 66 in x
True
>>> 77 in x
False
>>>
```

### Discussion

Although you could iterate over the list looking for the element, using the `in` command will make your code simpler and easier to read.

The `in` command also works on strings.

## See Also

For iterating over the elements of a list, see [Recipe 6.7](#)

## 6.10 Sorting a List

### Problem

You need to sort the elements of a list.

### Solution

Use the `sort` Python command:

```
>>> a = ["it", "was", "the", "best", "of", "times"]
>>> a.sort()
>>> a
['best', 'it', 'of', 'the', 'times', 'was']
```

The `sort` command uses the standard Python comparison operators. This means that for strings, list elements will be sorted alphabetically in ascending order.

### Discussion

When you sort a list, you're actually modifying it rather than returning a sorted copy of the original list. This means that if you also need the original list, you need to use the `copy` command in the standard library to make a copy of the original list before sorting it:

```
>>> from copy import copy
>>> a = ["it", "was", "the", "best", "of", "times"]
>>> b = copy(a)
>>> b.sort()
>>> a
['it', 'was', 'the', 'best', 'of', 'times']
>>> b
```

```
['best', 'it', 'of', 'the', 'times', 'was']  
>>>
```

The `copy` module is required to be able to copy objects. You can find out more about modules in [Recipe 7.11](#).

## 6.11 Cutting Up a List

### Problem

You need to make a sublist of a list, using a range of the original list's elements.

### Solution

Use the `[ : ]` Python construction. The following example returns a list containing the elements of the original list from index position 1 to index position 2 (the number after the `:` is exclusive—that is, elements up to but not including element 2:

```
>>> l = ["a", "b", "c", "d"]  
>>> l[1:3]  
['b', 'c']
```

Note that the character positions start at 0 (not at 1), so a position of 1 means the second character in the string, and 3 means the fourth; however, the character range is *exclusive* at the high end, so the letter *d* is not included in this example.

### Discussion

The `[ : ]` notation is quite powerful. You can omit either argument, in which case the start or end of the list is assumed, as appropriate.

For example:



```
>>> l = ["a", "b", "c", "d"]
>>> l[:3]
['a', 'b', 'c']
>>> l[3:]
['d']
>>>
```

You can also use negative indices to count back from the end of the list. The following example returns the last two elements in the list:

```
>>> l[-2:]
['c', 'd']
```

Incidentally, `l[:-2]` returns `['a', 'b']` in the preceding example.

## See Also

See [Recipe 5.17](#), in which the same syntax is used for strings.

## 6.12 Using Comprehensions

### Problem

You want a neater way of building one list from another, while filtering or transforming the original list.

### Solution

Use a *comprehension*.

Comprehensions don't do anything that you can't do with a regular loop, but they do simplify activities involving lists and, when well used, make your code more readable.

A comprehension will take an existing list and create a new list from the elements of the original list, either filtering the list so that only qualifying elements are included in the new list, or creating a new list containing the

same number of elements of the original list but with each element manipulated in some way.

As a reminder, without using a comprehension, the following code will take a list and filter it into a new list that just contains the elements that begin with the letter *a*:

```
new_list = []
people = ['agnes', 'andrew', 'jane', 'peter']
for person in people:
    if person[0] == 'a':
        new_list.append(person)
print(new_list)
```

When you run this program (*ch\_06\_filter.py*) you will see that `new_list` just contains the names starting with *a*:

```
$ python3 ch_06_filter.py
['agnes', 'andrew']
```

We can shorten this example considerably by using a comprehension (*ch\_06\_filter\_comp.py*):

```
people = ['agnes', 'andrew', 'jane', 'peter']
new_list = [person for person in people if person[0] == 'a']
print(new_list)
```

This will produce exactly the same output as *ch\_06\_filter.py*, using fewer lines of code. The comprehension is contained in square brackets, and the first word in the comprehension is `person`. This indicates that `person` (yet to be specified) is what will be added to the list copy. We then have `for person in people`, which is the normal way of iterating over a list, with the list being called `people` and each element in the list being called `person`. The final part of the comprehension is the condition `if person[0] == 'a'`. This is the part that rejects all the elements of the list that don't start with *a*.

The condition of a comprehension is optional and you may just want to use the comprehension to modify each element of the list. The example of *ch\_06\_change\_comp.py* capitalizes the names and returns a new list, with the initial letter of each name in uppercase:

```
people = ['agnes', 'andrew', 'jane', 'peter']
new_list = [person.capitalize() for person in people]
print(new_list)
```

```
$ python3 ch_06_change_comp.py
['Agnes', 'Andrew', 'Jane', 'Peter']
```

## Discussion

Comprehensions are a really powerful and useful technique for manipulating lists. At first the syntax might look a little strange for being enclosed in square brackets. But it is a great way of keeping your code concise without making it any harder to understand how it works.

## See Also

For iterating over a list without using a comprehension, see [Recipe 6.7](#).

## 6.13 Creating a Dictionary

### Problem

You need to create a lookup table in which you associate values with keys.

### Solution

Use a Python dictionary.

Lists are great when you need to access a list of items in order, or when you always know the index of the element that you want to use. Dictionaries are

an alternative to lists for storing collections of data, but they are organized very differently, as shown in **Figure 6-1**.

phone\_numbers

Key: Simon	Value: 01234 567899
Key: Jane	Value: 01234 666666
Key: Pete	Value: 01234 777555
Key: Linda	Value: 01234 887788

*Figure 6-1. A Python dictionary*

A dictionary stores key/value pairs in such a way that you can use the key to retrieve that value very efficiently and without having to search the entire dictionary.

To create a dictionary, you use the { } notation:

```
>>> phone_numbers = {'Simon': '01234 567899', 'Jane': '01234  
666666'}
```

## Discussion

In this example, the keys of the dictionary are strings, but they do not need to be; they could be numbers or any data type, although strings are most commonly used.

The values can also be of any data type, including other dictionaries or lists. The following example creates one dictionary (a) and then uses it as a value in a second dictionary (b):

```
>>> a = {'key1': 'value1', 'key2': 2}
```

```
>>> a
{'key2': 2, 'key1': 'value1'}
>>> b = {'b_key1':a}
>>> b
{'b_key1': {'key2': 2, 'key1': 'value1'}}
```

When you display the contents of a dictionary, notice that the order of the items in the dictionary might not match the order in which they were specified when the dictionary was created and initialized with some content:

```
>>> phone_numbers = {'Simon':'01234 567899', 'Jane':'01234
666666'}
>>> phone_numbers
{'Jane': '01234 666666', 'Simon': '01234 567899'}
```

Unlike lists, dictionaries have no concept of keeping items in order. Because of the way they are represented internally, the order of a dictionary's contents will be—for all intents and purposes—random.

The reason the order appears to be random is that the underlying data structure is a *hash table*. Hash tables use a `hashing` function to decide where to store each value; the `hashing` function calculates a numeric equivalent to any object.

You can find out more about hash tables on [Wikipedia](#).

## See Also

Dictionaries have much in common with the JSON data structuring language described in [Recipe 7.20](#).

## 6.14 Accessing a Dictionary

### Problem

You need to find and change entries in a dictionary.

## Solution

Use the Python `[]` notation. Specify the key of the entry to which you need access within the brackets, as follows:

```
>>> phone_numbers = {'Simon':'01234 567899', 'Jane':'01234
666666'}
>>> phone_numbers['Simon']
'01234 567899'
>>> phone_numbers['Jane']
'01234 666666'
```

## Discussion

The lookup process is in one direction only, from key to value.

If you use a key that is not present in the dictionary, you will get a “`KeyError`.” For example:

```
>>> phone_numbers = {'Simon':'01234 567899', 'Jane':'01234
666666'}
>>> phone_numbers['Phil']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'Phil'
>>>
```

As well as using the `[]` notation to read values from a dictionary, you can also use it to add new values or overwrite existing ones.

The following example adds a new entry to the dictionary with a key of Pete and a value of 01234 777555:

```
>>> phone_numbers = {'Simon':'01234 567899', 'Jane':'01234
666666'}
>>> phone_numbers['Pete'] = '01234 777555'
>>> phone_numbers['Pete']
'01234 777555'
```

If the key is not in use in the dictionary, a new entry is automatically added. If the key is already present, then whatever value was there before will be overwritten by the new value.

## See Also

For information on handling errors, see [Recipe 7.10](#).

## 6.15 Removing Entries from a Dictionary

### Problem

You need to remove an item from a dictionary.

### Solution

Use the `pop` command, specifying the key for the item that you want to remove:

```
>>> phone_numbers = {'Simon': '01234 567899', 'Jane': '01234
666666'}
>>> phone_numbers.pop('Jane')
'01234 666666'
>>> phone_numbers
{'Simon': '01234 567899'}
```

### Discussion

The `pop` command returns the value of the item removed from the dictionary. In the preceding example, even though Jane is removed from the dictionary, the `pop` command returns the Jane entry in case you want it for some other purpose.

## 6.16 Iterating Over Dictionaries

## Problem

You need to do something to each of the items in the dictionary in turn.

## Solution

Use the `for` command to iterate over the keys of the dictionary:

```
>>> phone_numbers = {'Simon':'01234 567899', 'Jane':'01234
666666'}
>>> for name in phone_numbers:
...     print(name)
...
Jane
Simon
```

Notice how the keys didn't print in the same order in which they were created. This is a feature of dictionaries. The order of the entries is not remembered.

## Discussion

You can use a couple of other techniques to iterate over a dictionary. The following form can be useful if you need access to the values as well as the keys:

```
>>> phone_numbers = {'Simon':'01234 567899', 'Jane':'01234
666666'}
>>> for name, num in phone_numbers.items():
...     print(name + " " + num)
...
Jane 01234 666666
Simon 01234 567899
```

## See Also

See the `for` command used elsewhere in Recipes [5.23](#) and [6.7](#).



# Chapter 7. Advanced Python

---

## 7.0 Introduction

In this chapter, we'll explore some of the more advanced concepts in the Python language—in particular reading and writing files, handling exceptions, using modules, and basic internet programming.

Although we have already met various aspects of object orientation, classes, and methods, in this chapter we will examine them in more detail and explain what is going on.

## 7.1 Formatting Numbers

### Problem

You want to format numbers to a certain number of decimal places.

### Solution

Apply a `format` string to the number.

For example:

```
>>> x = 1.2345678
>>> "x={:.2f}".format(x)
'x=1.23'
>>>
```

The result returned by the `format` method is a string, which will be displayed in the Terminal because we are working interactively. However, when using `format` in a program, it's most likely to be used inside a `print` statement, like this:

```
x = 1.2345678
print("x={:.2f}".format(x))
```

## Discussion

The formatting string can contain a mixture of regular text and markers delimited by { and }. The parameters to the `format` function (there can be as many as you like) will be substituted in place of the marker, according to the format specifier.

In the preceding example, the format specifier is `:.2f`, which means that the number will be specified with two digits after the decimal place and is a float, `f`.

If you wanted the number to be formatted so that the total length of the number is always seven digits (or padding spaces), you would add another number before the decimal place, like this:

```
>>> "x={:7.2f}".format(x)
'x=  1.23'
>>>
```

In this case, since the number is only three digits long, there are four spaces of padding before the 1. If you wanted the padding to take the form of leading zeros, you would use:

```
>>> "x={:07.2f}".format(x)
'x=0001.23'
>>>
```

A more complicated example might be to display the temperature in both degrees Celsius and degrees Fahrenheit, as shown here:

```
>>> c = 20.5
>>> "Temperature {:.2f} deg C, {:.2f} deg F.".format(c, c * 9 /
5 + 32)
```

```
'Temperature 20.50 deg C, 68.90 deg F.'  
>>>
```

You can also use the `format` method to display numbers in hexadecimal and binary.

For example:

```
>>> "{:X}".format(42)  
'2A'  
>>> "{:b}".format(42)  
'101010'
```

Since version 3.6 of Python, there has been a new way of formatting strings and other objects called *f-strings*. These allow you to place bits of Python (often just a variable name) inside a Python string for evaluation. For example:

```
>>> temp = 20.4  
>>> humidity = 80  
>>> F"Temperature C: {temp} Humidity: {humidity}"  
'Temperature C: 20.4 Humidity: 80'  
>>>
```

The f-string uses an `F` in front of the opening string quote mark and anything between `{` and `}` is treated as Python code to be evaluated. So, if you wanted your temperature in Fahrenheit, you could write:

```
>>> temp = 20.4  
>>> humidity = 80  
>>> F"Temperature F: {temp * 9 / 5 + 32} Humidity: {humidity}"  
'Temperature F: 68.72 Humidity: 80'  
>>>
```

The f-string syntax is in many cases much easier to read than the string format method.

## See Also

Formatting in Python involves a [whole formatting language](#).

For more information on f-strings, see <https://realpython.com/python-f-strings>.

## 7.2 Formatting Dates and Times

### Problem

You want to convert a date into a string and format it in a certain way.

### Solution

Apply a `format` string to the date object.

For example:

```
>>> from datetime import datetime
>>> d = datetime.now()
>>> "{:%Y-%m-%d %H:%M:%S}".format(d)
'2021-12-09 16:00:45'
>>>
```

The result returned by the `format` method is a string, which will be displayed in the Terminal because we are working interactively. However, when using `format` in a program, it's most likely to be used inside a `print` statement, like this:

```
from datetime import datetime
d = datetime.now()
print("{:%Y-%m-%d %H:%M:%S}".format(d))
```

### Discussion

The Python formatting language includes some special symbols for formatting the date: `%y` (which gives the year without century as a zero-padded decimal number), `%m`, and `%d` correspond to year, month, and day numbers, respectively.

Other symbols useful for formatting the date are `%B`, which supplies the full name of the month, and `%Y`, which gives the year in four-digit format, as shown here:

```
>>> "{:%d %B %Y}".format(d)
'09 December 2021'
```

The f-string syntax described at the end of [Recipe 7.1](#) can also be used to format dates as follows:

```
>>> from datetime import datetime
>>> d = datetime.now()
>>> F"{d:%B %d, %Y}"
'August 19, 2022'
>>>
```

## See Also

See [Recipe 7.1](#) for formatting of numbers.

See the [Python strftime cheatsheet](#) for more about all the options for formatting dates and times.

## 7.3 Returning More Than One Value

### Problem

You need to write a function that returns more than one value.

### Solution

Design your function to return a Python *tuple* and use the multiple variable assignment syntax. A tuple is a Python data structure that's a little like a list, except that tuples are enclosed in parentheses rather than brackets. They are also of fixed size.

For example, you could have a function that converts a temperature in Kelvin into both Fahrenheit and Celsius. You can arrange for this function to return the temperature in both these units by separating the multiple return values with commas:

```
>>> def calculate_temperatures(kelvin):
...     celsius = kelvin - 273
...     fahrenheit = celsius * 9 / 5 + 32
...     return (celsius, fahrenheit)
...
>>> (c, f) = calculate_temperatures(340)
>>>
>>> print(c)
67
>>> print(f)
152.6
```

When you call the function, you just provide the same number of variables before the =, and each of the return values will be assigned to the variable in the same position.

## Discussion

Sometimes, when you have just a few values to return, this is the best way to return multiple values. However, if the data is complex, you might find that a neater solution is to use Python's object-oriented features and define a class that contains the data. That way, you can return an instance of the class rather than a tuple.

## See Also

See [Recipe 7.4](#) for information on defining classes.

## 7.4 Defining a Class

### Problem

You need to group related data and functionality into a class.

### Solution

The concept of classes is central to that of object-orientation. A class is a little like a Python module (and in fact many Python modules contain classes) in that it collects together a set of functions. However, a class formalizes this structure insisting that classes are created in a certain way, and that all methods and variables relating to the class are bundled up into that class. Classes can also be arranged in a hierarchy, in which more specific classes can inherit methods from more generic classes, making it easier to write code that isn't repeated in multiple places in your program.

Define a class and provide it with the member variables you need.

The following example defines a class to represent an address book entry:

```
class Person:
    '''This class represents a person object'''

    def __init__(self, name, tel):
        self.name = name
        self.tel = tel
```

The first line inside the class definition uses triple single quotes to denote a *documentation string*, which should explain the purpose of the class.

Although entirely optional, adding a documentation string to a class allows others to see what the class does. This is particularly useful if the class is made available for others to use.

Documentation strings (or doc strings) are not like normal comments because, although they are not active lines of code, they do become associated with the class; thus, at any time, you can read the doc string for a

class using the following command (with double underscores on either side of the word `doc`):

```
Person.__doc__
```

Inside the class definition is the *constructor method*, which will be called automatically whenever you create a new instance of the class. A class is like a template, so in defining a class called `Person`, we do not create any actual `Person` objects until later:

```
def __init__(self, name, tel):  
    self.name = name  
    self.tel = tel
```

The constructor method must be named as shown, with double underscores on either side of the word `init`.

## Discussion

One way in which Python differs from most object-oriented languages is that you need to include the special variable `self` as a parameter to all the methods that you define within the class. This is a reference to, in this case, the newly created instance. The variable `self` is the same concept as the special variable `this` that you find in Java and some other languages.

The code in this method transfers parameters that were supplied to it into *member variables*. The member variables do not need to be declared in advance, but they do need to be prefixed by `self`.

So this line:

```
self.name = name
```

creates a variable called `name` that's accessible to every member of the class `Person` and initializes it with the value passed into the call to create



an instance, which looks like this:

```
p = Person("Simon", "1234567")
```

We can then check that our new `Person` object, `p`, has a name of "Simon" by typing the following:

```
>>> p.name  
Simon
```

In a complex program, it is good practice to put each class in its own file with a filename that matches the class name. This also makes it easy to convert the class into a module (see [Recipe 7.11](#)).

## See Also

See [Recipe 7.5](#) for information on defining methods.

## 7.5 Defining a Method

### Problem

You need to add some code to a class.

### Solution

Functions that are associated with a particular class are called *methods*.

The following example shows how you can include a method within a class definition:

```
class Person:  
    '''This class represents a person object'''  
  
    def __init__(self, first_name, surname, tel):
```

```
self.first_name = first_name
self.surname = surname
self.tel = tel

def full_name(self):
    return self.first_name + " " + self.surname
```

The `full_name` method concatenates the first name and surname attributes of the person, placing a space between them, and might product an output something like this:

```
Simon Monk
```

## Discussion

You can think of methods as functions that are tied to a specific class and may or may not use member variables of that class in their processing. So, as with a function, you can write whatever code you like in a method and also have one method call another.

## See Also

See [Recipe 7.4](#) for information on defining a class.

## 7.6 Inheritance

### Problem

You need a specialized version of an existing class.

### Solution

Use *inheritance* to create a subclass of an existing class and add new member variables and methods.

By default, all new classes that you create are subclasses of `object`. You can change this by specifying the class you want to use as a superclass in

parentheses after the class name in a class definition. The following example defines a class (`Employee`) as a subclass of `Person` and adds a new member variable (`salary`) and an extra method (`give_raise`):

```
class Employee(Person):  
  
    def __init__(self, first_name, surname, tel, salary):  
        super().__init__(first_name, surname, tel)  
        self.salary = salary  
  
    def give_raise(self, amount):  
        self.salary = self.salary + amount
```

Note that the preceding example is for Python 3. For Python 2, you can't use `super` the same way. Instead, you must write the following:

```
class Employee(Person):  
  
    def __init__(self, first_name, surname, tel, salary):  
        Person.__init__(self, first_name, surname, tel)  
        self.salary = salary  
  
    def give_raise(self, amount):  
        self.salary = self.salary + amount
```

## Discussion

In both of these examples, the initializer method for the subclass first uses the initializer method of the parent class (superclass) and then adds the member variable. This has the advantage of not requiring you to repeat the initialization code in the new subclass.

## See Also

See [Recipe 7.4](#) for information on defining a class.

The Python inheritance mechanism is very powerful and supports *multiple inheritance*, in which a subclass inherits from more than one superclass. For more on multiple inheritance, see the [official documentation for Python](#).

## 7.7 Writing to a File

### Problem

You need to write something to a file.

### Solution

Use the `open`, `write`, and `close` functions to open a file, write some data, and then close the file:

```
>>> f = open('test.txt', 'w')
>>> f.write('This file is not empty')
>>> f.close()
```

### Discussion

In the preceding example, the file has an extension of *txt*, implying a text file, but any file extension can be used here.

Once you have opened the file, you can make as many writes to it as you like before closing it. Note that it is important to use `close` because although each write should update the file immediately, it might be buffered in memory and data could be lost. It could also leave the file locked so that other programs can't open it.

The `open` function takes two parameters. The first is the path to the file to be written. This can be relative to the current working directory or, if it starts with a `/`, an absolute path.

The second (optional) parameter is the mode in which the file should be opened. If this is omitted, then read-only (`r`) mode is assumed. To overwrite an existing file or create the file with the name specified if it doesn't already exist, use `w`. [Table 7-1](#) shows the full list of file mode characters. You can combine these using `+`. For example, to open a file in read and binary mode, you would use this:

```
>>> f = open('test.txt', 'r+b')
```

*Table 7-1. File modes*

Mode	Description
r	Read
w	Write
a	Append to the end of an existing file rather than overwrite it
b	Binary mode
t	Text mode (default)
+	A shortcut for r+w

Binary mode allows you to read or write binary streams of data, such as images, rather than text.

## See Also

To read the contents of a file, see [Recipe 7.8](#).

For more information on handling exceptions, see [Recipe 7.10](#).

## 7.8 Reading from a File

### Problem

You need to read the contents of a file into a string variable.

### Solution

To read a file's contents, you need to use the file methods `open`, `read`, and `close`. The following example reads the entire contents of the file and assigns them to the variable `s`:

```
f = open('test.txt')
s = f.read()
f.close()
```

## Discussion

You can also read text files one line at a time using the method `readline`.

The preceding example will throw an exception if the file doesn't exist or is not readable for some other reason. You can handle this by enclosing the code in a `try/except` construction, like so:

```
try:
    f = open('test.txt')
    s = f.read()
    f.close()
except IOError:
    print("Cannot open the file")
```

## See Also

To write things to a file, and for a list of file open modes, see [Recipe 7.7](#).

For more information on handling exceptions, see [Recipe 7.10](#).

To parse JSON data, see [Recipe 7.20](#).

## 7.9 Using Pickling to Save and Load Data in a File

### Problem

You want to save the entire contents of a data structure to a file so that it can be read the next time the program is run.

### Solution

Use the Python *pickling* feature to dump the data structure to file in a format that can be automatically read back into memory as an equivalent data structure later on.

The following example saves a complex list structure to a file called *mylist.pickle*:

```
>>> import pickle
>>> mylist = ['some text', 123, [4, 5, True]]
>>> f = open('mylist.pickle', 'wb')
>>> pickle.dump(mylist, f)
>>> f.close()
```

To *unpickle* the contents of the file into a new list, use the following:

```
>>> f = open('mylist.pickle', 'rb')
>>> other_array = pickle.load(f)
>>> f.close()
>>> other_array
['some text', 123, [4, 5, True]]
```

## Discussion

Pickling will work on pretty much any data structure you can throw at it. It doesn't need to be a list. The extension of the file you pickle to doesn't matter either. Using *.pickle* makes sense, but you could equally use *.txt* or *.pic*.

The file is saved in a binary format that is not human-readable; you must open the file using the `wb` (write binary) option when writing the file and the `rb` (read binary) option when reading the file.

## See Also

To write things to a file and for a list of file open modes, see [Recipe 7.7](#).

An alternative to pickling is to save your objects as JSON files as described in [Recipe 7.21](#).

## 7.10 Handling Exceptions

### Problem

If something goes wrong while a program is running, you want to catch the error or exception and display a user-friendly error message.

### Solution

Use Python's `try/except` construct.

The following example, from [Recipe 7.8](#), catches any problems when opening a file:

```
try:
    f = open('test.txt')
    s = f.read()
    f.close()
except IOError:
    print("Cannot open the file")
```

Since you wrapped the potentially error-prone commands to open the file in a `try/except` construction, any error that occurs will be captured before it displays an error message, allowing you to handle it in your own way. Here, this means displaying the friendly message “Cannot open the file.”

### Discussion

A common situation in which runtime exceptions can occur, in addition to during file access, is when you are accessing a list and the index you are using is outside the bounds of the list. For example, this happens if you try to access the fifth (index 4) element of a three-element list:

```
>>> list = [1, 2, 3]
>>> list[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```



Errors and exceptions are arranged in a hierarchy, and you can be as specific or general as you like when catching the exceptions.

The class `Exception` is pretty near the top of that tree (most general) and will catch almost any exception. You can also have separate `except` sections for catching different types of exceptions and handling each in a different way. If you do not specify any exception class, all exceptions will be caught by the `except` command.

Python also allows you to have `else` and `finally` clauses in your error handling:

```
list = [1, 2, 3]
try:
    list[8]
except:
    print("out of range")
else:
    print("in range")
finally:
    print("always do this")
```

The `else` clause will be run if there is no exception, and the `finally` clause will be run whether there is an exception or not.

Whenever an exception occurs, you can get more information about it by using the `Exception` object, which is available only if you use the `as` keyword to put it in a variable, as shown in the following example:

```
>>> list = [1, 2, 3]
>>> try:
...     list[8]
... except Exception as e:
...     print("out of range")
...     print(e)
...
out of range
list index out of range
>>>
```

This enables you to handle the error in your own way, while keeping hold of the original error message.

## See Also

See the [official Python documentation](#) for Python exception class hierarchy.

## 7.11 Using Modules

### Problem

You want to use a Python module in your program.

### Solution

Use the `import` command:

```
import random
```

### Discussion

A large number of modules (sometimes called *libraries*) are available for Python. Many are included with Python as part of the standard library, and others can be downloaded and installed into Python. Standard Python libraries include modules for random numbers, database access, various internet protocols, object serialization, and many other functions.

One consequence of having so many modules is the potential for conflict—for example, if two modules have a function of the same name. To avoid such conflicts, specify how much of the module is accessible when importing a module.

So if you just use a command like this:

```
import random
```

there is no possibility of a conflict because you will only be able to access functions or variables in the module by prefixing them with `random` (e.g., `random.randint`). Incidentally, you'll be meeting the `random` package in the next recipe.

If, on the other hand, you use the command in the following example, every function or variable in the module will be accessible without your having to add anything in front of it; unless you know what all the functions are in all the modules you are using, there is a much greater chance of conflict:

```
from random import *
```

In between these two extremes, you can explicitly specify the components of a module that you need within a program so that they can be conveniently used without any prefix.

For example:

```
>>> from random import randint
>>> print(randint(1, 6))
2
>>>
```

Another option is to use the `as` keyword to provide a more convenient or meaningful name for the module when referencing it:

```
>>> import random as R
>>> R.randint(1, 6)
```

## See Also

The Python Standard Library includes a definitive list of all the [Python modules](#).

## 7.12 Generating Random Numbers

## Problem

You need to generate a random number within a range of numbers.

## Solution

Use the `random` library:

```
>>> import random
>>> random.randint(1, 6)
2
>>> random.randint(1, 6)
6
>>> random.randint(1, 6)
5
```

The number generated will be between the two arguments (inclusive)—in this case, simulating a gaming die.

## Discussion

The numbers generated are not truly random but are what is known as a *pseudorandom number sequence*; that is, they are a long sequence of numbers that, when taken in a large enough quantity, show what statisticians call a *random distribution*. For games, this is perfectly good enough, but if you were generating lottery numbers, you would need to look at special randomizing hardware. Computers are just plain bad at being random; it's not really in their nature.

A common use of random numbers is to select something at random from a list. You can do this by generating an index position and using that, but there is also a command in the `random` module specifically for this. Try the following example:

```
>>> import random
>>> random.choice(['a', 'b', 'c'])
'a'
>>> random.choice(['a', 'b', 'c'])
'b'
```

```
>>> random.choice(['a', 'b', 'c'])  
'a'
```

When making random selections like this, it's not uncommon to prevent choices from repeating. For example, if you have already chosen 'a' at random, it shouldn't be chosen again.

One way to do this is to take a copy of your list and then, whenever you have selected an item from it, remove that item so that it can't be chosen again. Here's how you could do that in a small program that you can find with the book downloads (see [Recipe 3.22](#)); this program is *ch\_07\_random.py*:

```
import random  
from copy import copy  
  
list = ['a', 'b', 'c']  
  
working_list = copy(list)  
while len(working_list) > 0 :  
    x = random.choice(working_list)  
    print(x)  
    working_list.remove(x)
```

Run the program, and it will display the list items, selected at random, just once:

```
$ python3 ch_07_random.py  
b  
c  
a
```

The order is likely to be different each time you run the program.

## See Also

See [the official reference for the random package](#) for more information on this.

## 7.13 Making Web Requests from Python

### Problem

You need to read the contents of a web page into a string using Python.

### Solution

Python has an extensive library for making HTTP requests called `urllib` (URL Library).

The following Python 3 example reads the contents of the Google home page into the string `contents`:

```
import urllib.request
contents =
urllib.request.urlopen("https://www.google.com/").read()
print(contents)
```

### Discussion

Having read the HTML, you are then likely to want to search it and extract the parts of the text that you really want. For this, you will need to use the string manipulation functions (see Recipes [5.16](#) and [5.17](#)).

### See Also

For more internet-related examples using Python, see [Chapter 17](#).

When the web request returns JSON data, you can parse it using [Recipe 7.20](#).

## 7.14 Specifying Command-Line Arguments in Python

### Problem

You want to run a Python program from the command line and pass it parameters.

Rather than just run a Python program, you want to supply some extra parameters to the program that the program can then use. For example:

```
$ python3 ch_07_cmdline.py a b c
```

## Solution

Import `sys` and use its `argv` variable, as shown in the following example. This returns a list, the first element of which is the name of the program. The other elements are any parameters (separated by spaces) that were typed on the command line after the program name.

The code for this example and for the other examples in this book can be downloaded (see [Recipe 3.22](#)); the program is called `ch_07_cmdline.py`:

```
import sys

for (i, value) in enumerate(sys.argv):
    print(f"arg: {i} {value}")
```

Running the program from the command line, with some parameters after it, results in the following output:

```
$ python3 ch_07_cmdline.py a b c
arg: 0 cmd_line.py
arg: 1 a
arg: 2 b
arg: 3 c
```

## Discussion

Being able to specify command-line arguments can be useful for automating the running of Python programs, either at startup ([Recipe 3.23](#)) or on a timed basis ([Recipe 3.25](#)).

## See Also

For basic information on running Python from the command line, see [Recipe 5.6](#).

To print out `argv`, we used list enumerations ([Recipe 6.8](#)).

Visit the [Python documentation](#) for an alternative and more advanced way to use command-line arguments.

# 7.15 Running Linux Commands from Python

## Problem

You want to run a Linux command or program from your Python program.

## Solution

Use the `system` command.

For example, to delete a file called *myfile.txt* in the directory from which you started Python, you could do the following:

```
import os
os.system("rm myfile.txt")
```

## Discussion

Sometimes rather than just execute a command blindly, as in the preceding example, you need to capture the response of the command. Let's say you wanted to use the `hostname` command to find the IP address (see [Recipe 2.2](#)) of the Raspberry Pi. In this case, you can use the `check_output` function in the `subprocess` library:

```
import subprocess
ip = subprocess.check_output(['hostname', '-I'])
```



The variable `ip` will contain the IP address of the Raspberry Pi. Unlike `system`, `check_output` requires the command itself and any parameters to be supplied as separate elements of a list.

## See Also

For documentation on the `OS` library, see <https://oreil.ly/1LL8G>.

For more information on the `subprocess` library, see <https://oreil.ly/HVBq->.

In [Recipe 15.7](#), you'll find an example that uses `subprocess` to display the IP address, hostname, and time of your Raspberry Pi on an ePaper display.

## 7.16 Sending Email from Python

### Problem

You want to send an email message from a Python program.

### Solution

Python has a library for the Simple Mail Transfer Protocol (SMTP) that you can use to send emails:

#### Passwords in Code

Be very careful putting usernames and passwords into your code, especially if the code is part of a project that you are uploading to the internet. It is all too easy to forget and upload your password to somewhere like GitHub.

The example that follows is for Google's Gmail. Google has a concept of application-specific passwords, which makes this kind of access more

secure by insisting on a long, random password that is different from your normal password. To get this password, you have to first log in to Google normally from a browser, go to <https://myaccount.google.com/>, and then click on Security in the lefthand navigation. In the “Signing in to Google” section, select the option called App Passwords (Figure 7-1). Note that your Google account must have two factor authentication enabled to access this option.



In the Select App drop-down list, choose “email.” In the Select Device drop-down list, choose “Other” and give the device (your Raspberry Pi) a name (such as “Raspberry Pi Python”) so that you will remember the purpose of the App Password. When you click the Generate button, a password will be generated for you (Figure 7-2).

You will need to copy this password and paste it into the Python program listed as follows (*ch\_07\_gmail.py*):

```
import smtplib

GMAIL_USER = 'your email address'
GMAIL_PASS = 'your password'
```

```
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = 587

def send_email(recipient, subject, text):
    smtpserver = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(GMAIL_USER, GMAIL_PASS)
    header = 'To:' + recipient + '\n' + 'From: ' + GMAIL_USER
    header = header + '\n' + 'Subject:' + subject + '\n'
    msg = header + '\n' + text + ' \n\n'
    smtpserver.sendmail(GMAIL_USER, recipient, msg)
    smtpserver.close()

send_email('destination email address', 'subject', 'message')
```

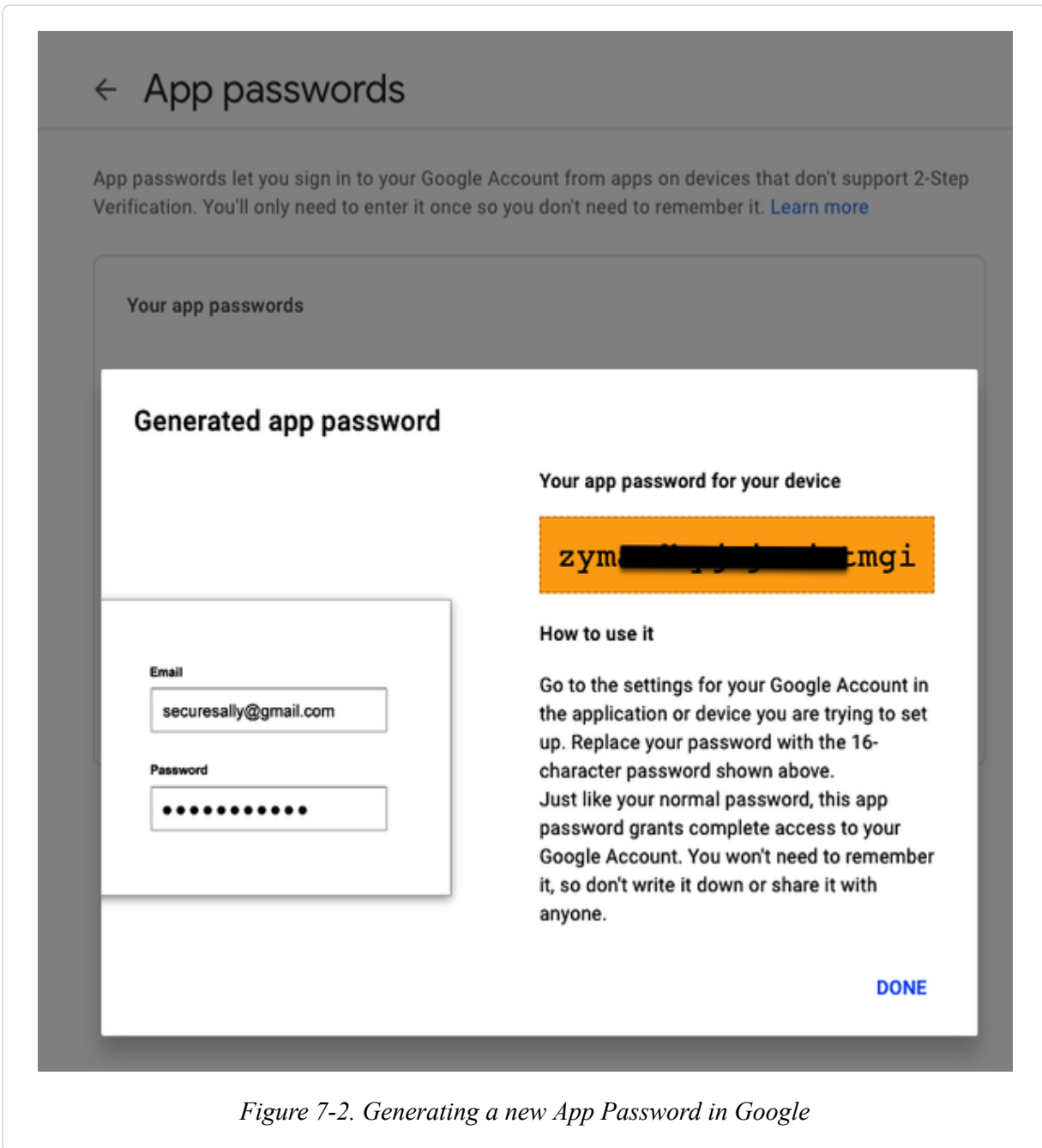


Figure 7-2. Generating a new App Password in Google

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

To use this example to send an email to an address of your choice, first change the variables `GMAIL_USER` and `GMAIL_PASS` to match your email credentials. For Gmail, the password should be the application-specific password you just generated.

If you are not using Gmail, you will also need to change the value of `SMTP_SERVER`, and possibly of `SMTP_PORT` as well, to match those values for your email provider.

You also need to change the destination email address in the last line, and you can change the subject and message here if desired.

## Discussion

The `send_email` method simplifies the use of the `smtplib` library into a single function that you can reuse in your projects.

Being able to send emails from Python opens up all sorts of project opportunities. For example, you could use a device such as a passive infrared (PIR) sensor to send an email when movement is detected.

## See Also

For a similar example that uses the IFTTT web service to send emails, see [Recipe 17.4](#).

To perform HTTP requests from the Raspberry Pi, see [Recipe 7.13](#).

Find more information on the `smtplib` at [Python.org](#).

Google Support has more information on [Google App Passwords](#).

For many more internet-related recipes, see [Chapter 17](#).

## 7.17 Writing a Simple Web Server in Python

### Problem

You need to create a simple Python web server, but you don't want to have to run a full web server stack.

### Solution

Use the `bottle` Python library to run a pure Python web server that will respond to HTTP requests.

To install `bottle`, use the following command:

```
$ sudo pip3 install bottle
```

The following Python program (called `ch_07_bottle_test.py`) simply serves up a message displaying what time the Raspberry Pi thinks it is. As with all the program examples in this book, you can also download it (see [Recipe 3.22](#)):

```
from bottle import route, run, template
from datetime import datetime

@route('/')
def index(name='time'):
    dt = datetime.now()
    time = "{:%Y-%m-%d %H:%M:%S}".format(dt)
    return template('<b>Pi thinks the date/time is: {{t}}</b>',
t=time)

run(host='0.0.0.0', port=80)
```

To start the program, you need to run it with superuser privileges:

```
$ sudo python3 ch_07_bottle_test.py
```

[Figure 7-3](#) shows the page you see if you connect to the Raspberry Pi from a browser anywhere on your network.

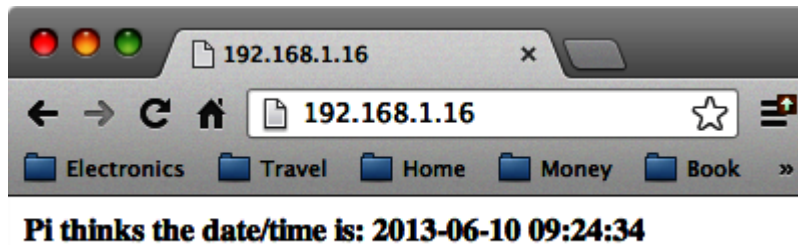


Figure 7-3. Browsing to a Python *bottle* web server

This example requires a little explanation.

After the `import` commands, the `@route` command links the URL path / with the handler function that follows it.

That handler function formats the date and time and then returns a string of HTML to be rendered by the browser. In this case, it uses a template into which values can be substituted.

The final `run` line actually starts the web serving process. Port 80 is the default port for web serving; if you want to use a different port, add a `:` followed by the port number after the server address.

## Discussion

You can define as many routes and handlers as you like within the program. `bottle` is perfect for small, simple web server projects, and because it's written in Python, it's very easy to write a handler function to control hardware in response to the user interacting with the page in a browser. You will find other examples using `bottle` in [Chapter 17](#).

The Raspberry Pi (especially a Raspberry Pi 4) is perfectly capable of running a full web server stack (web server, web framework, and database), a popular example being Apache, PHP, and MySQL. This will never perform as well as *proper* server hardware, but it can be a great playground for learning how these things work.

## See Also

To set up a Raspberry Pi as a LAMP (Linux, Apache, MySQL, and PHP), see <https://oreil.ly/M1E00>.

For more information, see the [bottle documentation](#).

For more on formatting dates and times in Python, see [Recipe 7.2](#).

For a whole load of internet-related recipes, see [Chapter 17](#).

## 7.18 Doing Nothing in Python

### Problem

You want Python to kill time for a while. You might want to do this, for example, to create a delay between sending messages to the Terminal.

### Solution

Use the `sleep` function in the `time` library as illustrated in the following code example, `ch_07_sleep_test.py`:

```
import time

x = 0
while True:
    print(x)
    time.sleep(1)
    x += 1
```

You can find the code for this example, as well as the other code examples in this recipe, with the code downloads for the book (see [Recipe 3.22](#)).

The main loop of the program will delay for one second before printing the next number.

### Discussion

The function `time.sleep` takes a value representing seconds as its parameter. However, if you want shorter delays than a second, you can



specify decimals. For example, to delay for a millisecond, you would use `time.sleep(0.001)`.

It's a good idea to put a short delay in any loop that continues indefinitely, or even just continues for more than a fraction of a second, because when `sleep` is being called, the processor is freed up to allow other processes to do some work.

When you are using the GPIO pins in [Recipe 11.1](#) and many other recipes, delays are used to do things like control the timing for LEDs blinking on and off.

## See Also

For an interesting discussion of how `time.sleep` can reduce the CPU load of your Python program, see <https://oreil.ly/FgpUQ>.

## 7.19 Doing More Than One Thing at a Time

### Problem

Your Python program is busy doing one thing, and you want it to do something else at the same time.

### Solution

Use the Python `threading` library.

The following example (`ch_07_thread_test.py`) sets a thread running that will interrupt the counting of the main thread. As with all the program examples in this book, you can also download it (see [Recipe 3.22](#)):

```
import threading, time, random

def annoy(message):
    while True:
        time.sleep(random.randint(1, 3))
        print(message)
```

```
t = threading.Thread(target=annoy, args=('BOO !!',))
t.start()

x = 0
while True:
    print(x)
    x += 1
    time.sleep(1)
```

The output on the console will look something like this:

```
$ python3 ch_07_thread_test.py
0
1
BOO !!
2
BOO !!
3
4
5
BOO !!
6
7
8
```

When you start a new *thread of execution* using the Python threading library, you must specify a (`target`) function that is to be run as that thread. In this example, the function, called `annoy`, contains a loop that will continue indefinitely printing out a message after a random interval of between 1 and 3 seconds. Note that the `args` parameter is used to pass a string to `annoy`.

To start the thread actually running, the `start` method on the `Thread` class is called. This method has two parameters: the first is the name of the function to run (in this case, `annoy`), and the second is a tuple that contains any parameters that are to be passed to the function (in this case, `'BOO !!'`).

You can see that the main thread, which is just happily counting, will be interrupted every few seconds by the thread running in the `annoy` function.

## Discussion

Threads like these are also sometimes called *lightweight processes* because they are similar in effect to having more than one program or process running at the same time. They do, however, have the advantage that threads running in the same program have access to the same variables, and when the main thread of the program exits, so do any threads that are started in it.

## See Also

For a good introduction to threading in Python, see <https://pymotw.com/3/threading>.

## 7.20 Parsing JSON Data

### Problem

You want to parse data in the popular JSON (JavaScript Object Notation) data structuring language.

This might be because you are downloading data from a web service or have data saved in a JSON file.

### Solution

Use the `json` package, as shown in the following example:

```
import json

s = '{"books" : [
    {"title" : "Programming Arduino", "price" : 10.95},
    {"title" : "Pi Cookbook", "price" : 19.95}
]}'

j = json.loads(s)
print(j['books'][1]['title'])
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)). The file is called `ch_07_parse_json.py`.

I have split the JSON string onto multiple lines in the previous example to make it easier to see the structure of the data.

The `loads` (load string) function parses the string into a data structure stored in the variable `j`. You can then access the contents of the structure as if it were a combination of Python lists and tables. In this case, the `title` of element 1 of the `books` list is printed (`Pi Cookbook`).

## Discussion

If you want to parse the content of a file containing JSON data, you could use [Recipe 7.8](#) to read the file into a string and then use the method just shown. However, it is more efficient, especially for large files, to use `json.load` (note that it's `load`, not `loads`) directly on the file.

For example, you could create a file called `ch_07_example_file.json` that contains the following JSON:

```
{ "books" : [
    { "title" : "Programming Arduino", "price" : 10.95 },
    { "title" : "Pi Cookbook", "price" : 19.95 }
]}
```

The following code would read the file and parse it, producing the same result as the first example in this recipe, but the code fetches its JSON from a file (you can find this example in the file `ch_07_parse_json_file.py`):

```
import json

file_name = 'ch_07_example_file.json'
json_file = open(file_name)

j = json.load(json_file)
json_file.close()

print(j['books'][1]['title'])
```

The final example in this recipe deals with parsing data from a web request. Most web service APIs have a JSON interface. The following example uses the weatherstack.com (formerly known as Apixu) weather service. To use this service, you will need to **sign up for an account** (a free one will do):

```
import json
import urllib.request

key = 'paste_your_key_here'

response =
urllib.request.urlopen('http://api.weatherstack.com/current?
    access_key=' + key + '&query=Paris')
j = json.load(response)

print(j['current']['weather_descriptions'][0])
```

Before running `ch_07_parse_json_url.py`, remember to change the value of `key` to your key. You may also want to change the location from “Paris” to your location.

When you run the program, you should see something like this:

```
$ python3 ch_07_parse_json_url.py
Partly cloudy
```

The API actually returns a lot of data. You can see it all if you change the program to include a final line `print(j)`.

You can then change how you navigate into the data to get the information you want.

## See Also

For reading and writing files, see Recipes [7.7](#) and [7.8](#).

## 7.21 Saving Dictionaries as JSON Files

## Problem

You have a dictionary that you want to save as a text file in JSON format.

## Solution

Use the `dump` function in the `json` package to write a dictionary or other object to a file.

The example in `ch_07_json_dump.py` is:

```
import json

phone_numbers = {'Simon': '01234 567899', 'Jane': '01234 666666'}

f = open('test.txt', 'w')
json.dump(phone_numbers, f)
f.close()
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

## Discussion

The `dump` function will also work on lists and any combination of lists and dictionaries or other objects that you might want to save like this.

When it comes to reconstituting the text in the file, you can do so using `json.load` as described in [Recipe 7.20](#).

Saving objects as JSON files rather than using pickling ([Recipe 7.9](#)) has the advantage that the files can be read and edited in a text editor, something that is not possible with pickling.

## See Also

For information on pickling, see [Recipe 7.9](#).

## 7.22 Creating User Interfaces

### Problem

You want to easily create a graphical user interface (GUI) for your Python app.

### Solution

Use `guizero`. Laura Sach and Martin O'Hanlon at the Raspberry Pi Foundation have created a Python library that makes it super easy to design GUIs for your projects.

Originally designed for the Raspberry Pi, `guizero` is also perfectly happy on most environments that run Python, so you can use it on your PC or Mac as well as on your Raspberry Pi. To install `guizero`, run the following command from the Terminal:

```
$ sudo pip3 install guizero
```

Once installation is complete, you can try out `guizero` using the example program `ch_07_guizero.py` that is included with the book downloads ([Recipe 3.22](#)):

```
from guizero import *

def say_hello():
    info("An Alert", "Please don't press this button again")

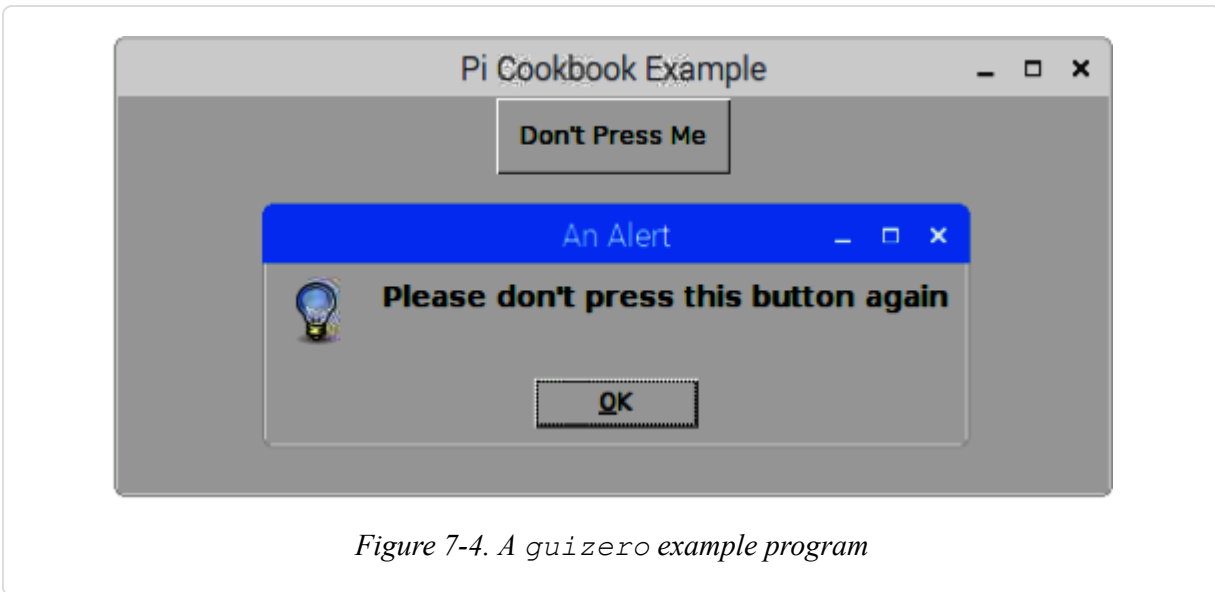
app = App(title="Pi Cookbook Example", height=200)
button = PushButton(app, text="Don't Press Me",
                    command=say_hello)

app.display()
```

When you run the program using the following command, a window with a button on it opens on your screen. If you click on the button, an alert

message appears (Figure 7-4):

```
$ python3 ch_07_guizero.py
```



*Figure 7-4. A guizero example program*

This example shows how easy it is to hook a Python function up to a button so that when the button is clicked, the function is run.

The function (`say_hello`) is defined first in the program. Then a new variable, `app`, is defined and initialized to be an instance of the class `App`, with some parameters that specify a title to appear at the top of the window and the window's height in pixels. Both parameters are optional, and many other available options are defined in the [documentation for guizero](#).

This `app` variable is then supplied as the first of the parameters to the `PushButton` that is created on the next line. The button uses the `command` parameter to specify the function to be run when the button is clicked. Note that when you specify the function to run, you do not put `()` after it because you are referring to the function, not calling it.

## Discussion

This is an introductory example of `guizero`, just to get you started. The library is by no means just limited to buttons on a screen. The main goal of



the library is to allow you to create simple user interfaces with a minimal amount of programming. When you want to start making things a bit fancier, you can delve into various ways of laying out the widgets (buttons, checkboxes, sliders, etc.) in your window and changing font sizes and colors. However, as always, start by keeping it simple.

## See Also

For full information on `guizero`, see the excellent documentation at this [guizero GitHub site](#).

`guizero` is also used in Recipes [11.9](#), [11.10](#), and [11.11](#).

## 7.23 Using Regular Expressions to Search for Patterns in Text

### Problem

You want to do a complex search, looking for something in a piece of text.

### Solution

Use Python's regular expression (regex) feature. Regular expressions have been around since the early days of computer science, when computer science was a branch of mathematics and benefited from the rigor of the mathematician's mind.

A regular expression is a way of describing a pattern that occurs in some text. This is similar to [Recipe 5.16](#). However, with regular expressions you can find more flexible wildcard matches, as shown here:

```
import re

text = "looking forward to finding the word for"
x = re.search("(^|\s)for($|\s)", text)

print(x.span())
```

As with all the program examples in this book, you can also download this program, called `ch_07_regex_find.py` (see [Recipe 3.22](#)).

If you run this program, you will get the following output:

```
$ python3 ch_07_regex_find.py
(35, 39)
```

This indicates that the word *for* has been found at character position 35 (actually the space before *for*) in the string. The second value is the end position index. Notice that the program has ignored the word *forward*. Let's take a look at how this code works.

First, we need to import the `re` (regular expression) module. Next, we add the variable `text` that contains the test string that we are going to search within.

We then use the `search` function to find what we want in the string. The first parameter is the regular expression, and the second is the string to search. In this case, the regular expression is the following string:

```
"(^|\s)for($|\s)"
```

Right in the middle of the regular expression is the word *for*. That's to be expected, because that's the word we're looking for. To either side of *for* are expressions in parentheses. Before it we have this:

```
(^|\s)
```

The three magic symbols are `^`, which means the start of the string; `|`, which means *or*; and `\s`, which means any whitespace character (space or tab). So you can read this section as matching either the start of the string or some whitespace, before trying to match *for*. That is, *for* must either come at the start of the string or be preceded by a space or some other whitespace

character. This ensures that the regular expression does not match words that end in *for*.

There is a similar expression after *for* that must also match:

```
(\s|$)
```

Here, the new special symbol is \$, which indicates the end of the string. In other words, after the letters *for*, a match will occur only if we are at the end of the string or there is a space or other whitespace character.

**Table 7-2** shows some of the most common regular expression symbols. You can find a complete list on this [W3Schools.com page](#).

*Table 7-2. Common regular expression symbols*

Special symbol	Meaning
.	Matches any single character.
^	Matches the beginning of the string.
\$	Matches the end of the string.
\d	Any digit.
\s	Whitespace.
\w	Alphanumeric (digits and uppercase and lowercase letters).
*	Zero or more occurrences of whatever follows it. For example, *\d will match a string of zero or more digits.
+	One or more occurrences of whatever follows it.
[ ]	Will match any of the characters contained between the brackets. You can also do ranges, such as [a-d], which will match any of the characters a to d.

The best way to become familiar with regular expressions is to play with an online regular expression tester.

## Discussion

It can be tricky tuning a regular expression so that it works just right. An [online regular expression tool](#) (Figure 7-5) can be a big help in learning how to properly structure and test a regular expression.

The online tester has an area where you can write your regular expression, and a test string area, where you can put the text that you want to use with your regular expression. The tool then highlights what has matched. In Figure 7-5, the tool has correctly highlighted the word *for*.

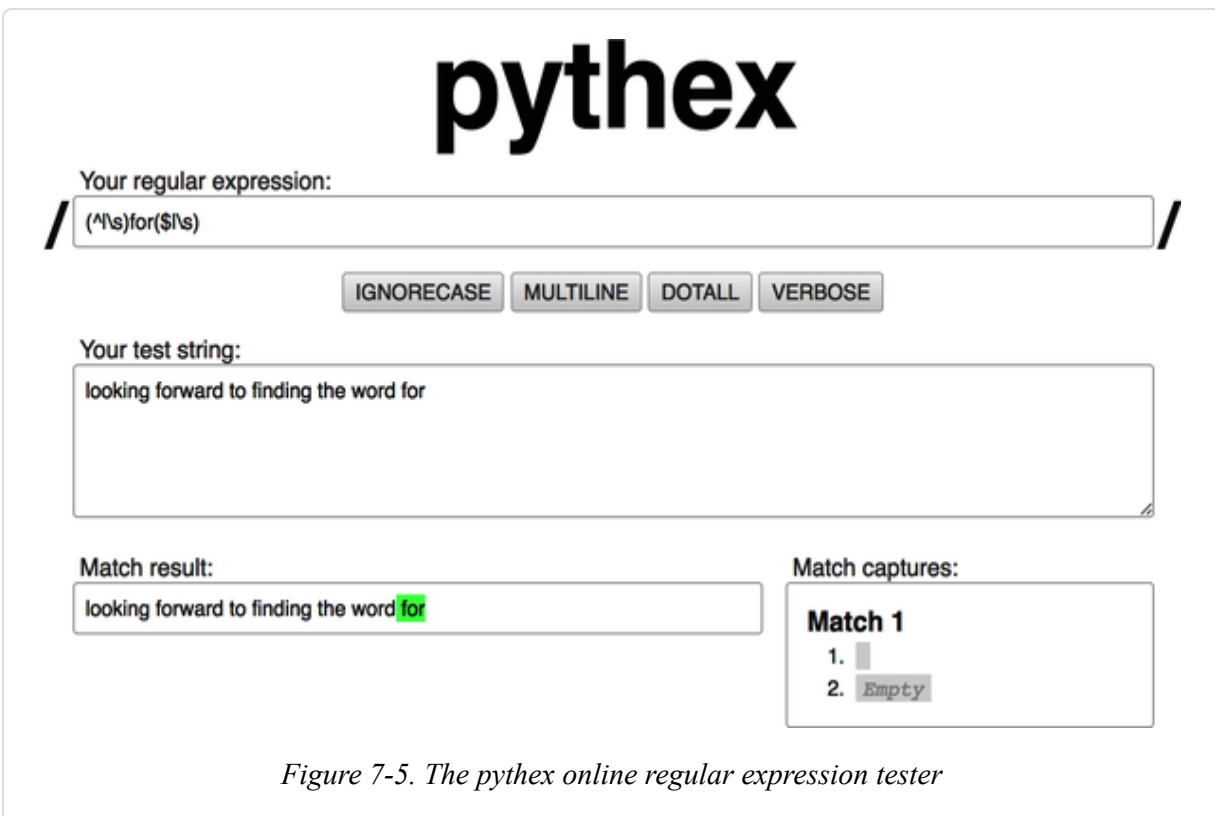


Figure 7-5. The pythex online regular expression tester

## See Also

To replace the text you have matched, see [Recipe 7.24](#).

For more details about regular expressions in Python, see this [W3Schools.com page](#).

## 7.24 Using Regular Expressions to Validate Data Entry

## Problem

You have some text that you want to validate; for example, you want to make sure that the text looks like an email address.

## Solution

Use a regular expression ([Recipe 7.23](#)).

Regular expressions are mainly used to validate information entered by a user. For example, if you have ever completed an online form that includes your email address, and you typed in something that didn't look like an email address, the message you received saying that the address wasn't a valid format probably came from a regular expression validation.

Try out the code in the file *ch\_07\_regex\_email.py* (all program examples in this book are available for download; see [Recipe 3.22](#)):

```
import re

regex = '^[\w_\.\+-]+@[\w_\.\-]+\.[\w_-]+$'
while True:
    text = input("Enter an email address: ")
    if re.search(regex, text):
        print("valid")
    else:
        print("invalid")
```

This program will repeatedly prompt you to enter an email address and then report whether it is valid. An online search will uncover alternative regular expressions for email and for pretty much any other type of validation.

This one looks for one or more alphanumeric (plus `_`, `+` or `-`), followed by an `@` symbol, followed by a repeat of that sequence, followed by the sequence again but without a period in the string, which makes sure that the email doesn't end in a period.

## Discussion

If you have a particular validation in mind (for example, a phone number or website), someone will almost certainly have made a regular expression for it. So before writing your own, do an internet search. There is no point in reinventing the wheel.

## See Also

For the basics on regular expressions, see [Recipe 7.23](#).

## 7.25 Using Regular Expressions for Web Scraping

### Problem

You want to write a Python program that automatically fetches (scrapes) information from a web page.

### Solution

Use regular expressions to match text in the page's contents, which are in HTML format.

Regular expressions are very useful for *web scraping*. Web scraping means automatically reading things from a web page's HTML. For example, if I want a Python program to automatically give me the current Amazon ranking of this book, I need to be able to grab the number from the Amazon sales rank (circled in [Figure 7-6](#)).

## Product details

**Paperback:** 400 pages

**Publisher:** O'Reilly Media; 3 edition (November 4, 2019)

**Language:** English

**ISBN-10:** 1492043222

**ISBN-13:** 978-1492043225

**Product Dimensions:** 7 x 9.2 inches

**Shipping Weight:** 1.9 pounds (View shipping rates and policies)

**Average Customer Review:** Be the first to review this item

**Amazon Best Sellers Rank:** #746,779 in Books (See Top 100 in Books)

#81 in **Electronic Sensors**

#154 in **Computer Hardware Peripherals (Books)**

#339 in **Single Board Computers (Books)**

*Figure 7-6. Web scraping from Amazon*

If I click View Source in my browser and then search for “Sellers Rank,” I can find the relevant piece of HTML, which looks like this:

```
<li id="SalesRank">
<b>Amazon Best Sellers Rank:</b>
#746,779 in Books (<a href="https://www.amazon.com/best-sellers-
books-Aazon
/zgbs/books/ref=pd_dp_ts_books_1">See Top 100 in Books</a>)
```

I can use this as my test text in an online regular expression tester and work on an expression that will extract the Amazon rank. We can assume that this will be everything between # and *in Books*.

Here is the code for this, which you can also find in the downloads for the book ([Recipe 3.22](#)) in the file `ch_07_regex_scraping.py`:

```
import re
import urllib.request

regex = '#([\d,]+) in Books'
```

```
url = 'https://www.amazon.com/Raspberry-Pi-Cookbook-Software-
Solutions/
      dp/1492043222/'

print("The Amazon rank is.....")
text = urllib.request.urlopen(url).read().decode('utf-8')
print(re.search(regex, text).group())
```

The output of the file will look something like this:

```
$ python3 test.py
The Amazon rank is.....
#746,779 in Books
```

The code first reads the web page contents. The text must then be converted to UTF-8 format (Latin alphabet only) before it can be used with the `re` regular expression module.

## Discussion

Many websites offer APIs (see [Recipe 7.20](#)). If the information you're trying to scrape is available through an API, then that is a much better way of getting it—not least because web scraping is very dependent on the appearance and wording of the page, which means that if the site is revamped, you'll probably need to come up with a new regular expression.

## See Also

To read the contents of a web page, see [Recipe 7.13](#).

For the basics on regular expressions, see [Recipe 7.23](#).



# Chapter 8. Computer Vision

---

## 8.0 Introduction

*Computer vision* (CV) allows your Raspberry Pi to “see” things. In practical terms, this means that your Raspberry Pi can analyze an image, look for items of interest, and even recognize faces and text.

If you connect a Raspberry Pi to a camera to supply the images, all sorts of possibilities open up. This theme is continued in [Chapter 9](#) where we take this a stage further into the realm of machine learning.

## 8.1 Installing OpenCV

### Problem

You want to install [OpenCV computer vision software](#) on your Raspberry Pi.

### Solution

To install OpenCV, first install the prerequisite packages and update the NumPy Python library using these commands:

```
$ sudo apt install libatlas-base-dev
$ pip3 install --upgrade pip
$ pip3 install imutils
$ pip3 install numpy --upgrade
```

Then install OpenCV itself using:

```
$ pip3 install opencv-python
```

After installation is complete, you can check that everything is working by starting Python 3, importing `cv2`, and checking the version:

```
$ python3
Python 3.9.2 (default, Mar 12 2021, 04:06:34)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more
information.
>>> import cv2
>>> cv2.__version__
'4.6.0'
>>>
>>> exit()
```

Note that `__version__` is the word “version” with two underscore characters on each side.

## Discussion

Computer vision is both processor and memory intensive, so although OpenCV will just about work on an older Raspberry Pi, it can be slow on anything earlier than a Raspberry Pi 2. And if you plan to try out the recipes in [Chapter 9](#), you will need at least a Pi 4 or 400.

## See Also

The first recipe in this chapter to use OpenCV is [Recipe 8.4](#). It contains useful details for getting started with OpenCV.

## 8.2 Setting Up a USB Camera for Computer Vision

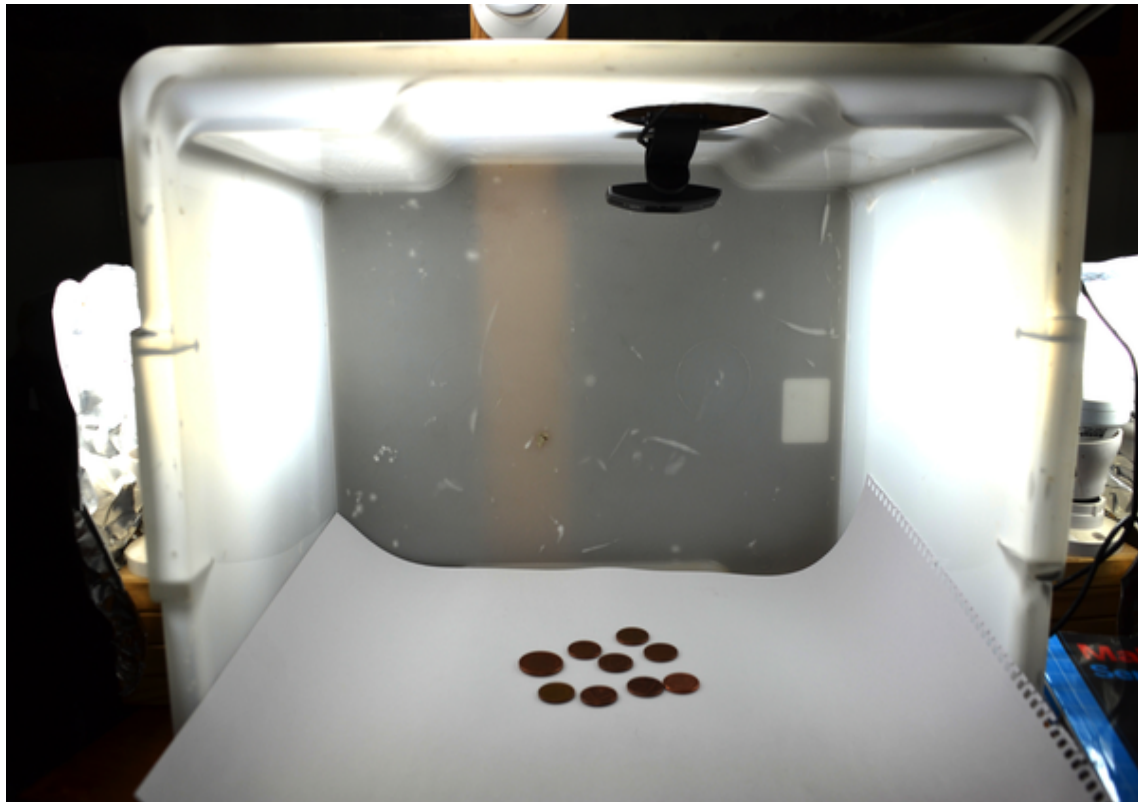
### Problem

You want to set up a USB webcam for use in computer vision (CV) projects.

## Solution

Use a **USB webcam** that is compatible with the Raspberry Pi. Choose a good-quality camera. If you are working on a project for which you need the camera close to the subject, select one that has a manual focus option. For getting really close to the subject, a low-cost USB endoscope can be useful.

Depending on your CV project, you might want to set up a well-lighted area. **Figure 8-1** shows a simple light box made from a translucent plastic storage box that is illuminated from the sides and top to give even lighting. The webcam is attached to a hole in the top of the box. This arrangement is used in **Recipe 8.4**.



*Figure 8-1. Using a homemade light box for even illumination*

You can also buy commercial *light tents*, designed for photography, that work well.

You might need a little trial and error to get your system brightly and evenly illuminated. Shadows can be particularly problematic.

## Discussion

You can test out your USB camera from the OpenCV console. Start Python 3 and then enter the following commands:

```
$ python3
Python 3.9.2 (default, Mar 12 2021, 04:06:34)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more
information.
>>> import cv2
>>> from imutils.video import VideoStream
>>> vs = VideoStream(src=0).start()
>>> img = vs.read()
>>> cv2.imshow('image',img)
>>> cv2.waitKey(0)
```

A window should open showing an image from your camera after the last line of code is entered. You may have to close the entire Terminal window to get the image window to close.

In OpenCV, even single images are just taken as frames from a video stream. Notice that on the third line you previously entered, we have `src=0`. This means the first camera that OpenCV can find. So, if you have multiple cameras, you can use a different number here.

Once the image has been read using `vs.read()`, you can use OpenCV's `imshow` utility method to display the image. You will find that you use this a lot to debug your computer vision projects.

The final `cv2.waitKey(0)` is required to allow OpenCV to actually render the image in the background until a key is pressed.

## See Also

To use a Raspberry Pi Camera Module with OpenCV, see [Recipe 8.3](#).

## 8.3 Using a Raspberry Pi Camera Module for Computer Vision

### Problem

You want to use a Raspberry Pi Camera Module that connects directly to your Raspberry Pi with OpenCV.

### Solution

The Raspberry Pi Camera Module should automatically show up as a camera device once you have followed [Recipe 1.16](#) to install it.

Having installed the camera, you can now try the following commands to make sure that it's working:

```
$ python3
Python 3.9.2 (default, Mar 12 2021, 04:06:34)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more
information.
>>> import cv2
>>> from imutils.video import VideoStream
>>> vs = VideoStream(src=0).start()
>>> img = vs.read()
>>> cv2.imshow('image',img)
>>> cv2.waitKey(0)
```

### Discussion

Note that in early versions of Raspberry Pi OS, you had to install a *driver* to make the camera module available to OpenCV; if OpenCV doesn't detect the camera module, try updating your Raspberry Pi OS to the latest version ([Recipe 3.40](#)).

### See Also

See [Recipe 1.16](#) for information on installing the Raspberry Pi Camera Module.

See <http://picamera.readthedocs.org> for information on the `picamera` Python module.

To use a USB camera with OpenCV, see [Recipe 8.2](#).

## 8.4 Counting Coins

### Problem

You want to use computer vision to count the number of coins in your webcam's view.

### Solution

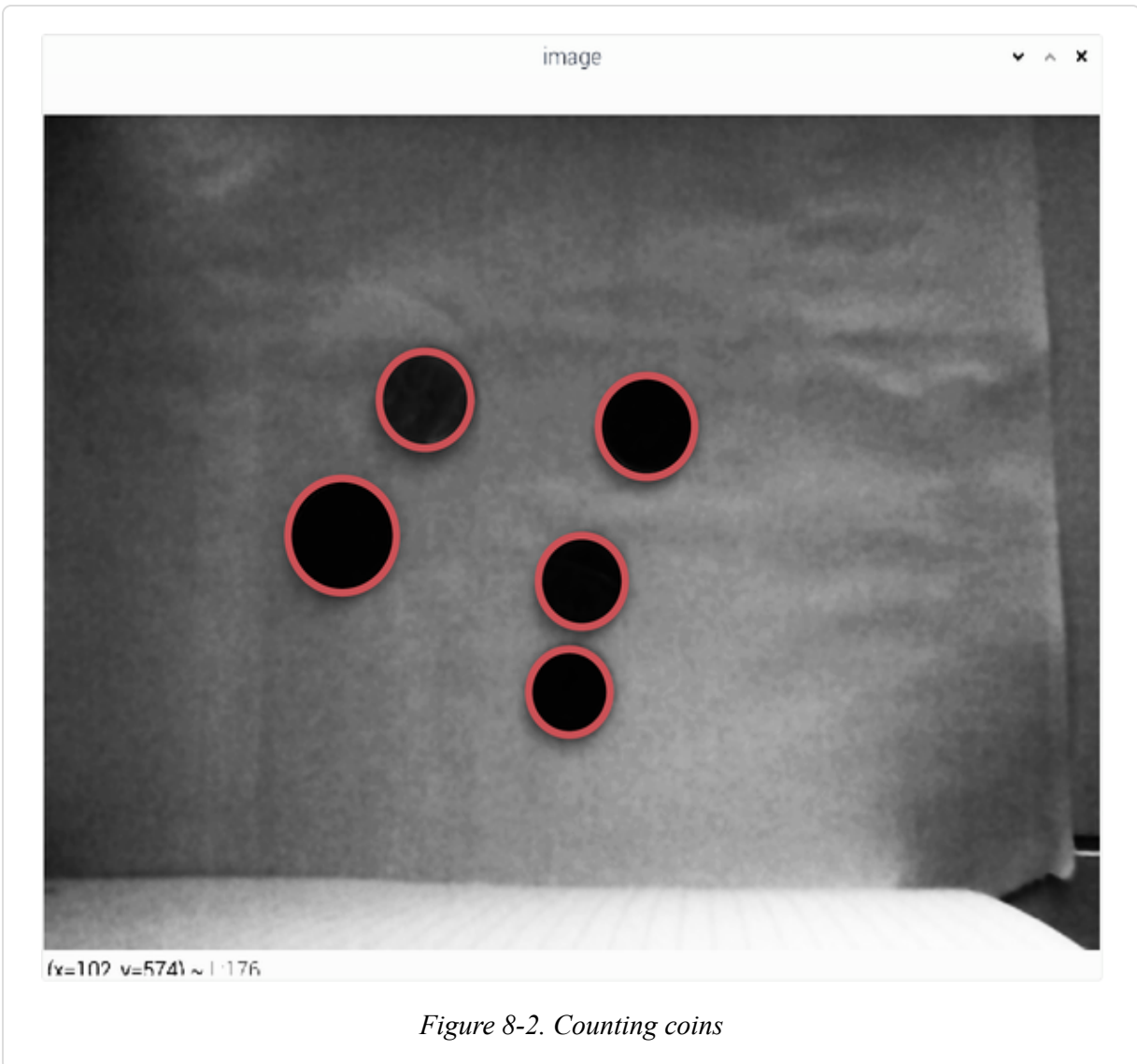
Use OpenCV's *Hough Circles* detector to provide a real-time count of the number of coins placed within view of the webcam. Hough Circles will detect any type of circles, and works well detecting most coins.

This is one use of CV for which you really need good lighting and a camera fixed in position. I used the setup shown in [Figure 8-1](#).

The critical part of many computer vision projects is getting the parameters right and this recipe is no exception. For this reason, before using the final program that just gives a count of coins, we will use a test program that draws outlines around the coins, so that we can see what's going on.

You can find the code for this example, as well as the other examples in this recipe, with the downloads for the book (see [Recipe 3.22](#)). The program is called `ch_08_coin_count_test.py`.

Put some coins under your camera and run the program. A window like that in [Figure 8-2](#) should appear.



If you are lucky, your coins will all have circles around them, and you should see output in the console like this:

```
$ python3 ch_08_coin_count_test.py
[[[380.5 338.5 37.9]
 [553.5 249.5 34.9]
 [538.5 357.5 31.4]
 [546.5 442.5 30.7]
 [418.5 244.5 33.1]]]
```

To refresh the image, press any key; when you want to exit the program, press the X key.

If your coins are not all circled, you'll need to adjust some parameters (param1, param2, minRadius, and maxRadius) in the program `ch_08_coin_count_test.py`:

```
import cv2
from imutils.video import VideoStream
from imutils import resize

vs = VideoStream(src=0).start()

while True:
    img = vs.read()
    img = resize(img, width=800)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = cv2.blur(img, (3, 3))

    detected_circles = cv2.HoughCircles(img,
        cv2.HOUGH_GRADIENT, 1, 20, param1 = 50,
        param2 = 30, minRadius = 15, maxRadius = 100)

    print(detected_circles)

    for pt in detected_circles[0]:
        a, b, r = pt[0], pt[1], pt[2]
        cv2.circle(img, (int(a), int(b)), int(r), (0, 255, 0),
2)

    cv2.imshow('image', img)
    key = cv2.waitKey(0)
    cv2.destroyAllWindows()
    if key == ord('x'):
        break

vs.stop()
```

You should not need to change the parameter `param1`. If you are interested in what it and the other parameters do, you can read about them at <https://oreil.ly/3AmKn>.

If you are getting a lot of false circles, try increasing the value of `param2`. But the most likely parameters in need of change are `minRadius` and `maxRadius`, as these will be sensitive to the resolution of your camera, its



lens focal length, and the distance to the coins. So if no coins are circled, increase the value of `maxRadius`.

Tweak the parameters until your coins are being correctly identified.

## Discussion

Here's a quick run-through of how the test program works.

Most of the code lives in a `try` block within the `while True:` block. This ensures that when you press Ctrl-C to quit the program, the video stream is stopped.

After reading the image, there are a couple of processing stages. First, the image is resized to a width of 800 pixels, then converted to grayscale, and finally a blur filter is applied. The blur filter helps improve the circle matching.

The call to `cv.HoughCircles` returns an array of the circles that OpenCV has found. The three values are the  $x$  and  $y$  coordinates of the center of the circle and the circle's radius.

To render these circles on top of the image of the coins, a `for` loop is used to iterate over each of the detected circles and then the `cv2.circle` method is used to draw a black (0, 0, 0) circle 2 pixels wide around each coin.

The actual coin counting program is just a simplification of the test program, so when you are ready, run the program `ch_08_coin_count.py`. Try moving coins in and out of the field of view and notice how the count changes. You could also add some different shaped objects among the coins and verify that they are not identified as coins.

An interesting project would be to use the radius of the coins to identify their monetary value and add up the value of the coins on the table.

## See Also

For information on installing OpenCV, see [Recipe 8.1](#)

For information on setting up a camera, see [Recipe 8.2](#).

## 8.5 Face Detection

### Problem

You want to find the coordinates of faces in a photograph or webcam image.

### Solution

Use the HAAR-like feature detection in OpenCV to analyze an image and pick out the faces. HAAR stands for High Altitude Aerial Reconnaissance, and we are using some of the features developed for that application here.

If you have not already done so, install OpenCV (see [Recipe 8.1](#)).

You can find the code for this example, as well as the other examples in this recipe, with the downloads for the book (see [Recipe 3.22](#)). The program is called *ch\_08\_faces.py*.

You will find a suitable image file for testing called *faces.jpg* in the same folder as the Python program. Run the program, and you should see output like this and an image like [Figure 8-3](#):

```
$ python3 ch_08_faces.py
[[173 139 66 66]
 [367 60 66 66]
 [564 73 66 66]]
```

Note that the *faces.jpg* file (or whatever image file you use) must be in the same directory as the Python program.



*Figure 8-3. Detecting faces*

## Discussion

Here's the listing for the program (*ch\_08\_faces.py*):

```
import cv2, pkg_resources

haar_file = pkg_resources.resource_filename('cv2',
    'data/haarcascade_frontalface_default.xml')
face_cascade = cv2.CascadeClassifier(haar_file)

img = cv2.imread('faces.jpg', cv2.IMREAD_GRAYSCALE)
```

```

scale_factor = 1.4
min_neighbors = 5

faces = face_cascade.detectMultiScale(img, scale_factor,
min_neighbors)
print(faces)

for (x,y,w,h) in faces:
    img = cv2.rectangle(img, (x,y), (x+w,y+h), (255, 255, 255),
2)

cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

The OpenCV system contains a whole load of classifiers for detecting faces and other features. These are all contained in the directory found using the `pkg_resources` utility package. The actual face detector that we are going to use in this program is contained in an XML descriptor file called *haarcascade\_frontalface\_default.xml*.

The image is read into a file as grayscale. The parameters `scale_factor` and `min_neighbors` may need tweaking if you use an image of your own rather than the test image.

`scale_factor`

This determines the step size that the face detector will use in trying to find faces by automatically changing the image scale. In this case, a value of 1.4 means the scale will change by 40% each time. Setting this to a higher number will speed up face matching but may result in some faces being missed.

`min_neighbors`

If this parameter is too low, the algorithm essentially becomes less fussy about what it considers to be a face.

Once the faces are detected, the `print` command shows the coordinates of the rectangles found, and then the `for` loop superimposes them on the image before displaying it.

There are many built-in HAAR features. You can list them all using the following command:

```
$ cd ~/.local/lib/python3.9/site-packages/cv2/data/
$ ls
haarcascade_eye.xml
haarcascade_lowerbody.xml
haarcascade_frontalcatface_extended.xml
haarcascade_profileface.xml
haarcascade_frontalcatface.xml
  haarcascade_righteye_2splits.xml
haarcascade_frontalface_alt2.xml
  haarcascade_russian_plate_number.xml
haarcascade_frontalface_alt_tree.xml      haarcascade_smile.xml
haarcascade_frontalface_alt.xml
haarcascade_upperbody.xml
haarcascade_frontalface_default.xml      __init__.py
haarcascade_fullbody.xml                 __pycache__
haarcascade_lefteye_2splits.xml
```

As you can see, they are all associated with parts of the body. You can even look for smiles!

## See Also

In [Chapter 9](#) we will return to object recognition, but by using machine learning techniques we will be able to detect all sorts of objects.

For information on installing OpenCV, see [Recipe 8.1](#).

For information on setting up a camera, see [Recipe 8.2](#).

For more information on face detection, see <https://oreil.ly/iNJ8>.

## 8.6 Motion Detection

### Problem

You want to use a camera connected to your Raspberry Pi to detect something moving in its field of view.

### Solution

Use OpenCV and NumPy to detect changes between successive frames from the camera.

The program that follows compares each captured image with the previous image. It then uses NumPy (a numeric library for Python) to calculate how different the two images are. If this measure of difference exceeds a threshold, it prints out a message saying that movement was detected.

You can find the code for this example, as well as for the other examples in this recipe, with the downloads for the book (see [Recipe 3.22](#)). The program is called *ch\_08\_detect\_motion.py*.

Attach your USB webcam or Raspberry Pi camera and then run the program. Try moving your hand in front of the field of view, and you should see a message that says “Movement detected”:

```
import cv2
import numpy as np
from imutils.video import VideoStream
from imutils import resize

diff_threshold = 1000000

vs = VideoStream(src=0).start()

def getImage():
    im = vs.read()
    im = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    im = cv2.blur(im, (20, 20))
    return im

old_image = getImage()

while True:
    new_image = getImage()
    diff = cv2.absdiff(old_image, new_image)
    diff_score = np.sum(diff)
    # print(diff_score)
    if diff_score > diff_threshold:
        print("Movement detected")
    old_image = new_image
```

If you get too many false alarms, then increase the value of `diff_threshold`. You might also want to uncomment the `line print(diff_score)`, as this will show you what sort of difference values are being detected.

The results are improved by setting the image to grayscale and applying a blur filter.

## Discussion

Successive frames of the image might look like Figures 8-4 and 8-5. When the first image is subtracted from the second, the resulting image will look like Figure 8-6.



*Figure 8-4. Movement detection, frame 1*



*Figure 8-5. Movement detection, frame 2*





*Figure 8-6. Movement detection, the difference image*

Although the code for this recipe just displays a message, there is no reason why your code shouldn't turn on a light, or perform some other action using the GPIO pins.

## **See Also**

For information on installing OpenCV, see [Recipe 8.1](#).

For information on setting up a camera, see [Recipe 8.2](#).

An alternative way to detect movement is to use a passive infrared (PIR) sensor; see [Recipe 13.9](#).

## **8.7 Extracting Text from an Image**

## Problem

You want to be able to convert an image containing text to actual text.

## Solution

Use the Tesseract optical character recognition (OCR) software to extract text from the image. To install Tesseract, run the following commands (you will probably want to copy and paste this from the file *long\_commands.txt* in the book downloads [Recipe 3.22]):

```
$ sudo apt install tesseract-ocr
$ sudo apt install libtesseract-dev
```

To try out Tesseract, you will need an image file that contains some text. You will find one called *ocr\_test.tiff* with the downloads for the book (see Recipe 3.22). To convert the image to text, run the following command:

```
$ cd ~/raspberrypi_cookbook_ed4
$ tesseract ocr_test.tiff stdout
Page 1
This is an image

of some text.
```

If you take a look at the image file *ocr\_test.tiff*, you will see that it is indeed an image containing those words.

## Discussion

Although I used a TIFF image, the `tesseract` library will work with most image types, including PDF, PNG, and JPEG files.

## See Also

For more information on the `tesseract` library, see <https://oreil.ly/Evdxw>.

# Chapter 9. Machine Learning

---

## 9.0 Introduction

You may be surprised to hear that your humble Raspberry Pi is a great platform for experimenting with machine learning. In this chapter you will experiment with recognizing objects in real-time video, recognizing sounds, and linking all this to your own Python programs.

Programming a computer involves giving the computer a list of instructions to follow. A procedure is created to accomplish the thing we want the computer to do. This is expressed in a programming language such as Python and works really well for things like storing data or making calculations. However, it's quite hard to think of how you would write a program to respond to a spoken command, or to identify objects in a photograph. The way humans and other animals learn such things is through practice. Our brains gradually learn to recognize things through experience. No program is running in our heads to do these things; we learn to do them.

Machine learning (ML) involves a normal computer running special machine learning programs (that bit is normal programming) that process large amounts of data and learn from that data in much the same way as a brain does. We can, for example, train the computer to recognize spoken commands by providing it with lots of samples of the commands, along with other examples and background noise that we want the computer to learn to ignore.

Most ML is concerned with *classification*. That is, taking some input and putting it into a category. For example, this could be taking data about a sound sample and assigning probabilities as to which of a set of predefined words or phrases that sample contains. Or, it might be classifying objects in an image to decide which animal a picture represents or whether that animal is present in a video stream containing lots of things.

There are various mechanisms for this learning, some of which rely on conventional statistics and others that use neural network simulations. The neural network approach uses a conventional program to simulate a network of neurons like those in a brain. Each neuron has an output that “fires” when the weighted values of all its input exceed some threshold. These weights are adjusted during the training process, improving the accuracy of the neural network, until the neural network is operating acceptably well. Training is actually often a lot harder than this, and it often requires changes to the configuration of the neural network rather than just changes to weights. However, good software can help to automate much of this process.

In this chapter we will start using pretrained models that have been made from huge sets of data by machine learning experts and are available for us to use for free, courtesy of TensorFlow. We will look at how we can use these models to classify objects in video and sounds in real time, and also how we can hook our own Python code into this process. These standard, pretrained models do require us to do some conditioning of our video source.

We will then take a look at the Edge Impulse platform to illustrate creating and training our own machine learning model. Edge Impulse greatly simplifies the process of using machine learning by providing a cloud service that does all the work that needs high-performance computing—the data preparation and training. Doing all the heavy lifting in the cloud allows you to download the trained model onto a much lower-power device like the Raspberry Pi.

TensorFlow and Edge Impulse require the use of a Raspberry Pi 4 or 400. In addition, you will need a webcam (or Raspberry Pi Camera Module) connected to your Raspberry Pi and a microphone. The microphone on a USB webcam will work just fine (just plug it in), or you can use a USB microphone as described in [Recipe 16.6](#) for the sound projects.

Machine learning is a huge topic about which many books have been written, so by necessity this chapter is intended only to get you started with some ML projects.

## 9.1 Identifying Objects in Video with TensorFlow Lite

### Problem

You want your Raspberry Pi to dynamically identify objects from video.

### Solution

Follow [Recipe 8.1](#) to install OpenCV, which will be needed for most of the recipes in this chapter.

Use a TensorFlow pretrained model with a USB webcam or Raspberry Pi Camera Module. This will dynamically annotate your video stream as objects are identified ([Figure 9-1](#)).

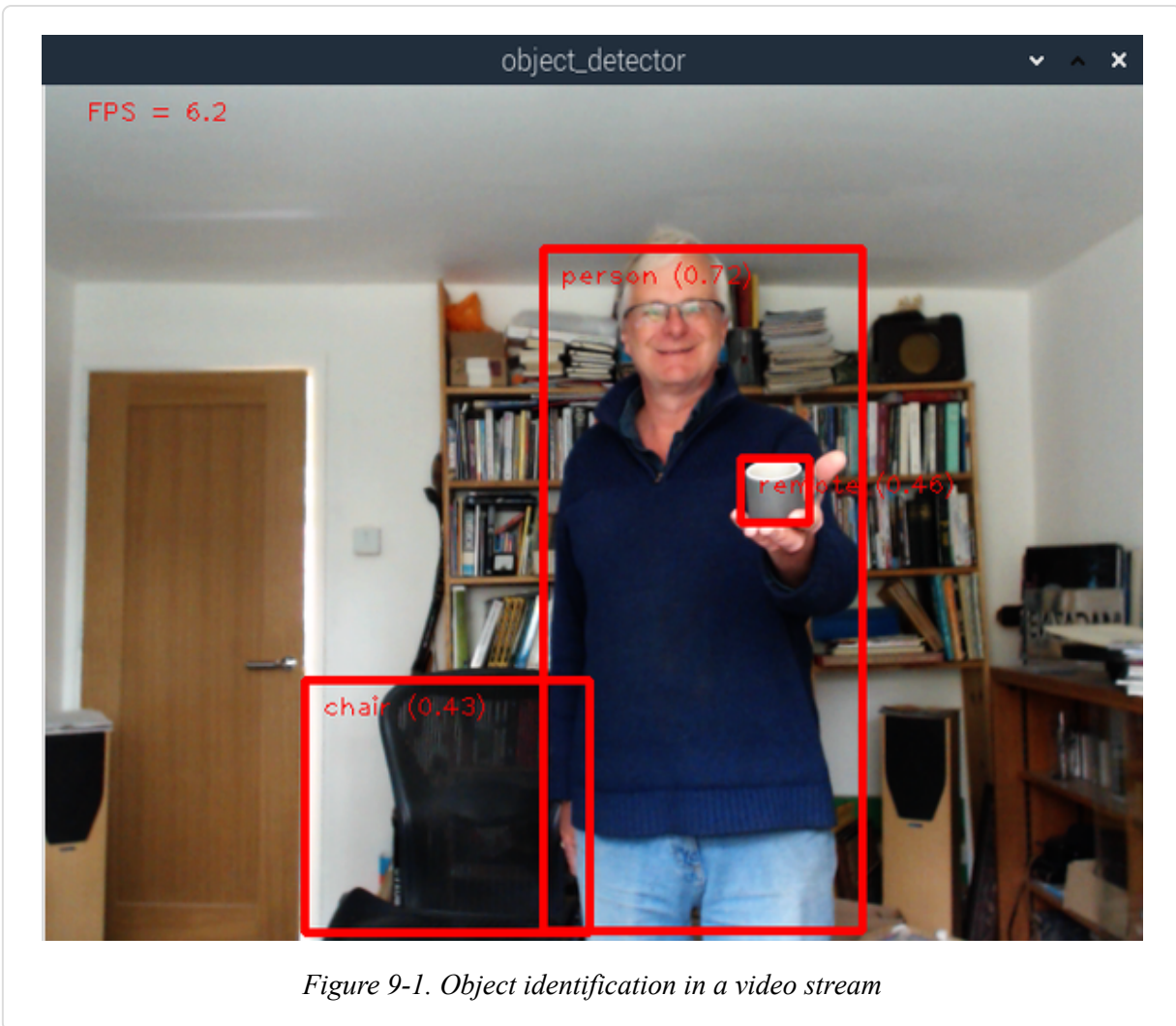


Figure 9-1. Object identification in a video stream

To run this example, change to your home directory and download the TensorFlow examples using the command:

```
$ git clone https://github.com/tensorflow/examples --depth 1
```

Change directory to the following example project folder, build the project by running the *setup.sh* script, and finally run the Python program. This will annotate the video at about 6 frames per second, identifying all sorts of things from your webcam or Raspberry Pi Camera Module. Try holding up various objects for it to identify:

```
$ sudo apt install libportaudio2
```

```
$ cd ~/examples/lite/examples/object_detection/raspberry_pi
$ sh setup.sh
$ python3 detect.py --model efficientdet_lite0.tflite
```

## Discussion

In [Chapter 8](#) we went as far as counting coins. But this relied on classical image processing techniques, looking for edges in images, looking for differences between images, etc. This approach relied on controlled lighting and no machine learning was involved.

Just take a moment to think about what is going on here. This is pretty amazing: your Raspberry Pi is actually seeing and identifying things. It is able to do this because the model has been trained on thousands of images of objects and has learned to distinguish them, or at least hazard a good guess. And it can do this whatever the angle they are viewed at, whatever the lighting and background.

One interesting feature of [Figure 9-1](#) is that the small coffee mug in the author's hand has been misidentified as a “remote,” perhaps because of the way it is being held. Even the mistakes are the sort of mistake that a human might make, because the “seeing” is taking place in a similar way.

## See Also

In [Recipe 9.2](#) we will modify this example program to run some of our own code when an object of interest is detected.

You can use other [ready-trained TensorFlow Lite models](#). Learn more about TensorFlow at <https://www.tensorflow.org>.

## 9.2 Reacting to Objects in Video with TensorFlow Lite

### Problem

You want to use a Python program to carry out an action when a certain type of object is identified from a video stream.

## Solution

Modify the TensorFlow object identification example from [Recipe 9.1](#) so that when a particular type of object is detected, some of your own code is run.

Follow [Recipe 9.1](#) to install the pretrained TensorFlow model for object detection.

Copy the file `ch_09_person_detector.py` to the working folder for the object detection example. Then change to that directory and run the program using the following commands:

```
$ cd ~/examples/lite/examples/object_detection/raspberry_pi/  
$ cp ~/raspberrypi_cookbook_ed4/python/ch_09_person_detector.py .  
$ python3 ch_09_person_detector.py
```

As with all the program examples in this book, you can download this code (see [Recipe 3.22](#)).

When you run the program, whenever a person is in front of your webcam, a message will appear in the Terminal saying “Something Detected!”. When this happens, a date-stamped PNG image file will be created, containing the image that the Raspberry Pi saw:

```
$ ls *.png  
2022-04-21-05-57-27.png  
2022-04-21-05-57-37.png  
2022-04-21-06-19-17.png
```

If you view these files using the File Explorer, then you can double-click one to open it and view it.

## Discussion



The program `ch_09_person_detector.py` started life as a copy of TensorFlow's example program `detect.py`. The program for this recipe is too long to list in full here, so you might like to open it in an editor. Here's the key part, all contained in the `run` function:

```
last_detection_time = 0

# Continuously capture images from the camera and run inference
while cap.isOpened():
    success, image = cap.read()
    if not success:
        sys.exit(
            'ERROR: Unable to read from webcam. Please verify your
            webcam settings.'
        )
    image = cv2.flip(image, 1)

    # Convert the image from BGR to RGB as required by the TFLite
    model.
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Create a TensorImage object from the RGB image.

    input_tensor = vision.TensorImage.create_from_array(rgb_image)

    # Run object detection estimation using the model.
    detection_result = detector.detect(input_tensor)

    # Draw keypoints and edges on input image
    image = utils.visualize(image, detection_result)

    for detection in detection_result.detections:
        object_type = detection.categories[0].category_name
        time_now = time.time()
        if object_type == 'person' and time_now >
last_detection_time + 10:
            print("*****")
            print("Person Detected!")
            print("*****")
            # do your own thing here !
            last_detection_time = time_now
            ts = "{:%Y-%m-%d-%H-%M-
            %S}".format(datetime.datetime.now())
            cv2.imwrite(ts + ".png", image)
```

Our additions to the code are shown in bold.

The variable `last_detection_time` keeps track of when a person was last detected so that it can wait for a while after detecting someone before writing another file. Otherwise, a lot of files would be created.

The line `detection_result = detector.detect(input_tensor)` creates a list of detection events, each of which might look like this:

```
Detection(bounding_box=BoundingBox(origin_x=418, origin_y=285,
width=16,
    height=40),
categories=[Category(index=83, score=0.3515625,
display_name='', category_name='book')])
```

This is a list of tuples, with the first part of the tuple being the bounding box of the thing detected and the second part a list of categories that the object could fit, with the most likely first. That means that to get the name of the category, which will tell us the type of the object, we need to get the first category in that list `[0]` and use its `category_name`.

Iterating over this list of `detection_result`, we check the time and then see if the `object_type` is “person.” If it is, and sufficient time has elapsed since an object was last detected (10 seconds), the file is saved using `cv2.imwrite` using the current date and time as the filename.

You could do all sorts of things with a program like this. For example, you could detect your pet and make an automated pet feeder that controls a motor and deposits some food for the animal.

## See Also

You can use [other ready-trained TensorFlow Lite models](#). Learn more about TensorFlow at <https://www.tensorflow.org>.

## 9.3 Identifying Sounds with TensorFlow Lite

## Problem

You want to make your Raspberry Pi identify different types of sounds that it hears with its microphone.

## Solution

Use a pretrained TensorFlow model from the TensorFlow examples.

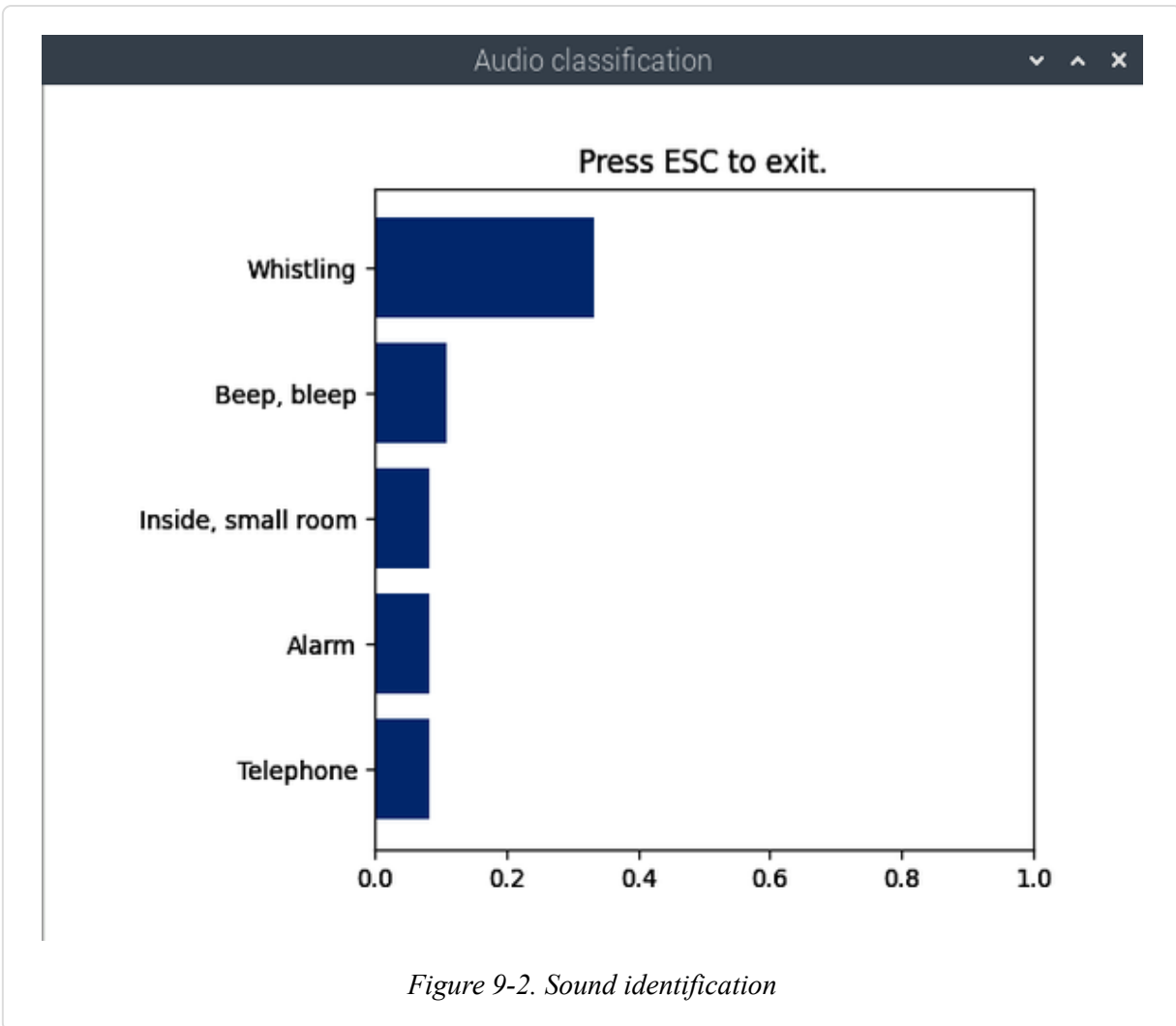
If you have not already done so, change to your home directory and then download the TensorFlow examples using the command:

```
$ git clone https://github.com/tensorflow/examples --depth 1
```

Change directory to the following example project folder, build the project by running the *setup.sh* script, and then finally, run the Python program:

```
$ cd ~/examples/lite/examples/sound_classification/raspberry_pi
$ sh setup.sh
$ python3 classify.py
```

This will open the window shown in [Figure 9-2](#). Try whistling and making some other noises and see how the classifier does.



## Discussion

In this example, anything with a score of less than 0.2 is probably incorrect—either that or TensorFlow has much better hearing than the author. However, the top result seems pretty reliable and the model can identify a lot of different sounds.

Pretrained models like the one used here rely on many thousands of sound samples that have been carefully categorized. Then the model is carefully trained on these samples, along with other background noise, to make something that can usefully identify different sounds.

## See Also

See [Recipe 9.4](#) to add your own code to do something when whistling is detected.

See [Recipe 9.6](#) for training your own network to recognize a spoken command.

## 9.4 Reacting to a Whistle with TensorFlow Lite

### Problem

You want your Raspberry Pi to respond to a whistle by running some of your own Python code.

### Solution

First, install the TensorFlow example in [Recipe 9.3](#), and then adapt the example Python program.

Copy the file `ch_09_detect_whistle.py` to the working folder for the sound classification example. Then change to that directory and run the program using the following commands:

```
$ cd ~/examples/lite/examples/sound_classification/raspberry_pi/  
$ cp ~/raspberrypi_cookbook_ed4/python/ch_09_detect_whistle.py .  
$ python3 ch_09_detect_whistle.py
```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

When you run the program, you will see something like the following. Every time you whistle near the microphone, you'll see the message “Whistling Detected”:

```
$ python3 ch_09_detect_whistle.py  
Listening for Whistles...  
Whistling Detected
```

## Discussion

Let's take a look at the code for this:

```
import time
from tfllite_support.task import audio
from tfllite_support.task import core
from tfllite_support.task import processor

model = 'yamnet.tflite'
num_threads = 4
score_threshold = 0.6
overlapping_factor = 0.5

# Initialize the audio classification model.
base_options = core.BaseOptions(
    file_name=model, use_coral=False, num_threads=num_threads)
classification_options = processor.ClassificationOptions(
    max_results=1, score_threshold=score_threshold)
options = audio.AudioClassifierOptions(
    base_options=base_options,
    classification_options=classification_options)
classifier = audio.AudioClassifier.create_from_options(options)

# Initialize the audio recorder and a tensor to store the audio
input.
audio_record = classifier.create_audio_record()
tensor_audio = classifier.create_input_tensor_audio()

# We'll try to run inference every interval_between_inference
seconds.
# This is usually half of the model's input length to create an
overlapping
# between incoming audio segments to improve classification
accuracy.
input_length_in_second = float(len(
    tensor_audio.buffer)) / tensor_audio.format.sample_rate
interval_between_inference = input_length_in_second * (1 -
overlapping_factor)
pause_time = interval_between_inference * 0.1
last_inference_time = time.time()

# Start audio recording in the background.
audio_record.start_recording()

print('Listening for Whistles...')

while True:
    tensor_audio.load_from_audio_record(audio_record)
```

```
results = classifier.classify(tensor_audio)
if len(results.classifications) > 0:
    classification = results.classifications[0]
    if len(classification.categories) > 0:
        top_category = classification.categories[0]
        if top_category.category_name == 'Whistling':
            print('Whistling Detected')
time.sleep(pause_time)
```

The code is a simplified version of the TensorFlow example, and like that example, a lot of the work is hidden away in the `AudioClassifier` class, which we can make use of in our programs.

The audio classifier is configured with parameters. Most of these options, such as `max_results` and `score_threshold`, are obvious. The `num_threads` option specifies the number of threads of execution (see [Recipe 7.19](#)) devoted to running the example code, and `enable_edge_tpu` is used only if you have machine learning accelerator hardware attached to your Raspberry Pi.

The program will periodically take audio samples, then try to infer the presence of whistling on them. It does this using overlapping time windows, so as to make it more likely that no whistles will be missed.

The main `while` loop checks to see if it's time to look for whistles again. If it is, it classifies the audio, and if there is a result, checks to see if it is a whistle.

The final `time.sleep` allows your Pi's processor to get on with other things for a while before it's needed for classifying again.

## See Also

To hook your own code into video classification, see [Recipe 9.2](#).

## 9.5 Installing Edge Impulse

### Problem

You want to use the Edge Impulse platform to train your own ML model for use on your Raspberry Pi.

## Solution

Much of the work of Edge Impulse takes place on the company's servers and is accessed through [its website](#)). To use these servers, Edge Impulse, not unreasonably, asks you to create an account. For noncommercial “developer” use, this is free.

In addition to the EdgeImpulse website, we also need to install a local part of the software on our Raspberry Pi. Before you start, it's a good idea to make sure that your Raspberry Pi OS is up to date ([Recipe 3.40](#)).

Start by running the following commands in a Terminal window:

```
$ curl -sL https://deb.nodesource.com/setup_12.x | sudo bash -
$ sudo apt install -y gcc g++ make build-essential nodejs sox
  gstreamer1.0-tools
  gstreamer1.0-plugins-good gstreamer1.0-plugins-base
  gstreamer1.0-plugins-base-apps
$ npm config set user root && sudo npm install edge-impulse-linux
-g
  --unsafe-perm
```

Because copying such long commands is error prone and tedious, these commands are also in the file *long\_commands.txt* in the code for the book ([Recipe 3.22](#)).

We are also going to need the Python interface for Edge Impulse. Edge Impulse relies on a Python library called NumPy. This needs to be updated to its latest version:

```
$ pip3 install numpy --upgrade
```

## Discussion

Edge Impulse is based on the TensorFlow machine learning library that we used in our earlier recipes. However, it adds a slick and helpful web-based



user interface and cloud-based servers to make it a lot easier to get started with training our own ML models.

Although Edge Impulse is now installed onto your Raspberry Pi, we need an example to run. We'll explore this in [Recipe 9.6](#).

## See Also

The [Edge Impulse website](#) has a wealth of material.

Learn about [TensorFlow](#), on which Edge Impulse is based.

## 9.6 Recognizing a Spoken Command (in the Cloud)

### Problem

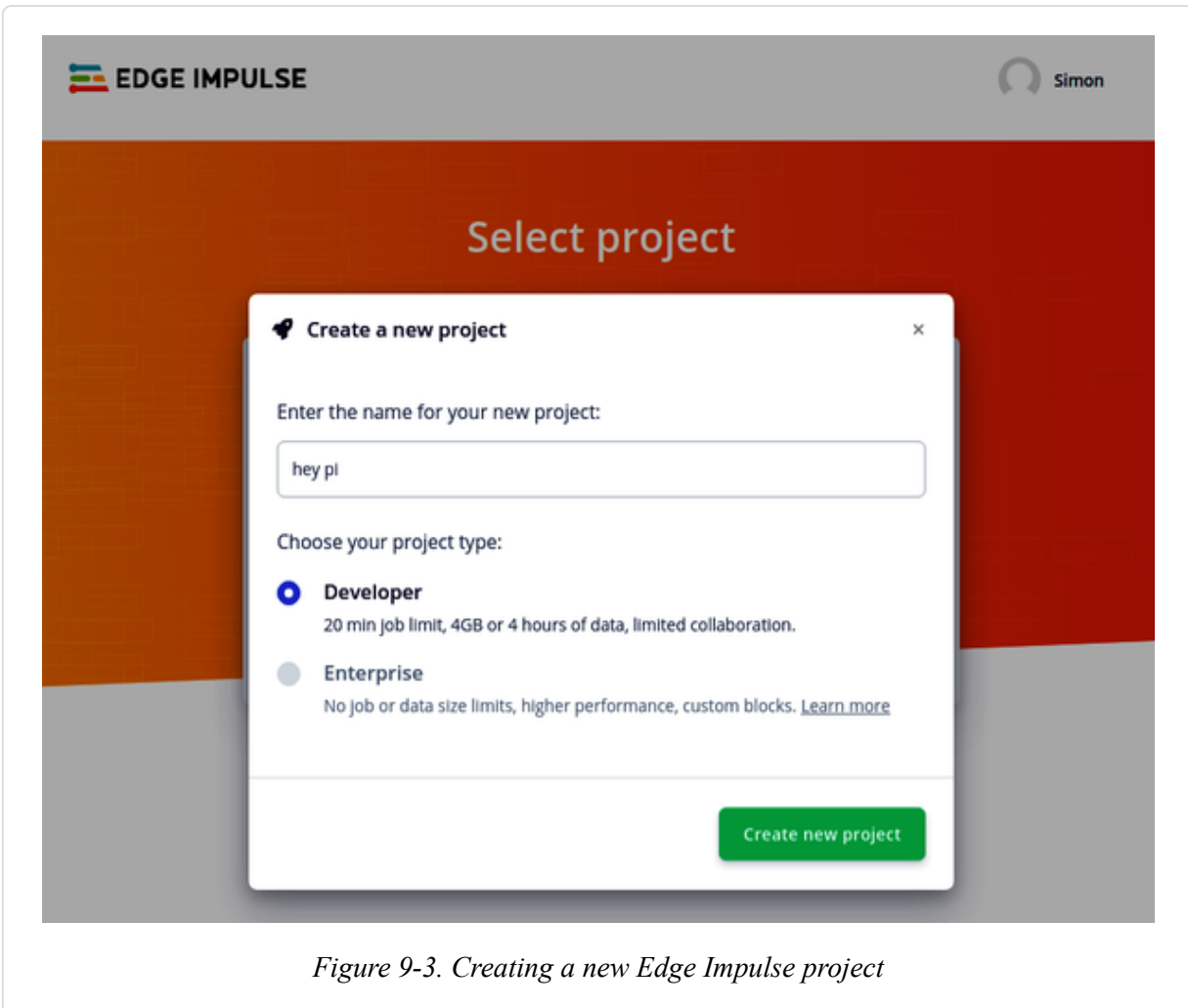
You want your Raspberry Pi to be able to recognize a spoken command using a cloud service.

### Solution

After registering on the Edge Impulse website and installing the software on your Raspberry Pi, use the *wizard* to train a neural network. You can then test how well the trained model works in your browser, before running it locally on your Raspberry Pi.

Go to <https://studio.edgeimpulse.com> and log in using the account you created in [Recipe 9.5](#).

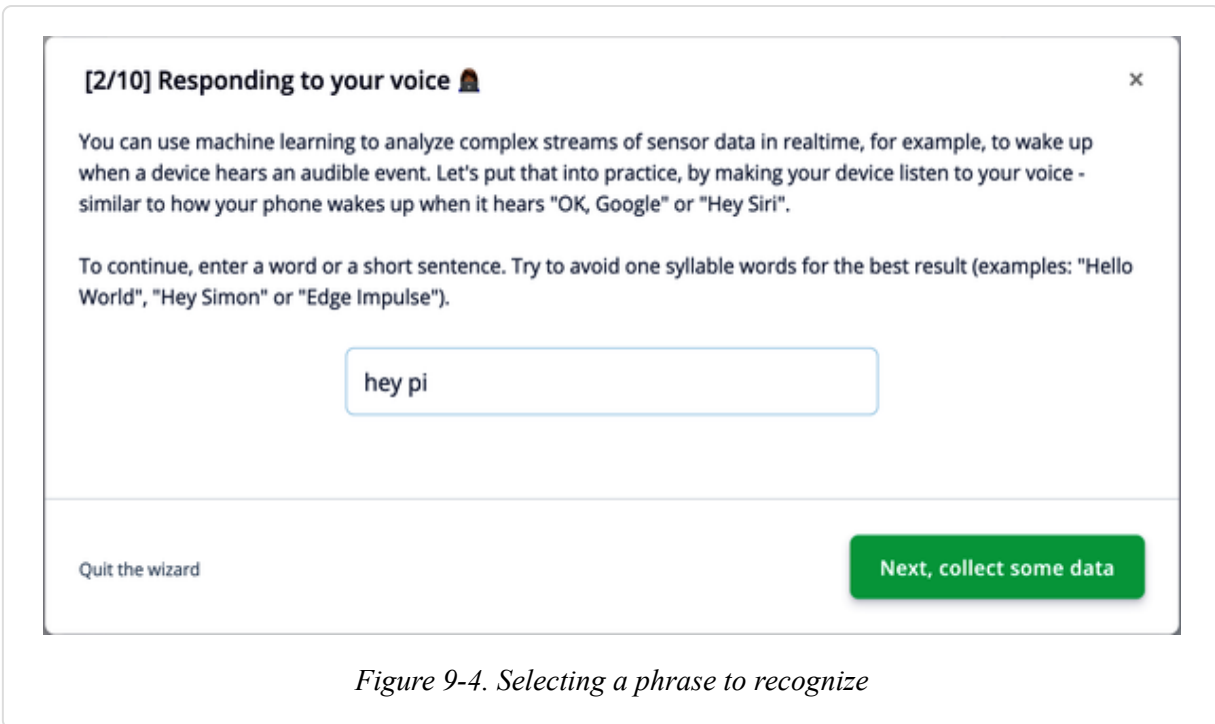
You now need to create a new project and give it a name. If you have just registered with Edge Impulse, the website will probably take you to the New Project Wizard. Cancel this wizard if it has started, and instead click on the “+ Create new project” button. This will open a dialog ([Figure 9-3](#)) where you should enter a name for your project; use the name “hey pi.”



*Figure 9-3. Creating a new Edge Impulse project*

Select the Developer option. This is the free option and imposes a few constraints on the use of Edge Impulse, which we will not be likely to exceed.

Clicking on the “Create new project” button takes us to a dialog for choosing what type of data we are going to be dealing with. However, we’re going to use the getting started wizard, so close this popup; you’ll return to the project page for the “hey pizza” project. Scroll to the bottom of the page and select the option “Launch getting started wizard.” This will warn you that your project will be cleared out, but that’s fine, so select Yes and then confirm the action. The Welcome wizard will launch and offer to make your model in five minutes. Click on the button to accept this option and, for your phrase to be recognized, use “hey pi” (Figure 9-4) and then click Next.



*Figure 9-4. Selecting a phrase to recognize*

Now comes the fun part. For our Raspberry Pi to recognize the phrase “hey pi,” we need to give it lots of examples of us saying that. The wizard will now ask us to record ourselves saying “hey pi” repeatedly for an oddly specific 38 seconds ([Figure 9-5](#)). You’ll also be asked for access to your microphone, which you should grant.

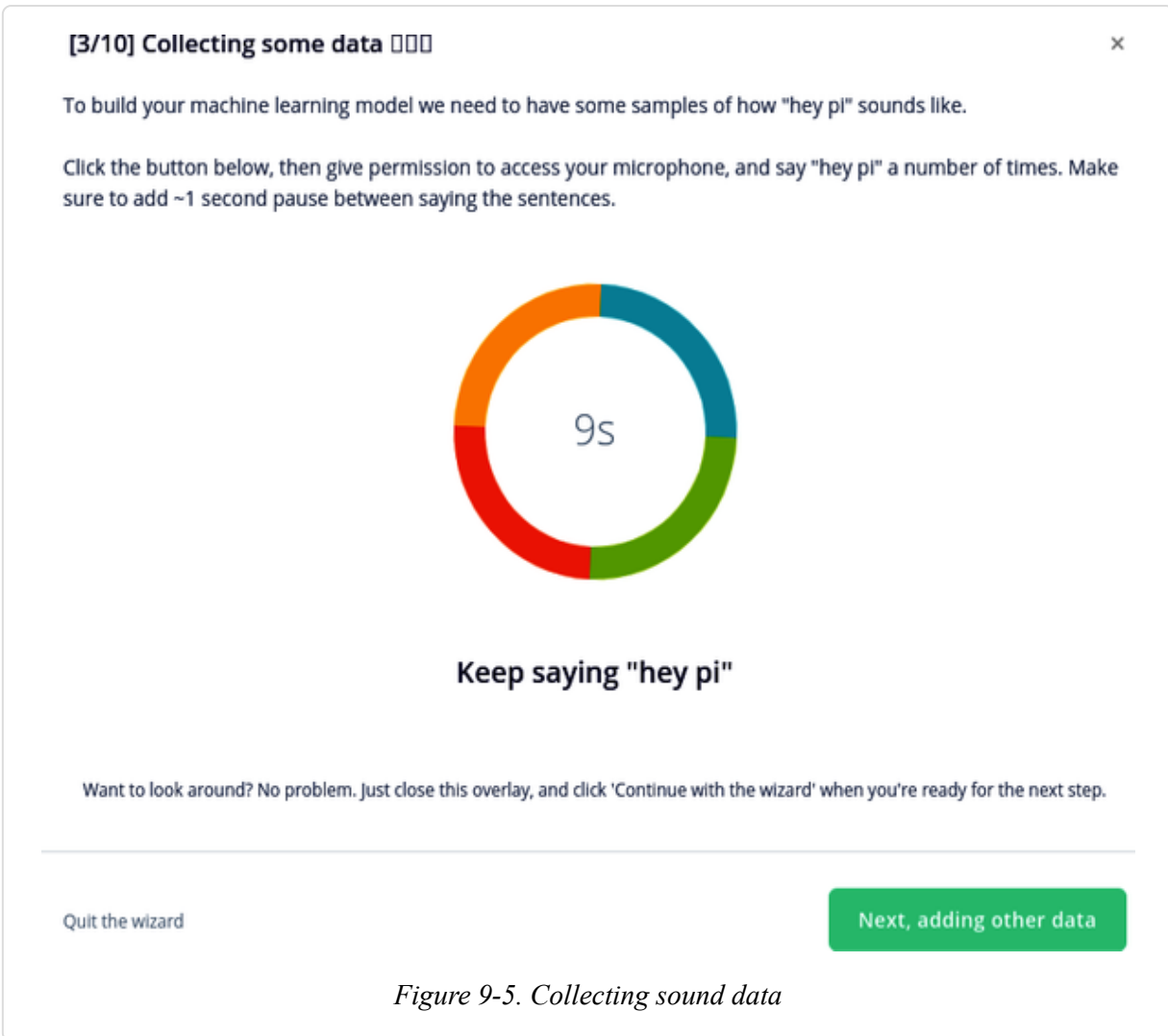


Figure 9-5. Collecting sound data

If Edge Impulse needs more data, it may ask you to say the phrase some more. Eventually, it will have enough samples, and you can move onto the next step of adding other data.

This other data takes the form of random words and background noise from Edge Impulse’s collections of such data and will help the neural network to distinguish “hey pi” from any other sounds that the microphone may be picking up.

The next step of the wizard is to “Design your Impulse.” An *impulse* is Edge Impulse’s way of describing the neural network (or other type of ML model) as well as the associated preprocessing of data. In designing the impulse, we

also get the opportunity to look inside the model and gain some reassurance about how this is all working.

If you want to take a look at what's going on behind the wizard, you can just close the wizard overlay and then continue with it later. For example, **Figure 9-6** shows what's called a *feature explorer* view of the sound samples, where samples of "hey pi," noise, and other random phrases are all plotted in a 3D space. The instances of "hey pi" are all clustered together, indicating a good recognition by the impulse.

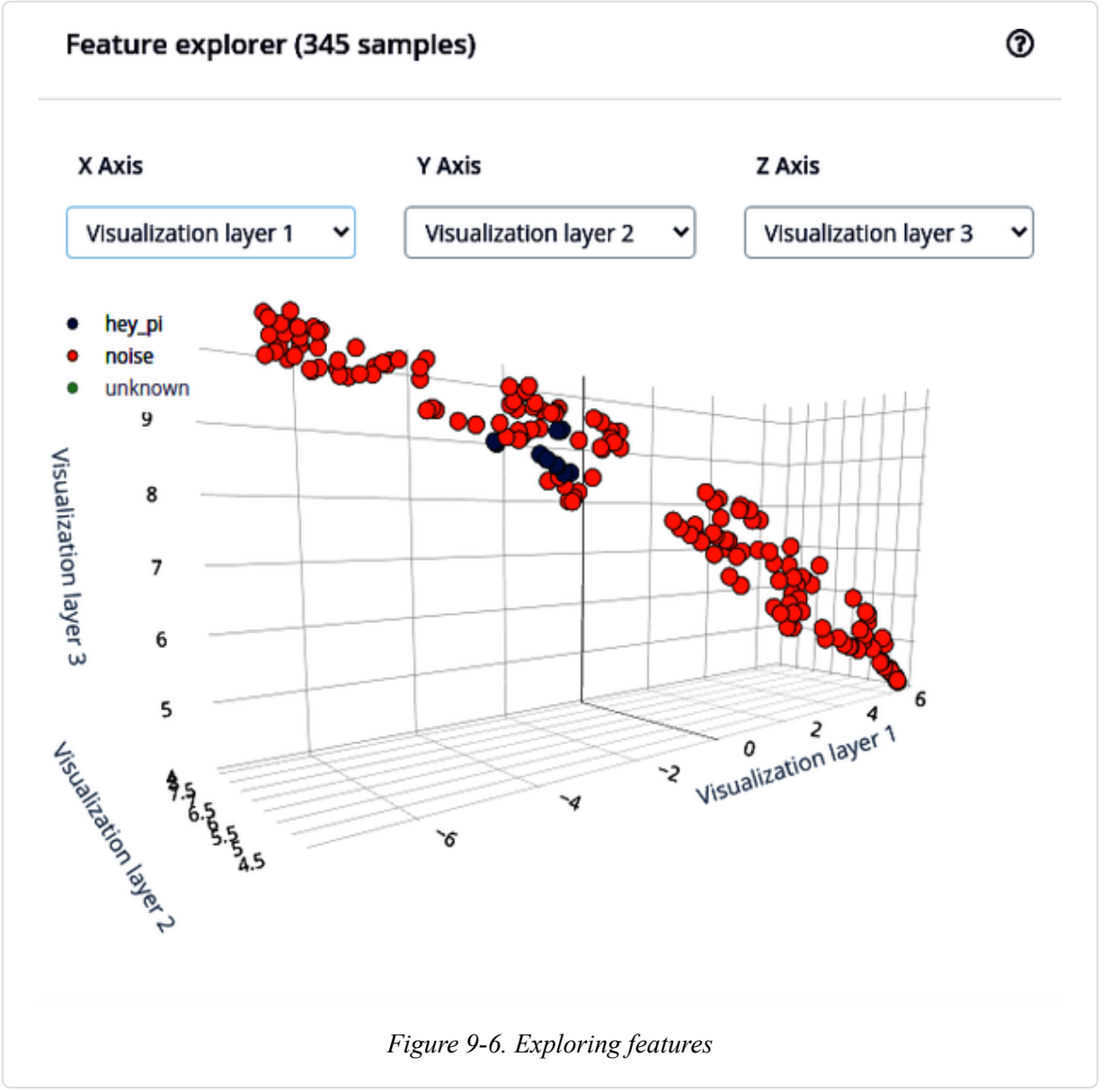


Figure 9-6. Exploring features

When the wizard resumes, its next step is to train the neural network. This takes a few minutes as the weights of the neural network are adjusted until it can accurately identify the phrase: “hey pi.” We can now check out the impulse and see how well it performs, all within our browser ([Figure 9-7](#)). Note that before letting us play with the impulse, it is built into a deployable package that we will later be able to deploy onto our Raspberry Pi.

## [9/10] Taking the model for a spin

×

You have trained your model, now let's take it for a spin... Wait for the model to build, then click 'Give access to the microphone', and see how the model performs!



### Classifier



...

	HEY_PI	NOISE	UNKNOWN
493	0.00	0.14	<b>0.86</b>
492	0.00	0.03	<b>0.97</b>
491	0.01	<b>0.59</b>	0.40
490	0.00	0.01	<b>0.99</b>
489	0.01	0.00	<b>0.99</b>
488	<b>0.87</b>	0.09	0.04
487	0.01	0.41	<b>0.58</b>
486	0.07	<b>0.61</b>	0.32

Quit the wizard

Next, check out next steps

Figure 9-7. Testing the impulse

In [Figure 9-7](#) whenever we say something, or a noise is heard, the impulse categorizes it as either HEY\_PI, NOISE, or UNKNOWN, and against each of these, it offers a probability. So if you look at the row that starts 488 (the sample number), you can see that there is an 87% probability that “hey pi” was detected and very low probability that noise or an unknown sound was detected.

## Discussion

To see how well this speech recognition is working, try getting other people to say “hey pi” or try saying similar phrases yourself.

This makes a very impressive demonstration. But, at the moment, everything is happening on the Edge Impulse servers. The only thing happening on our Raspberry Pi is the passing of sound data from our microphone to Edge Impulse. In [Recipe 9.7](#) we will look at how we can bring some of the action back to our Raspberry Pi.

## See Also

Edge Impulse has a lot of [useful documentation](#).

# 9.7 Recognizing a Spoken Command (Locally)

## Problem

You want your Raspberry Pi to be able to recognize a spoken command locally, without relying on the internet.

## Solution

If you haven’t already done so, you’ll need to follow [Recipes 9.5](#) and [9.6](#).

To run our phrase recognition impulse locally on the Raspberry Pi, we can download it and then run it using the Edge Impulse Linux runner. This will



already be installed as part of [Recipe 9.5](#).

Run the following command (note that if this is the first time that you are doing this, then you don't need the `--clean` option):

```
$ edge-impulse-linux-runner --clean
Edge Impulse Linux runner v1.3.5
? What is your user name or e-mail address (edgeimpulse.com)?
  anonymised@email.com
? What is your password? [hidden]

[RUN] Downloading model...
[BLD] Created build job with ID 2540059
[BLD] Writing templates...
.... lots of build messages
[BLD] Building binary OK
[RUN] Downloading model OK
[RUN] Stored model version in /home/pi/.ei-linux-
runner/models/93963/v2/model.eim
[RUN] Starting the audio classifier for Simon / hey pi (v2)
[RUN] Parameters freq 16000Hz window length 1000ms. classes
  [ 'hey_pi', 'noise', 'unknown' ]
? Select a microphone USB-Audio - HD Pro Webcam C920
[RUN] Using microphone hw:2,0
classifyRes 11ms. { hey_pi: '0.0049', noise: '0.9479', unknown:
'0.0472' }
classifyRes 5ms. { hey_pi: '0.9590', noise: '0.0001', unknown:
'0.0409' }
classifyRes 5ms. { hey_pi: '0.9899', noise: '0.0000', unknown:
'0.0101' }
q
```

This is actually quite a lengthy process, with a lot of build messages appearing in the Terminal, that I have omitted from the preceding transcript.

When you run the command, you will be asked for your Edge Impulse login details (email and password). This will start the process of building and downloading the impulse into a compact form that you can run on your Raspberry Pi. Building takes a while, but it only needs to be done once.

Once the build process is complete, you will be prompted to select the microphone that you want to use and finally your Raspberry Pi will start

listening to you, and reporting the results of each sample it hears and providing a probability of each sample being “hey pi.”

## **Discussion**

The really neat thing about this process is that our advanced machine learning model has been built and compressed into a form that we can run on a relatively modestly powerful Raspberry Pi. You can see information about this from the Edge Impulse Dashboard.

Take a little time to explore the information there. In particular, try clicking on Transfer Learning from the Impulse Design option on the sidebar (Figure 9-8).



Figure 9-8. Details of the deployed impulse

As you can see, among other interesting information, the deployed impulse can run in 45k bytes of memory and requires only about 123k bytes of permanent storage.

Although we can now run the impulse directly on our Raspberry Pi, it would be nice to get this to work with Python so that we can write our own projects using Edge Impulse. This is the topic of [Recipe 9.8](#).

## See Also

You can find documentation for Edge Impulse on Linux at <https://oreil.ly/9gCre>.

# 9.8 Responding to a Spoken Command in Python

## Problem

Seeing the Edge Impulse system recognize a spoken phrase is pretty amazing, but you want to take this a step further and have it trigger an action in your own Python program.

## Solution

First of all, set up Edge Impulse by following [Recipe 9.5](#), and then create a trained impulse by following [Recipe 9.6](#).

Next, you need to install the Python SDK (software development kit):

```
$ sudo apt install libatlas-base-dev libportaudio0 libportaudio2  
libportaudiocpp0  
portaudio19-dev  
$ pip3 install edge_impulse_linux -i https://oreil.ly/ualnS
```

I found that I had to update my version of the Python NumPy library and install the PyAudio module by running:

```
$ pip3 install numpy --upgrade  
$ pip3 install pyaudio
```

Next, let's download the trained model from the Edge Impulse server using the following command:

```
$ edge-impulse-linux-runner --download modelfile.eim
```

The program *09\_hey\_pi.py* listed as follows will respond by printing “Hello you!” every time it hears the phrase “hey pi.” Before running it, you may need to change the line `audio_device_id` from 2 to the ID for your microphone:

```
import sys
import signal
from edge_impulse_linux.audio import AudioImpulseRunner

modelfile = '/home/pi/modelfile.eim'
audio_device_id = 2

runner = None

def signal_handler(sig, frame):
    print('Interrupted')
    if (runner):
        runner.stop()
    sys.exit(0)

signal.signal(signal.SIGINT, signal_handler)

with AudioImpulseRunner(modelfile) as runner:
    try:
        model_info = runner.init()
        labels = model_info['model_parameters']['labels']
        print('Loaded runner for "' + model_info['project']
['owner'] + ' / ' +
            model_info['project']['name'] + '"')

        for res, audio in
runner.classifier(device_id=audio_device_id):
            score = res['result']['classification']['hey_pi']
            if (score > 0.7):
                print('Hello you!')

    finally:
        if (runner):
            runner.stop()
```

Run the program and try experimenting with different phrases to see how accurate the detection of “hey pi” is:

```
$ python3 07_hey_pi.py
Loaded runner for "Simon / Hey Pi"
selected Audio device: 2
Hello you!
Hello you!
Hello you!
```

## Discussion

The program is based around the `AudioImpulseRunner` from the Edge Impulse Python library. This runner repeatedly listens to audio samples and then attempts to classify the sounds it hears. In this case, it's determining if the phrase "hey pi" has been heard. This runner holds the microphone as a resource, so it is important that if Ctrl-C is pressed to stop the program, the code ends neatly, releasing the audio device. That is where the `signal_handler` function comes in, linked to `signal.SIGINT` (Ctrl-C).

Inside the runner's `with/as` block, the runner is first initialized using `runner.init()` and then details of the model being used are displayed.

The `for` loop effectively iterates over an endless stream of results supplied by `runner.classifier`. Each result has a `score`, and if the `score` is greater than 0.7, the message "Hello you!" is printed. The threshold of 0.7 is quite low, so you'll find that you can fool your Pi into responding using similar phrases. Try increasing it to 0.95.

## See Also

Although all we are doing here is printing a message when the phrase is detected, there is no reason why we couldn't use this to control hardware using the GPIO pins ([Recipe 10.1](#)) or with some of the recipes in [Chapter 18](#).

# Chapter 10. Hardware Basics

---

## 10.0 Introduction

This chapter contains some basic recipes for setting up and using the Raspberry Pi’s general-purpose input/output (GPIO) connector. This connector enables you to connect all sorts of interesting electronics to your Raspberry Pi.

## 10.1 Finding Your Way Around the GPIO Connector

### Problem

You need to connect electronics to the GPIO connector, but first you need to know more about what all the pins do.

### Solution

There have been three versions of the Raspberry Pi GPIO connector: two 26-pin layouts for the original Raspberry Pi, and one 40-pin layout that was introduced with the Raspberry Pi “+” models and has been in use ever since.

A 26-pin Raspberry Pi is more of a vintage collectible than a practical computer, and you’ll likely find it slow and incompatible with a whole slew of software. So, for practical terms, you’ll probably need to get a Raspberry Pi 4 or 400, or at least a Pi 3.

**Figure 10-1** shows the current 40-pin layout, which is the same for all 40-pin GPIO Raspberry Pi models right up to the Raspberry Pi 4 and 400.

The top 26 pins are the same as the 26 pins of the original Raspberry Pi model B revision 2. This allows the 40-pin Raspberry Pi models to use

hardware and software designed for the earlier 26-pin Raspberry Pi designs. The extra pins of the 40-pin connector are made up of three useful extra GND connections and nine GPIO pins. The ID\_SD and ID\_SC pins are intended for use in communicating with a special serial memory chip, which can be included on interface boards that conform to the hardware attached on top (HAT) standard and allows the Raspberry Pi to identify the board (see the Discussion section).



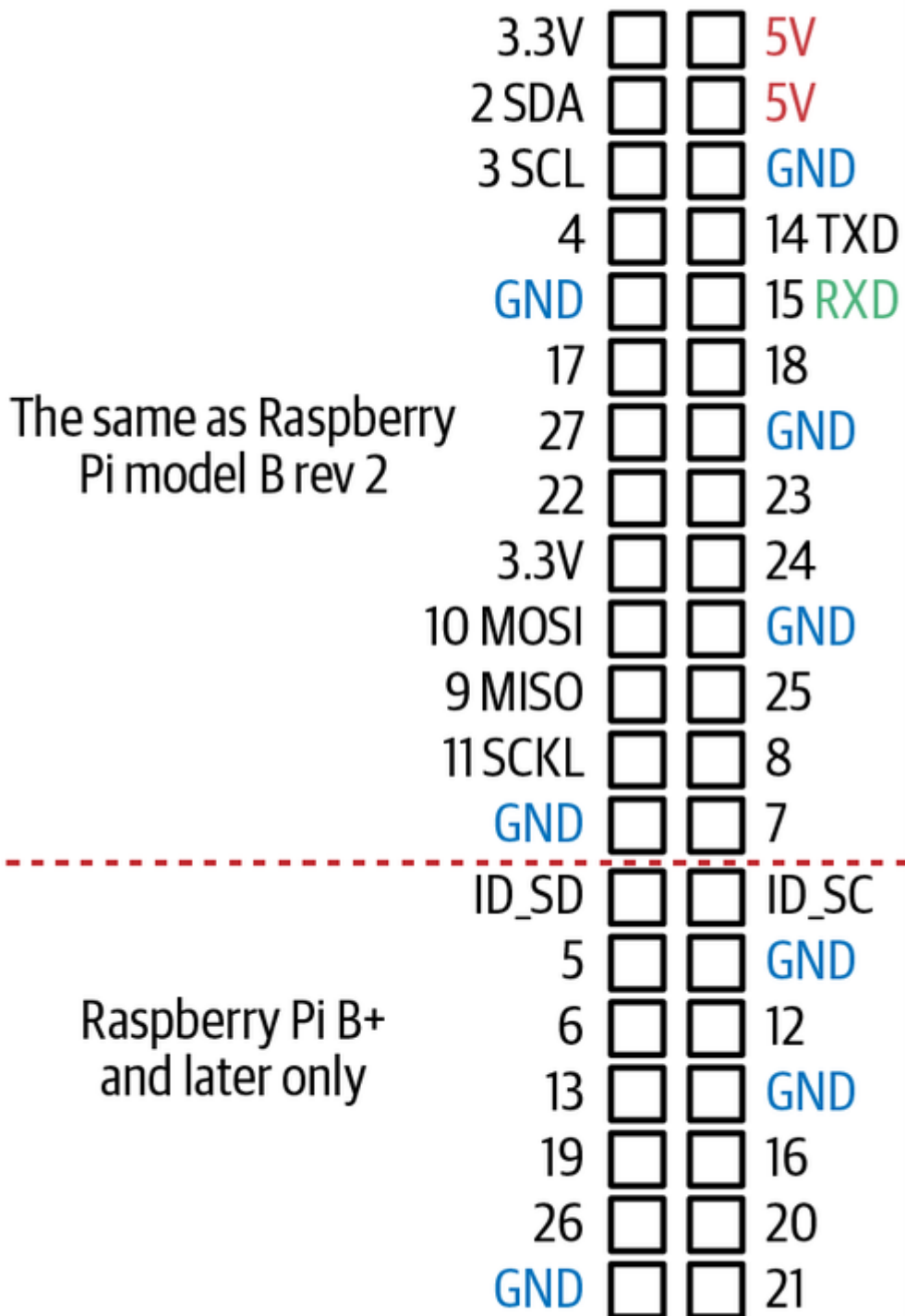


Figure 10-1. The GPIO pinout (40-pin models)

At the top of the connector, there are 3.3V and 5V power supplies. The GPIO uses 3.3V for all inputs and outputs. Any pin with a number next to it can act as a GPIO pin. Those that have another name after the number also have some other special purpose: 14 TXD and 15 RXD are the transmit and receive pins of the serial interface; 2 SDA and 3 SCL form the I2C interface; and 10 MOSI, 9 MISO, and 11 SCKL form the SPI interface.

### **3V Only**

The GPIO connector has both 3V (actually 3.3V) and 5V power pins. This gives the false impression that the Raspberry Pi is OK with you connecting 5V electronics to it. Although it can supply 5V power to a device, all connections to GPIO pins on the Pi must be 3V, or they will damage your Raspberry Pi. Typically, connecting a 5V connection to a GPIO pin will burn out that pin and possibly the whole of the Raspberry Pi's processor.

### **Discussion**

Working out which pin is which on a Raspberry Pi is quite error prone if you rely on counting down the pin connector to find the pin you need. A better way of finding the correct pin is to use a GPIO template like the Raspberry Leaf shown in [Figure 10-2](#).

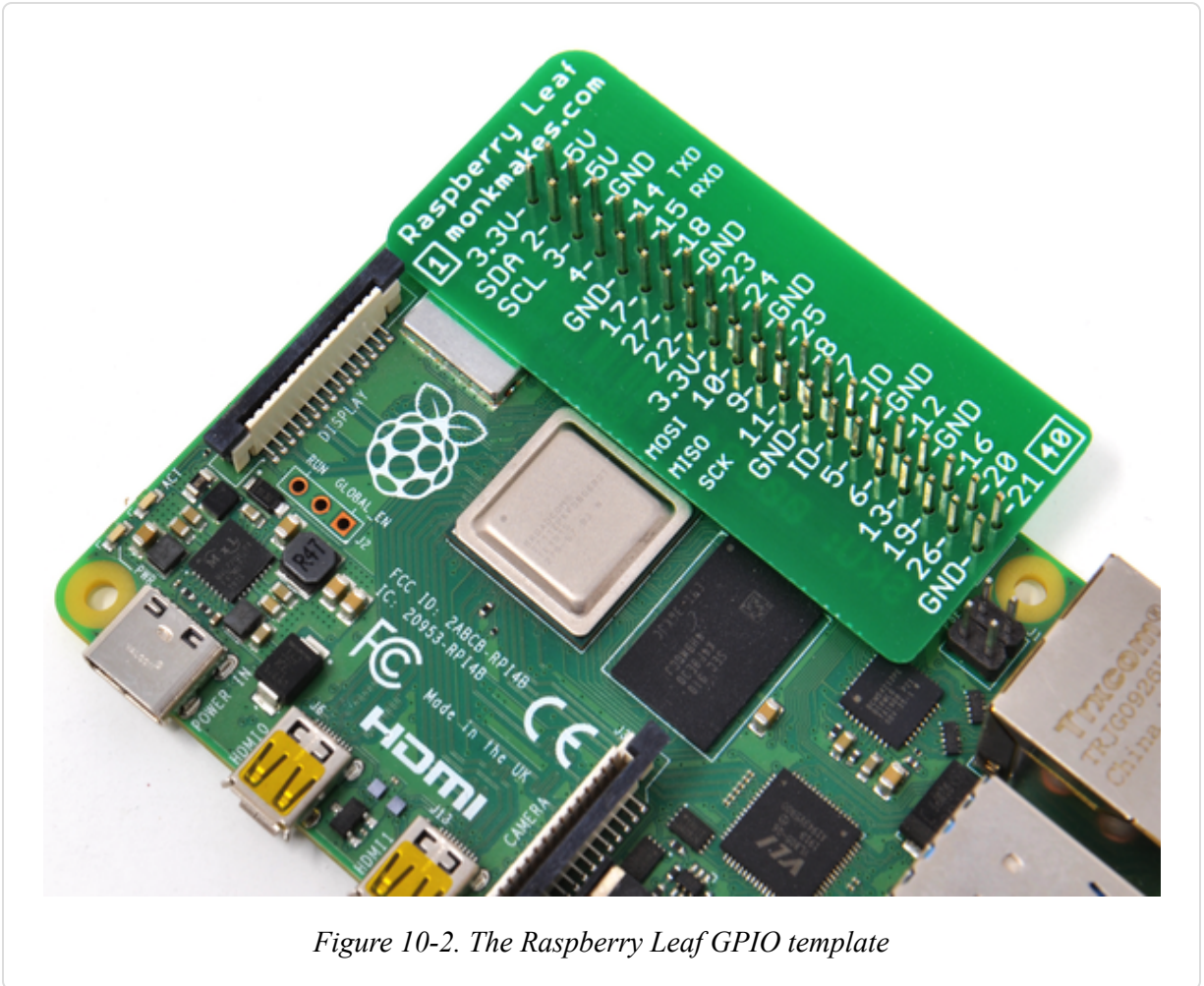


Figure 10-2. The Raspberry Leaf GPIO template

This template fits over the GPIO pins, indicating which pin is which. Other GPIO templates include the Pi GPIO reference board.

The HAT standard is an interface standard that you can use with the Raspberry Pi 4, 3, 2, B+, A+, and Zero. This standard does not in any way stop you from just using GPIO pins directly; however, interface boards that conform to the HAT standard can call themselves HATs. HATs differ from regular Raspberry Pi interface boards in that they must contain a little electrically erasable programmable read-only memory (EEPROM) chip that is used to identify the HAT so that ultimately the Raspberry Pi can autoinstall necessary software. As of this writing, HATs have not quite met that level of sophistication, but the idea is a good one. The pins ID\_SD and ID\_SC are used to communicate with a HAT EEPROM.

## See Also

The Raspberry Pi GPIO connector has only digital inputs and outputs; it doesn't have the analog inputs found on some similar boards. You can get around this shortcoming by using a separate analog-to-digital converter (ADC) chip ([Recipe 14.7](#)) or by using resistive sensors ([Recipe 14.1](#)).

For an example of a HAT, see the Sense HAT described in [Recipe 10.15](#).

## 10.2 Using the GPIO Connector on a Raspberry Pi 400

### Problem

The GPIO connector on the Raspberry Pi 400 is a little hard to reach because it's on the back and recessed. How do I access it easily?

### Solution

The pinout of the GPIO connector on a Raspberry Pi 400 is the same as all the 40-pin GPIO Raspberry Pis. To make it easier to attach jumper wires to the connector, use a GPIO adapter like the [MonkMakes GPIO Adapter](#) for Pi 400 shown in [Figure 10-3](#).

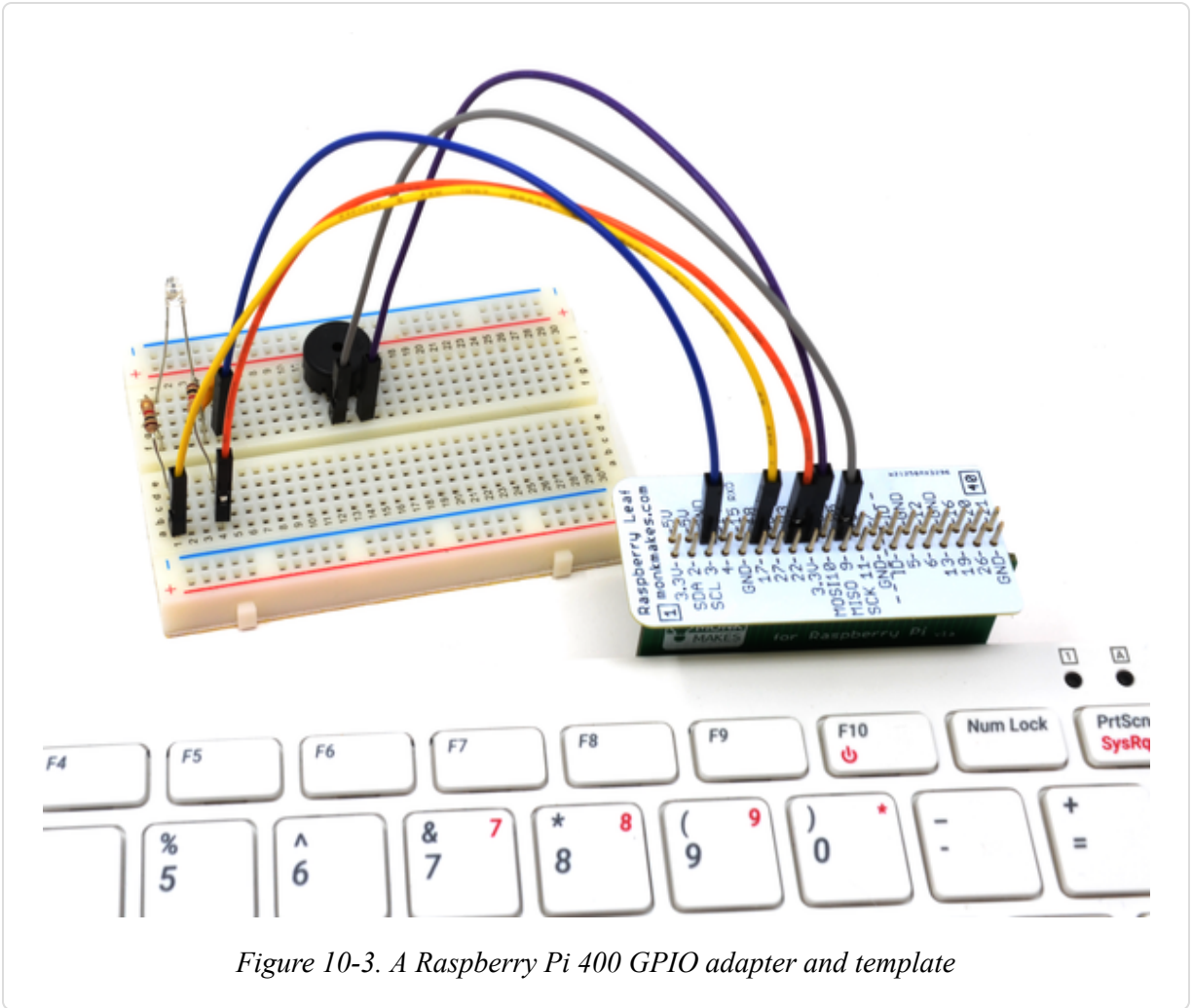


Figure 10-3. A Raspberry Pi 400 GPIO adapter and template

Other types of GPIO adapters are available from SparkFun, Pi Hut, and others.

## Discussion

You are unlikely to want to embed a bulky and relatively expensive Pi 400 into your electronics project. However, if your goal is to learn about electronics with your Pi 400, then an adapter is a lot easier than trying to attach jumper wires to the back of the Pi 400.

HATs don't fit onto the Pi 400 without an adapter either, so a GPIO adapter also allows add-on HATs to be used with the Pi 400, such as the Sense HAT shown in [Figure 10-4](#).



*Figure 10-4. A Raspberry Pi 400 with Sense HAT*

There are a few add-on boards (Adafruit calls them *bonnets*) designed to work well with the Pi 400, such as the Air Quality board shown in [Figure 10-5](#), the Pimoroni Breakout Garden for Pi 400, and the Adafruit Cyberdeck Bonnet.



Figure 10-5. The Air Quality board for Raspberry Pi 400

## See Also

For more information on the Sense HAT pictured in [Figure 10-4](#), see [Recipe 10.15](#).

## 10.3 Keeping Your Raspberry Pi Safe When Using the GPIO Connector

### Problem

You want to connect external electronics to your Raspberry Pi and don't want to accidentally damage or break it.

### Solution

Obey these simple rules to reduce the risk of damaging your Raspberry Pi when using the GPIO connector:

- Do not poke at the GPIO connector with a screwdriver or any metal object when the Pi is powered up.
- Do not connect electronic components to the GPIO pins or a breadboard with the Raspberry Pi powered on.
- Do not power the Pi with more than 5V.
- Always connect the Raspberry Pi GND pin to the GND connection of whatever device you are attaching.
- Do not put more than 3.3V on any GPIO pin being used as an input.
- Do not draw more than 16mA per output; keep the total for all outputs below 100mA on a 40-pin Raspberry Pi and below 50mA for an old 26-pin Raspberry Pi.
- When using LEDs, 3mA is enough to light a red LED reasonably brightly with a 470Ω series resistor.
- Do not draw more than a total of 250mA from the 5V supply pins for Raspberry Pi models 1 to 3. For the Pi 4, 5V power comes directly from USB, so the maximum depends on your power supply. 1A is a reasonable maximum with a 3A power supply.

## Discussion

There is no doubt about it: the Raspberry Pi is a little fragile when it comes to adding external electronics. The newer Raspberry Pi models are a bit more robust but still quite easy to break. Exercise caution and check what you have done *before* you power up the Raspberry Pi, or you run the risk of having to replace it.

## See Also

Read this very good discussion of the [Raspberry Pi's GPIO output capabilities](#).

## 10.4 Setting Up I2C

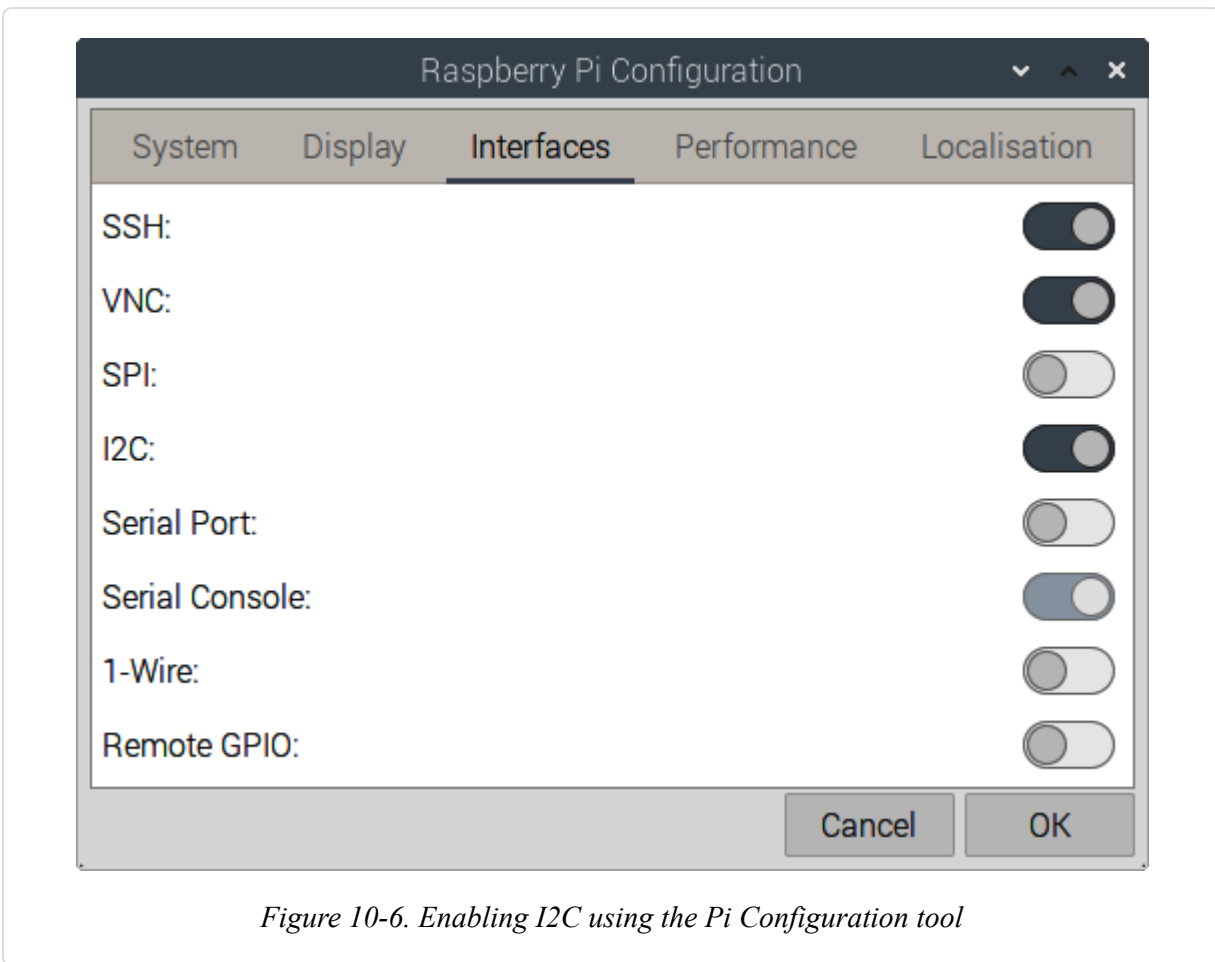


## Problem

You want to set up the I2C bus (Inter-Integrated Circuit) so that you can use some add-ons that require it with your Raspberry Pi.

## Solution

In the latest versions of Raspberry Pi OS, enabling I2C is simply a matter of using the Raspberry Pi Configuration tool that you will find on the Raspberry Menu, under Preferences (**Figure 10-6**). On the Interfaces tab, click the toggle switch for I2C to turn it on and then click OK.



*Figure 10-6. Enabling I2C using the Pi Configuration tool*

On older versions of Raspberry Pi OS, or if you prefer the command line, the `raspi-config` utility does the same job.

Start `raspi-config` using the following command:

```
$ sudo raspi-config
```

Then, from the menu that appears, select Interfacing Options and scroll down to I2C (Figure 10-7).

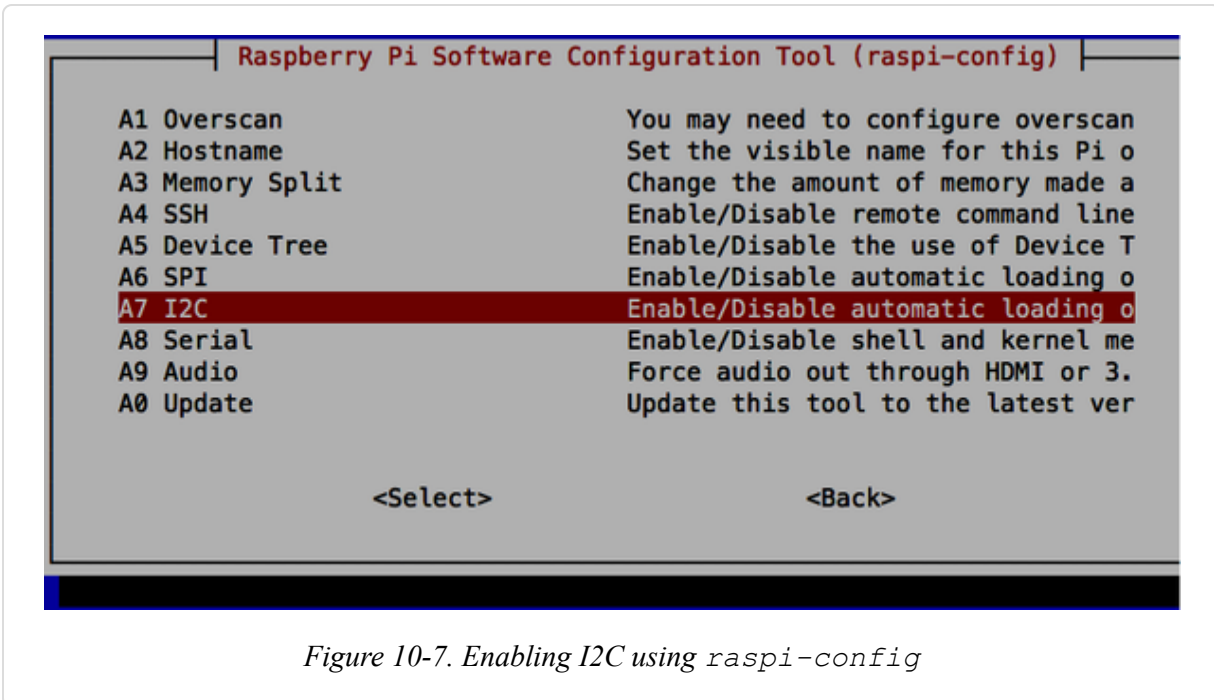


Figure 10-7. Enabling I2C using *raspi-config*

You are then asked, “Would you like the ARM I2C interface to be enabled?” to which you should respond **Yes**. You will also be asked if you want the I2C module loading at startup, to which you should also respond **Yes**.

At this point, you’ll probably also want to install the Python I2C library using this command:

```
$ sudo apt install python-smbus
```

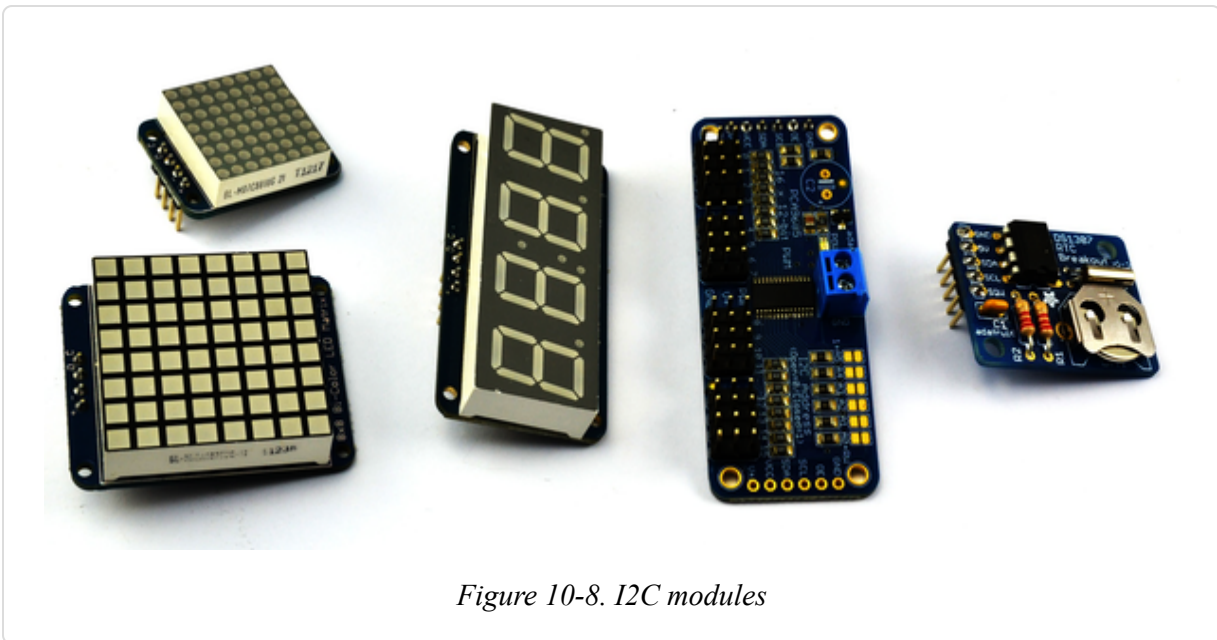
You will then need to reboot the Raspberry Pi for the changes to take effect.

## Discussion

Using I2C modules is a really good way of interfacing with the Pi. It reduces the number of wires that you need to connect everything (to just four), and some really neat I2C modules are available.

However, don't forget to calculate the total of the current used by the I2C modules and make sure that it doesn't exceed the limits specified in [Recipe 10.3](#).

[Figure 10-8](#) shows a selection of I2C modules available from Adafruit. Other suppliers, such as SparkFun, also have I2C devices. From left to right in the figure, there are LED matrix displays; a four-digit, seven-segment LED display; a 16-channel PWM/servo controller; and a real-time clock module.



*Figure 10-8. I2C modules*

Other I2C modules include FM radio transmitters, ultrasonic range finders, OLED (organic light-emitting diode) displays, and various types of sensors.

## See Also

See some of the I2C recipes in this book, including [Recipes 12.3](#), [15.1](#), and [15.2](#).

## 10.5 Using I2C Tools

### Problem

You have an I2C device attached to your Raspberry Pi, and you want to check that it is attached properly and find the correct I2C address to use with the device.

### Solution

Install and use `i2c-tools`.

#### TIP

On newer distributions, you might find that `i2c-tools` is already installed.

From a Terminal window on your Pi, type the following commands to fetch and install the `i2c-tools`:

```
$ sudo apt install i2c-tools
```

Attach your I2C device to the Pi and run the command:

```
$ sudo i2cdetect -y 1
```

Note that if you are using a very old Raspberry Pi revision 1 board, you need to change `1` to `0` in the preceding line of code.

If I2C is available, you will see some output like that shown in [Figure 10-9](#), which indicates that two I2C addresses are in use—`0x68` and `0x70`.

## HEXADECIMAL

Hexadecimal (or just *hex*) is a way of representing numbers using the number base 16 rather than the number base 10 that we use in everyday life.

In hexadecimal, each digit can have one of sixteen possible values. In addition to the familiar digit values of 0 to 9, hex uses the letters A to F; the letter A represents decimal 10, and F decimal 15.

There is no particular reason to use hex over decimal, except that in the unlikely event that you want to convert a number into binary, it's much easier to do from hex than decimal.

To avoid confusion as to whether a number is being represented in decimal or hex, it is common to prefix hex numbers with 0x. In the preceding example, the hex number 0x68 is (in decimal)  $6 \times 16 + 8 = 104$ , and 0x70 is  $7 \times 16 = 112$ .



```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  68  --  --  --  --  --  --
70: 70  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspberrypi ~ $
```

Figure 10-9. *i2c-tools*

## Discussion

`i2cdetect` is a useful diagnostic tool and worth running the first time you use a new I2C device.

You need to ensure that the I2C address used by the device is the same as that required by the software. You'll sometimes find little switches or solder bridges that can be used to change the I2C address of the device. This is

especially useful if you have more than one device with the same address connected to the same Raspberry Pi.

## See Also

See some of the I2C recipes in this book, including Recipes [12.3](#), [15.1](#), and [15.2](#).

For more information on installing with `apt`, see [Recipe 3.17](#).

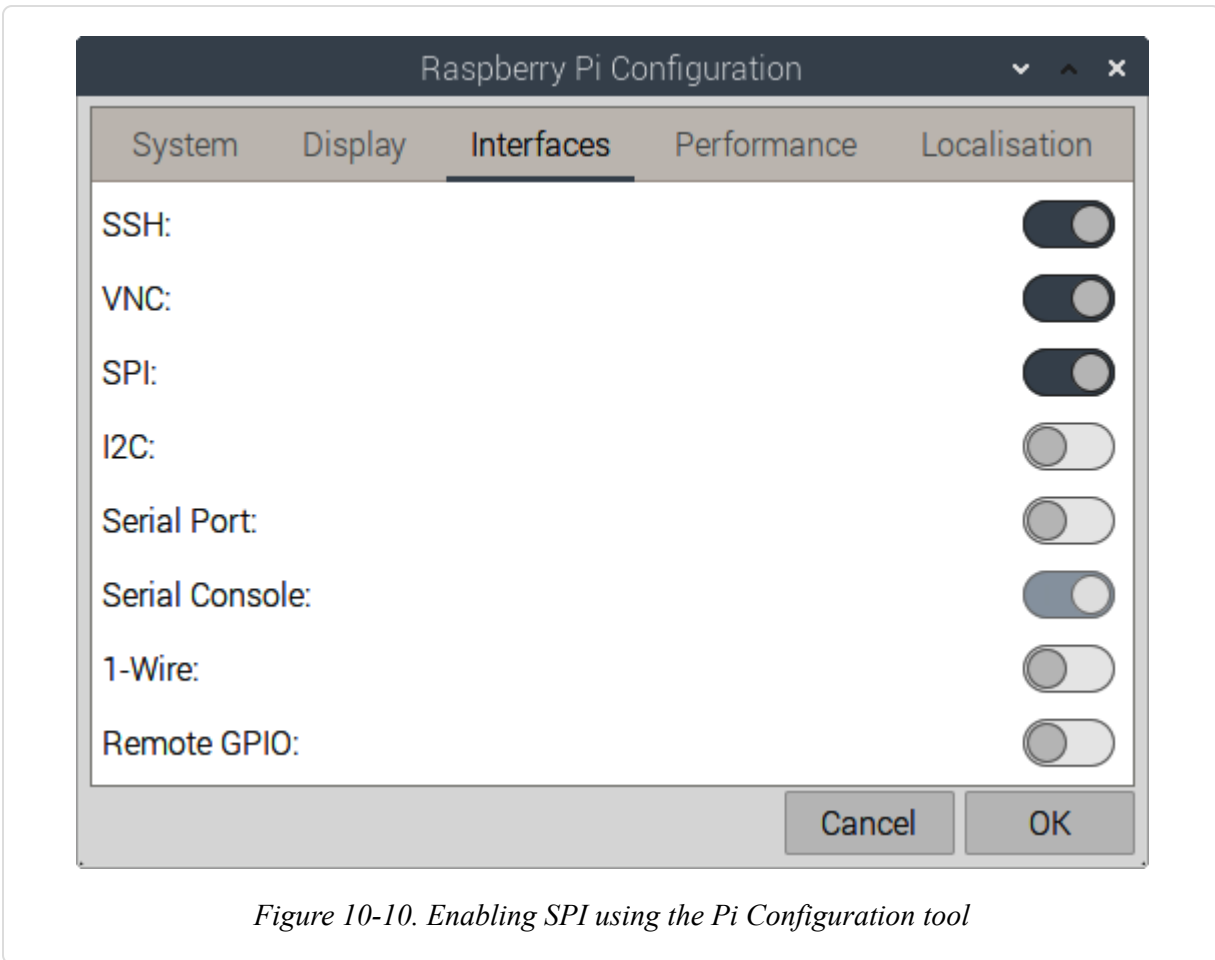
## 10.6 Setting Up SPI

### Problem

You have a device that uses the Serial Peripheral Interface (SPI) bus, and you want to use the device with your Raspberry Pi.

### Solution

By default, SPI is disabled in Raspberry Pi OS. To enable it, the procedure is almost the same as [Recipe 10.4](#). On the Raspberry Menu, under Preferences, open the Raspberry Pi Configuration tool ([Figure 10-10](#)). Go to the Interfaces tab, click the toggle switch for SPI, and click OK.



On older versions of Raspberry Pi OS, or if you prefer the command line, use the `raspi-config` utility:

```
$ sudo raspi-config
```

Select Interfacing Options, followed by SPI, and then respond **Yes** before rebooting your Raspberry Pi. After the reboot, SPI will be available.

## Discussion

SPI allows serial transfer of data between the Raspberry Pi and peripheral devices, such as ADCs and port expander chips (for extra GPIO pins), among other devices.

You can check that SPI is working with the following command:

```
$ ls /dev/*spi*  
/dev/spidev0.0 /dev/spidev0.1
```

If, instead of `spidev0.0` and `spidev0.1` being reported, nothing appears, it means that SPI is not enabled.

## See Also

We use an SPI analog-to-digital converter chip in [Recipe 14.7](#).

## 10.7 Installing pySerial for Access to the Serial Port from Python

### Problem

You want to use the serial port receive and transmit (RXD and TXD pins) on the Raspberry Pi using Python.

### Solution

First, enable the serial port using [Recipe 2.6](#).

Then, install the `pyserial` library:

```
$ sudo pip3 install pyserial
```

### Discussion

The library is pretty easy to use. Create a connection by using the following syntax:

```
ser = serial.Serial(DEVICE, BAUD)
```



DEVICE is the device for the serial port (`/dev/serial0`) and BAUD is the baud rate as a number, not a string. The RXD and TXD pins on the GPIO connector are mapped to the Linux device `/dev/serial0`, and 9600 is the closest there is to a standard baud rate and is used by many devices:

```
ser = serial.Serial('/dev/serial0', 9600)
```

After a connection is established, you can send data over serial like this:

```
ser.write('some text')
```

Listening for a response normally involves a loop that reads and prints, as illustrated in this example:

```
while True:
    print(ser.read())
```

## See Also

You will need to use this technique in recipes that connect hardware to the serial port, such as [Recipe 13.10](#).

# 10.8 Installing Minicom to Test the Serial Port

## Problem

You want to send and receive serial commands from a Terminal session.

## Solution

Install Minicom:

```
$ sudo apt install minicom
```

After Minicom is installed, you can start a serial communication session with a serial device connected to the RXD and TXD pins of the GPIO connector using this command:

```
$ minicom -b 9600 -o -D /dev/serial0
```

The parameter after `-b` is the baud rate, and the parameter after `-D` is the serial port. Be sure to use the same baud rate as the one on the device you are communicating with.

This will start a Minicom session. A peculiarity of the (very old) Minicom standard is that nothing appears on the screen when you type. So, one of the first things you want to do is turn on `local Echo` so that you can see the commands as you type. To do this, press `Ctrl-A` and then `Z`; you'll see the command list shown in [Figure 10-11](#). Press `E` to turn on `local Echo`.

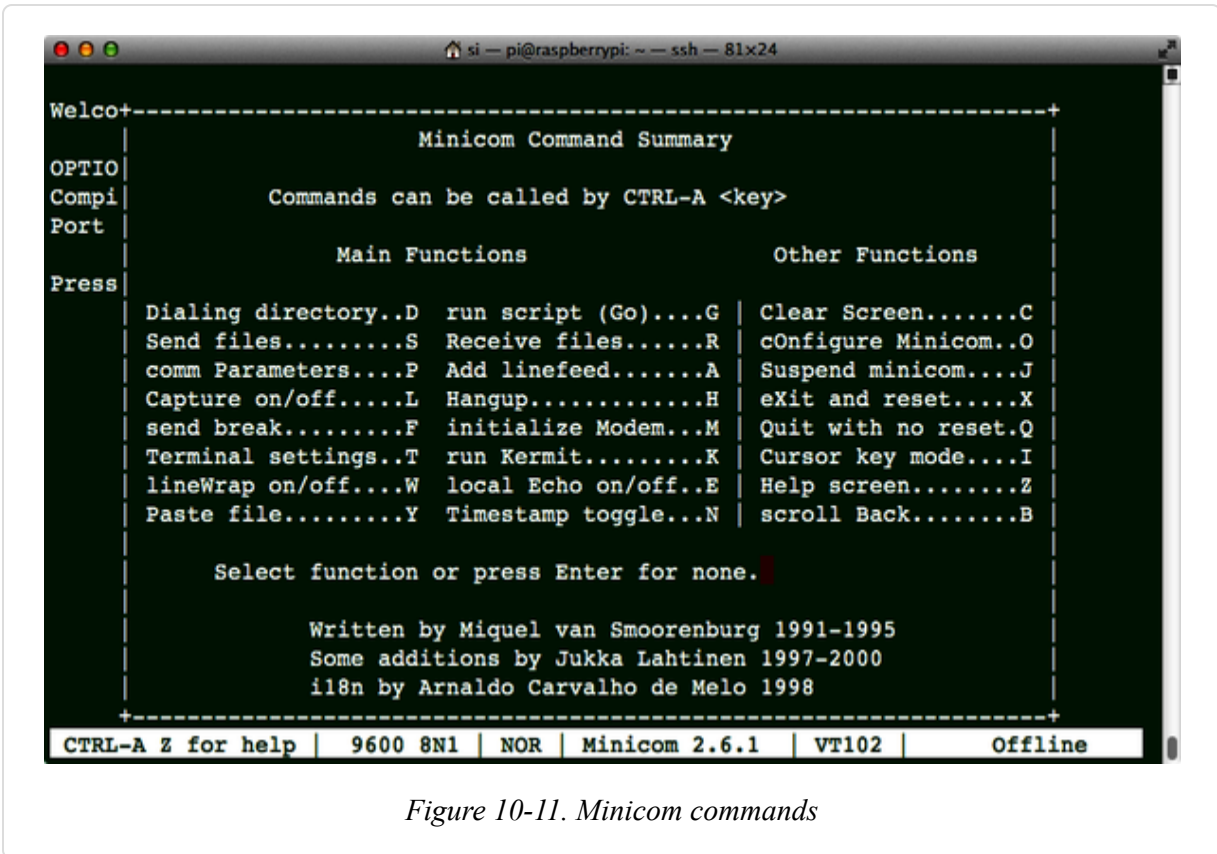


Figure 10-11. Minicom commands

Now, anything you type will be sent to the serial device, and all messages coming from the device will also be displayed.

## Discussion

Minicom is a great tool for checking out the messages coming from a serial device or for making sure that it's working.

## See Also

Check out the [Minicom documentation](#).

If you want to write a Python program to handle the serial communications, you will need the Python `pyserial` library ([Recipe 10.7](#)).

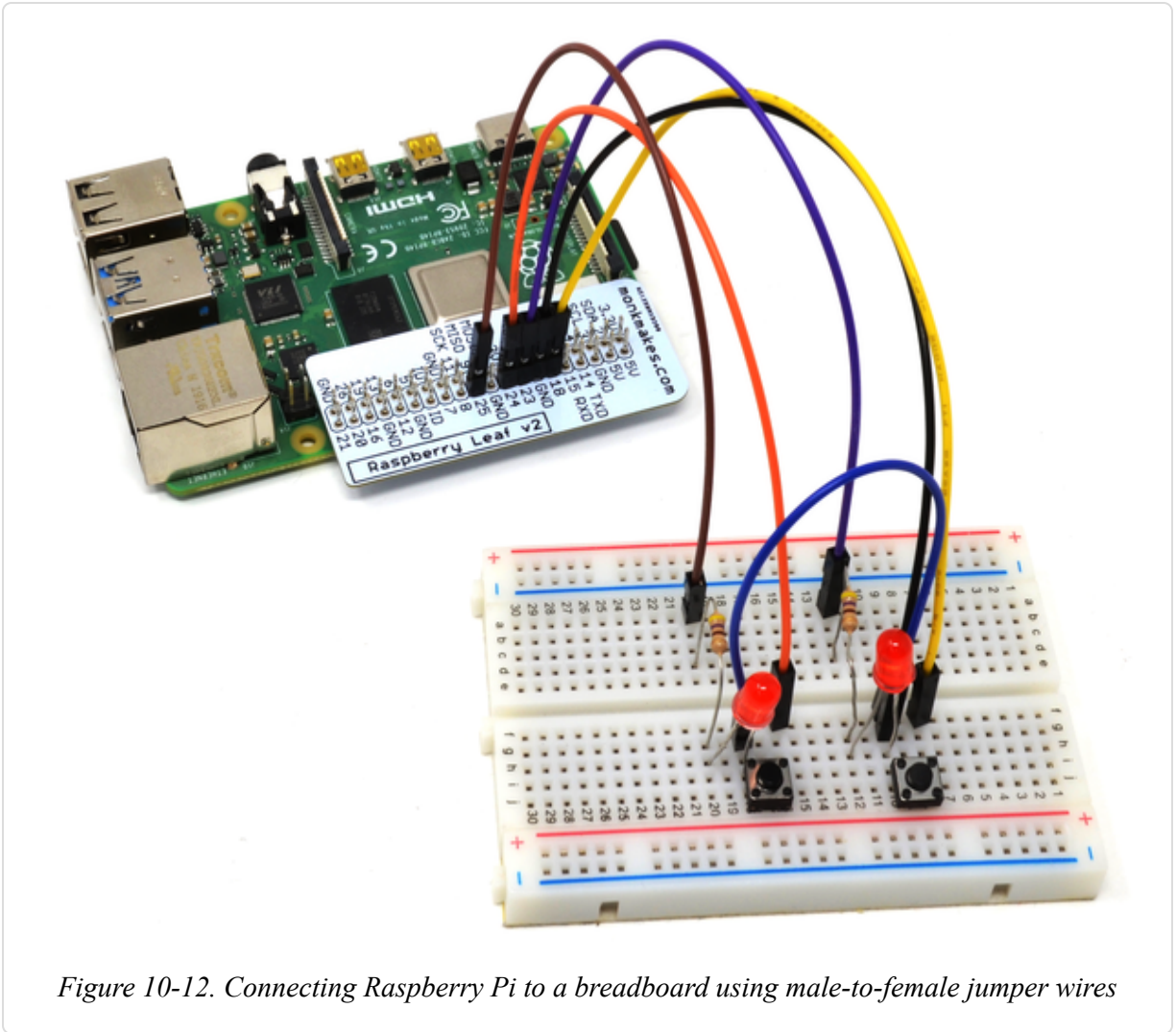
# 10.9 Using a Breadboard with Jumper Leads

## Problem

You want to do some electronic prototyping using your Raspberry Pi and a solderless breadboard.

## Solution

Use male-to-female jumper wires and a GPIO pin label template like the Raspberry Leaf ([Figure 10-12](#)).



*Figure 10-12. Connecting Raspberry Pi to a breadboard using male-to-female jumper wires*

## **Discussion**

It's not always easy to identify the pins that you want on a bare Raspberry Pi board. You can greatly simplify this by printing out a template, like the Raspberry Leaf, to fit over the pins.

It's also useful to have a selection of male-to-male jumper wires for making connections from one part of the breadboard to another.

Female-to-female jumper wires are useful for connecting modules with male header pins directly to the Raspberry Pi, when no other components might warrant the use of a breadboard.

A good way of getting a breadboard, Raspberry Leaf, and set of jumper wires is to buy a starter kit based around a breadboard, like the [Project Box 1 kit for Raspberry Pi from MonkMakes](#).

## See Also

We fully discuss an example of connecting an LED in [Recipe 11.1](#).

## 10.10 Using a Raspberry Squid

### Problem

You want to connect an RGB LED to your Raspberry Pi without having to build something on a breadboard.

### Solution

Use a Raspberry Squid RGB LED ([Figure 10-13](#)).

The Raspberry Squid is an RGB LED with built-in series resistors and female header leads; thus, it can be plugged directly onto the GPIO pins of a Raspberry Pi. The Squid has color-coded leads. The black lead goes to one of the GPIO GND pins, and the red, green, and blue leads go to GPIO pins used for the red, green, and blue channels. The red, green, and blue outputs can be simple digital outputs or pulse-width modulation (PWM) outputs ([Recipe 11.3](#)) that allow you to mix different colors.

You can find [instructions for making your own Squid](#), but you can also buy a [ready-made Squid](#).

The `gpiozero` Python library comes pre-installed on Raspberry Pi OS and has support for RGB LEDs like the Squid.

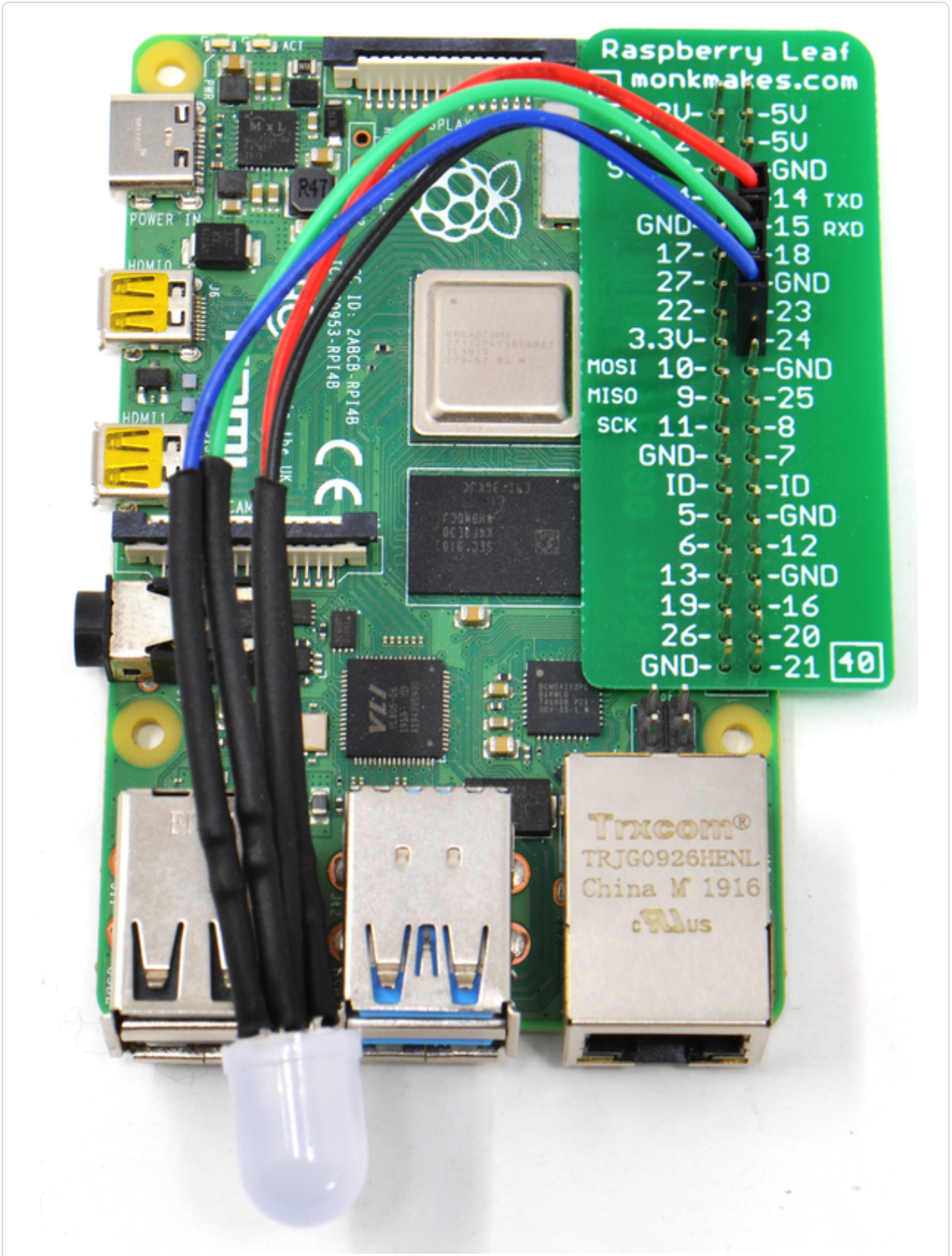


Figure 10-13. The Raspberry Squid

As with all the program examples in this book, you can download this program (see [Recipe 3.22](#)). The file is called `ch_10_squid_test.py`. This program tells you pretty much all you need to know about using the Raspberry Squid:

```
from gpiozero import RGBLED
from time import sleep
from colorzero import Color

led = RGBLED(18, 23, 24)
led.color = Color('red')
sleep(2)
led.color = Color('green')
sleep(2)
led.color = Color('blue')
sleep(2)
led.color = Color('white')
sleep(2)
```

Having imported the various modules you need, you can create a new `RGBLED` object, supplying the three pins to be used for its red, green, and blue channels (in this case, 18, 23, and 24). You can then set the color by using the `led.color = command`, which expects a color.

The color is supplied using the `Color` class from the `colorzero` module. This enables you to specify a color by name, as we did here (most work), or by specifying the separate red, green, and blue color values. For example, the following would set the LED to red:

```
led.color = Color(255, 0, 0)
```

After the color is set, `time.sleep(2)` is used to create a two-second delay before the next color change.

## Discussion

You do not need to use all three color channels of a Squid, and it can be quite handy to just check that a GPIO pin is turning on and off as you

expect before attaching some other electronics to it.

## See Also

For information on the Squid Button, see [Recipe 10.11](#).

[Recipe 11.11](#) is an example project that controls an RGB LED (Squid- or breadboard-based).

## 10.11 Using a Raspberry Squid Button

### Problem

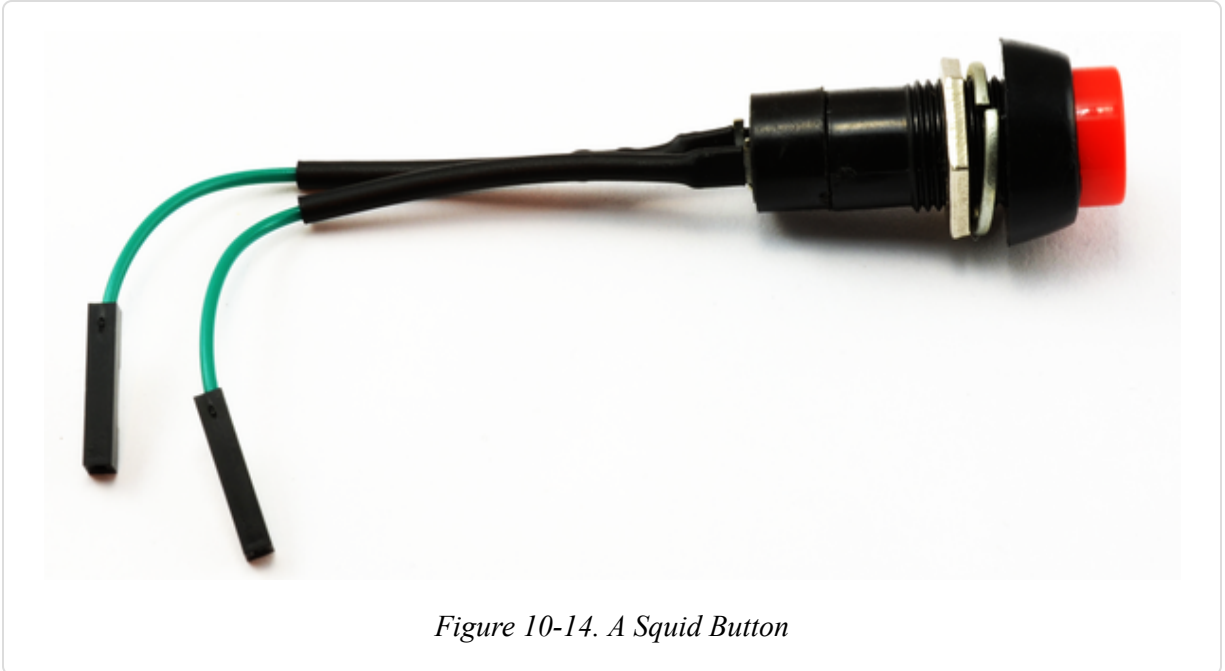
You want to connect a push switch to your Raspberry Pi without having to build something on a breadboard.

### Solution

Use a Squid Button.

The Squid Button ([Figure 10-14](#)) is a push button with female header leads connected to the contacts so that you can plug it directly into the GPIO connector of a Raspberry Pi. The Squid Button also includes a low-value resistor that limits the current that would flow if the Squid Button were to be accidentally connected to a digital output rather than a digital input.





You can use the Squid Button directly with the `gpiozero` library, as the following example shows. As with all the program examples in this book, you can download it (see [Recipe 3.22](#)). The file is called `ch_10_button_test.py`:

```
from gpiozero import Button
import time

button = Button(7)

while True:
    if button.is_pressed:
        print(time.time())
```

The number (in this case, 7) indicates the GPIO pin that the button is connected to. The other pin is connected to GND.

When the button is pressed, the timestamp in seconds is printed.

### Discussion

The Squid Button is useful for testing projects that use a digital input, but because the button is suitable for panel mounting, you can also build it into

more permanent projects.

## See Also

For more information on using switches, see Recipes [13.1](#) through [13.6](#).

For information on the Squid RGB LED, see [Recipe 10.10](#).

## 10.12 Converting 5V Signals to 3.3V with Two Resistors

### Problem

The Raspberry Pi operates at 3.3V, but you want to connect the 5V output of an external module to a GPIO pin on the Pi without damaging it.

### Solution

Use a pair of resistors as a potential divider to reduce the output voltage. [Figure 10-15](#) shows how you might use the 5V serial connection of an Arduino Uno to a Raspberry Pi.

On the Raspberry Pi, GPIO14 is also the TXD pin and GPIO15 is the RXD pin.

To make this recipe, you will need:

- 270 $\Omega$  resistor (see “[Resistors and Capacitors](#)” in [Appendix A](#))
- 470 $\Omega$  resistor (see “[Resistors and Capacitors](#)” in [Appendix A](#))

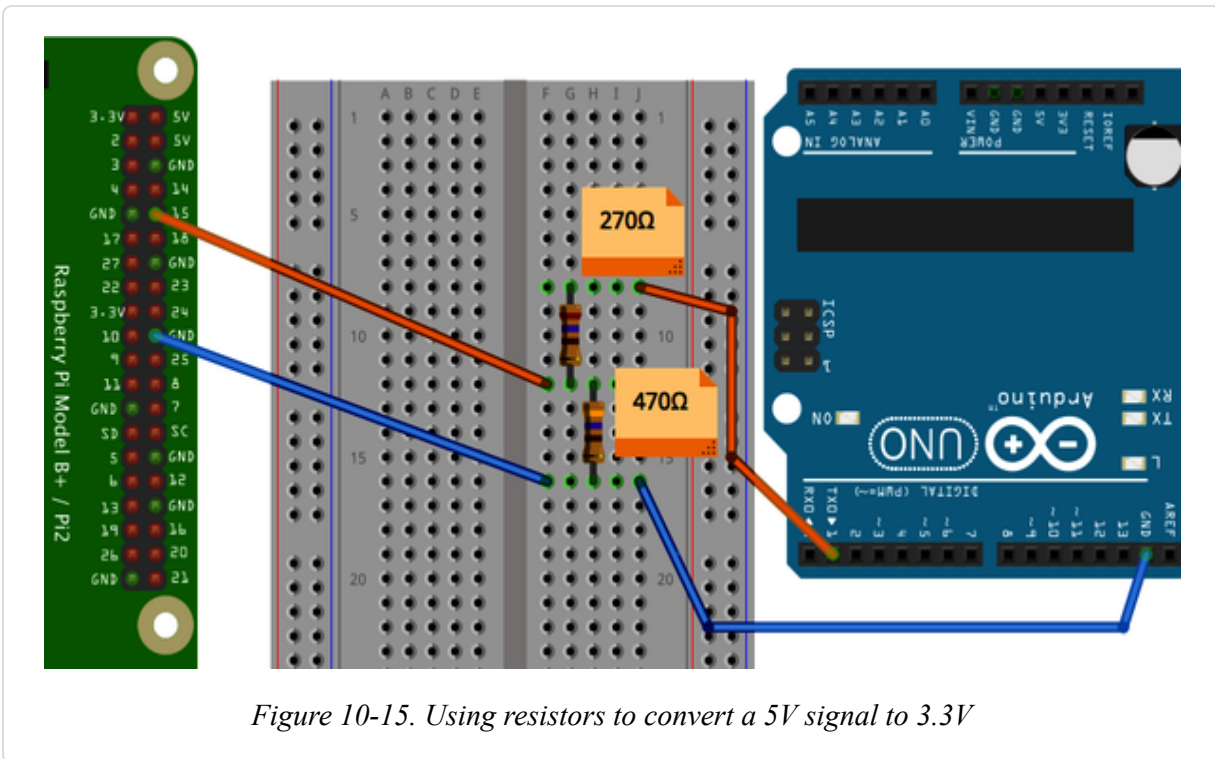


Figure 10-15. Using resistors to convert a 5V signal to 3.3V

The TXD signal from the Pi is a 3.3V output. This can be connected directly to a 5V input on the Arduino without any problem. The Arduino module will recognize anything over about 2.5V as being high.

The problem arises when you need to connect the 5V output of the Arduino module to the RXD pin of the Pi. You *must not* connect this directly to the RXD input—the 5V signal could damage the Pi. Instead, the two resistors shown in [Figure 10-15](#) are used.

## Discussion

The resistors used here will draw a current of 6mA. Given that the Pi uses a fairly hefty 500mA, this will not noticeably affect the current consumption of the Pi.

If you want to minimize the current used by the potential divider, use larger value resistors, scaled proportionally—for example, 27kΩ and 47kΩ, which will draw a miserly 60μA.

## See Also

If you have multiple signals to convert between 3.3V and 5V, it's probably best to use a multichannel level converter module—see [Recipe 10.13](#).

## 10.13 Converting 5V Signals to 3.3V with a Level Converter Module

### Problem

The Raspberry Pi operates at 3.3V. You want to connect a number of 5V digital pins to GPIO pins on the Pi without damaging it.

### Solution

Use a bidirectional level converter module, such as the ones shown in [Figure 10-16](#).

These modules are very easy to use. One side has the power supply at one voltage and a number of channels that can be either inputs or outputs at that voltage. The pins on the other side of the module have a power pin at the second voltage, and all the inputs and outputs on that side are automatically converted to the voltage level for that side.

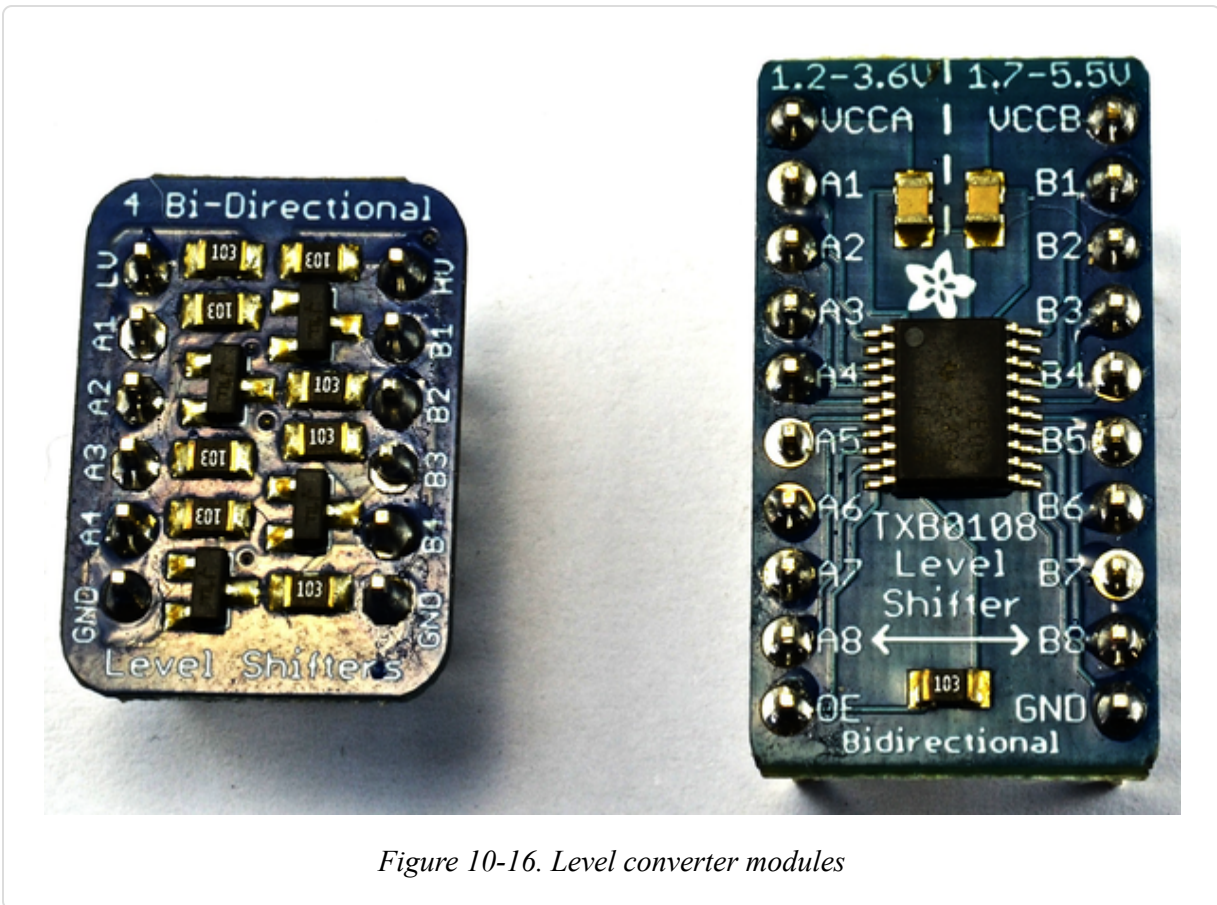


Figure 10-16. Level converter modules

## Discussion

These level converters are available with differing numbers of channels. The two shown in [Figure 10-16](#) have four and eight channels.

You can find sources for such level converters in [Appendix A](#).

## See Also

See [Recipe 10.12](#), especially if you have only one or two levels to convert.

Normally 5V logic inputs will accept 3.3V outputs without a problem; however, in some instances, such as when using LED strips ([Recipe 15.5](#)), this might not be the case, and thus you could use one of the just-described modules to raise the logic level.

## 10.14 Powering a Raspberry Pi with a LiPo Battery

### Problem

You want to power your Raspberry Pi from a 3.7V lithium-ion polymer (LiPo) battery.

### Solution

Use a boost regulator module ([Figure 10-17](#)). The module shown is from SparkFun, but similar, less-expensive designs are available on eBay.

As always with such low-cost purchases from eBay, you should test the module thoroughly before using it. They do not always work exactly as advertised, and quality can be quite variable.

The advantage of this kind of module is that it acts as a voltage regulator to supply 5V to the Pi and also has a USB socket of its own to supply power to its charging circuit. If you plug the Pi's power adapter into the socket on the charger, the Pi will be powered and the battery charged, allowing you to unplug the USB power and use the Pi on the battery for as long as it has enough power.

With a 1300mA LiPo battery, you can expect the Pi to be powered for two or three hours.

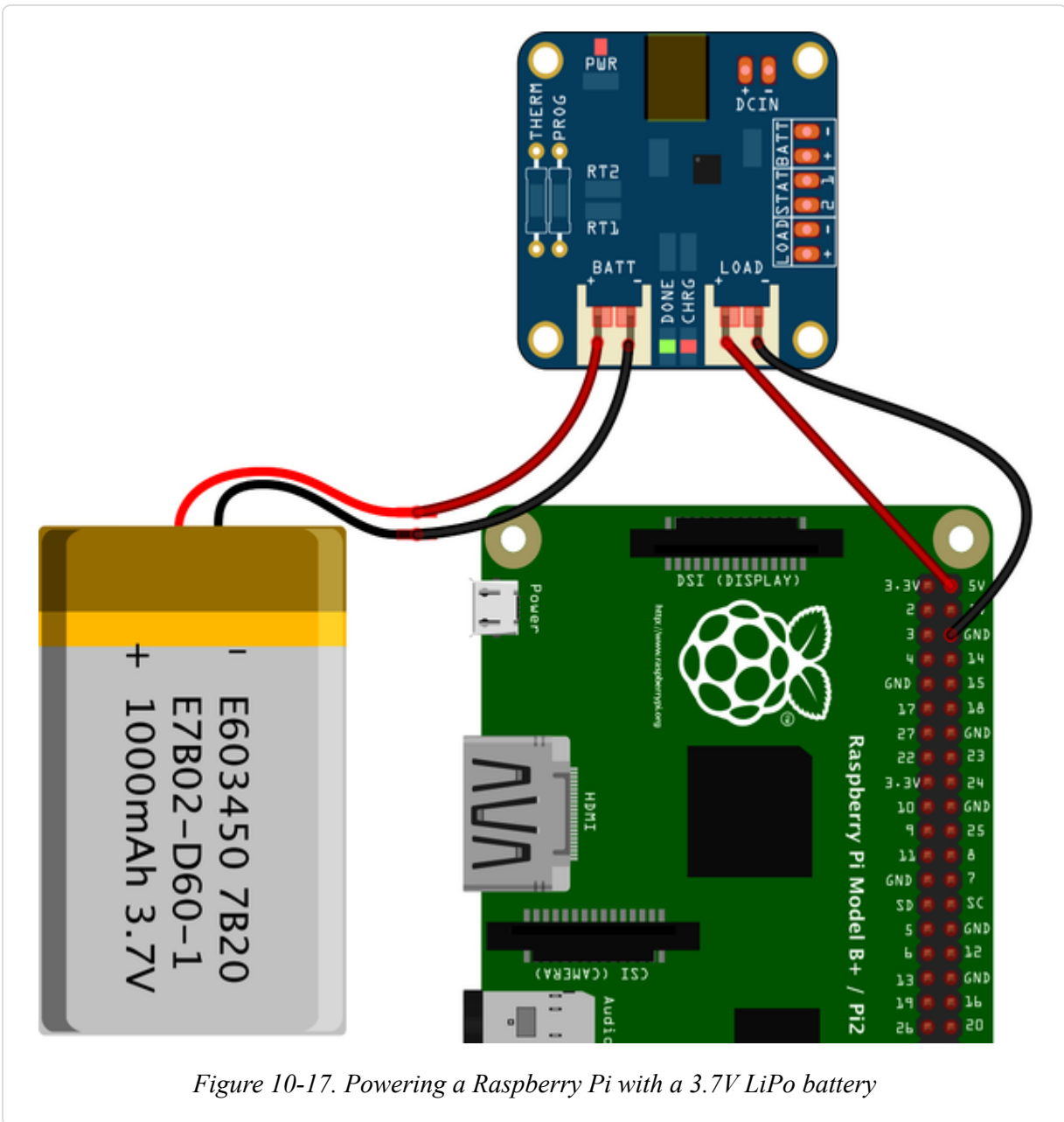


Figure 10-17. Powering a Raspberry Pi with a 3.7V LiPo battery

## Discussion

If you plan to handle the charging of the battery elsewhere, you can just go for a boost converter module, without the charger, at a lower cost.

Battery power is far more practical for earlier versions of Raspberry Pi such as models 2 and 3, but the 4 uses a lot more current (a 3A power supply is recommended).

Alternatively there are LiPo battery-based portable 5V USB battery packs that can be used to power your Raspberry Pi and effectively replicate the previously described approach, but as a cased consumer product. But make sure you get one that can supply enough current for your model of Raspberry Pi (see [Recipe 1.4](#)).

## See Also

Find out more about SparkFun's charger booster module at <https://oreil.ly/UoXQm>.

## 10.15 Getting Started with the Sense HAT

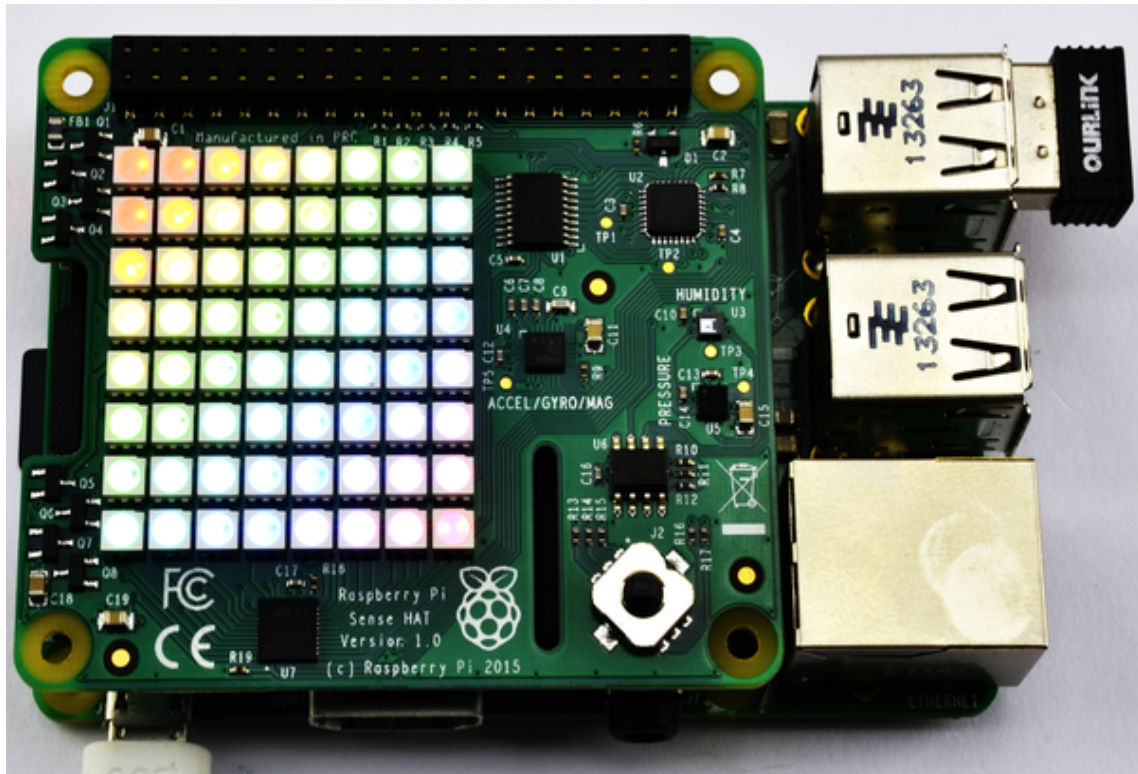
### Problem

You want to know how to use a Raspberry Pi Sense HAT.

### Solution

The Raspberry Pi Sense HAT ([Figure 10-18](#)) is a useful and somewhat confusingly named interface board for the Raspberry Pi. Yes, it includes sensors—in fact, it can measure temperature, relative humidity, and atmospheric pressure ([Recipe 14.12](#)). It also has an accelerometer, a gyroscope ([Recipe 14.16](#)), and a magnetometer ([Recipe 14.15](#)) for navigation-type projects. It also has a full-color 8×8 LED matrix display ([Recipe 15.3](#)).





*Figure 10-18. The Raspberry Pi Sense HAT*

Put the Sense HAT onto your Raspberry Pi before powering it up.

Raspberry Pi OS already includes all the software that you need for the Sense HAT. The Sense HAT uses I2C, so you need to follow the usual I2C setup ([Recipe 10.4](#)).

## Discussion

More recipes that use the Sense HAT are in this book, but for now, you can just check that everything is working by using the following to open a Python console:

```
$ sudo python3
```

Then enter the following commands into the console:

```
>>> from sense_hat import SenseHat
>>> hat = SenseHat()
>>> hat.show_message('Raspberry Pi Cookbook')
```

The message “Raspberry Pi Cookbook” should then scroll across the screen of the LED matrix.

## See Also

See the programming reference for the [Sense HAT](#).

To measure temperature, humidity, and atmospheric pressure, see [Recipe 14.12](#).

To use the Sense HAT’s accelerometer and gyroscope, see [Recipe 14.16](#).

To use the magnetometer to detect north and detect the presence of a magnet, see [Recipes 14.15](#) and [14.18](#), respectively.

## 10.16 Getting Started with the Explorer HAT Pro

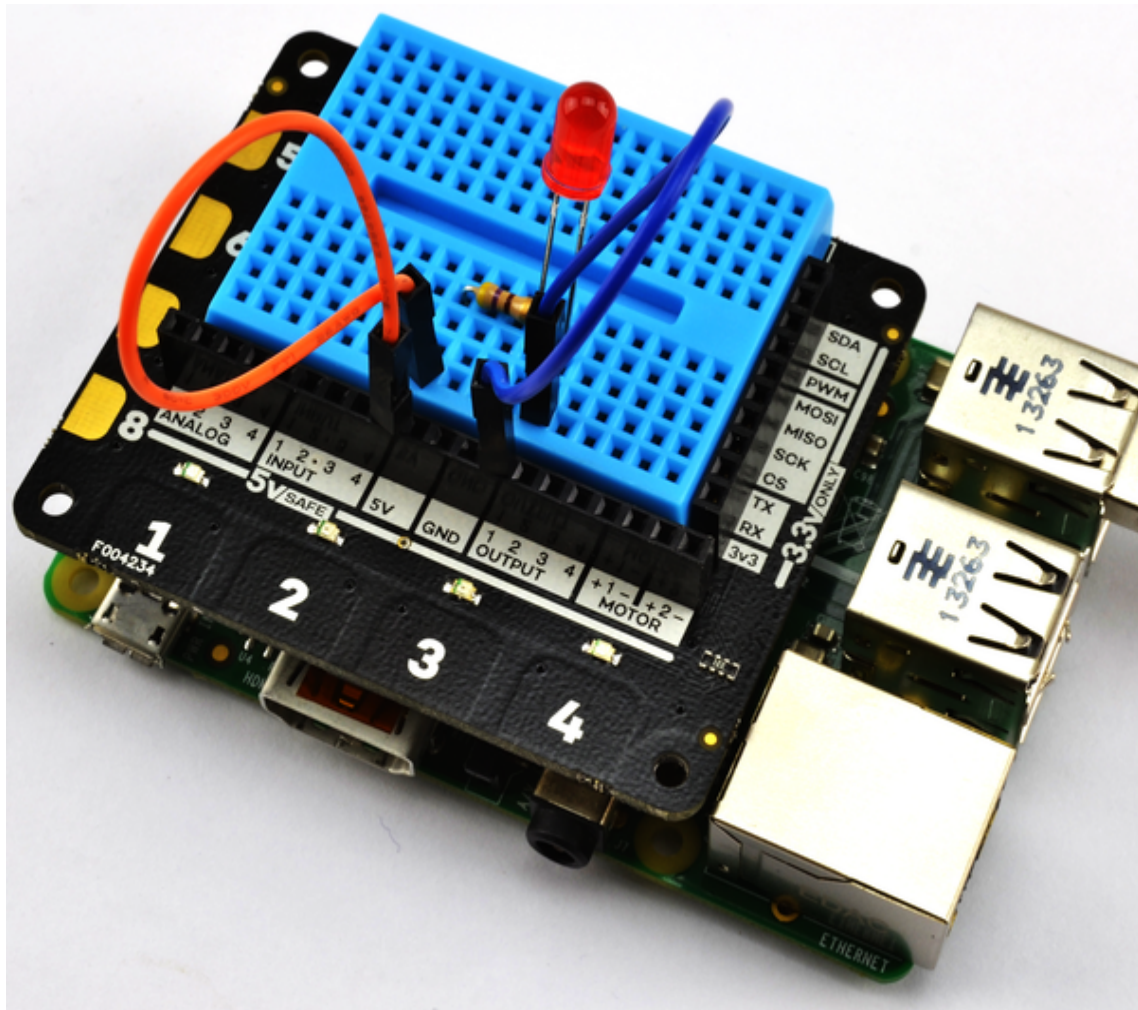
### Problem

You want to know how to get started with the Pimoroni Explorer HAT Pro.

### Solution

Plug the HAT into your Raspberry Pi and install the `explorerhat Pro` Python library.

[Figure 10-19](#) shows a Pimoroni Explorer HAT Pro on a Raspberry Pi B+.



*Figure 10-19. A Pimoroni Explorer HAT Pro*

The Explorer HAT Pro has some useful input/output options as well as an area where a small solderless breadboard can be attached. Some of its features are:

- 4 LEDs
- 4 buffered inputs
- 4 buffered outputs (up to 500mA)
- 4 analog inputs
- 2 low-power motor drivers (max 200mA)
- 4 capacitive touch pads
- 4 capacitive crocodile clip pads

Here's a little experiment you can try that makes the built-in red LED blink. Open an editor and paste in the following code:

```
import explorerhat, time

while True:
    explorerhat.light.red.on()
    time.sleep(0.5)
    explorerhat.light.red.off()
    time.sleep(0.5)
```

As with all the program examples in this book, you can download this program (see [Recipe 3.22](#)). The file is called *ch\_10\_explorer\_hat\_blink.py*.

## Discussion

The Explorer HAT Pro provides four buffered inputs and outputs—that is, inputs and outputs that are not connected directly to the Raspberry Pi but instead are connected to chips on the Explorer HAT Pro. This means that if you accidentally connect things incorrectly, the Explorer HAT Pro will be damaged rather than your Raspberry Pi.

## See Also

You can use the Explorer HAT Pro for capacitive touch sensing ([Recipe 14.21](#)).

## 10.17 Making a HAT

### Problem

You want to create a prototype Raspberry Pi interface board that conforms to the HAT standard.

### Solution

Use a Perma-Proto Pi HAT (Figure 10-20).

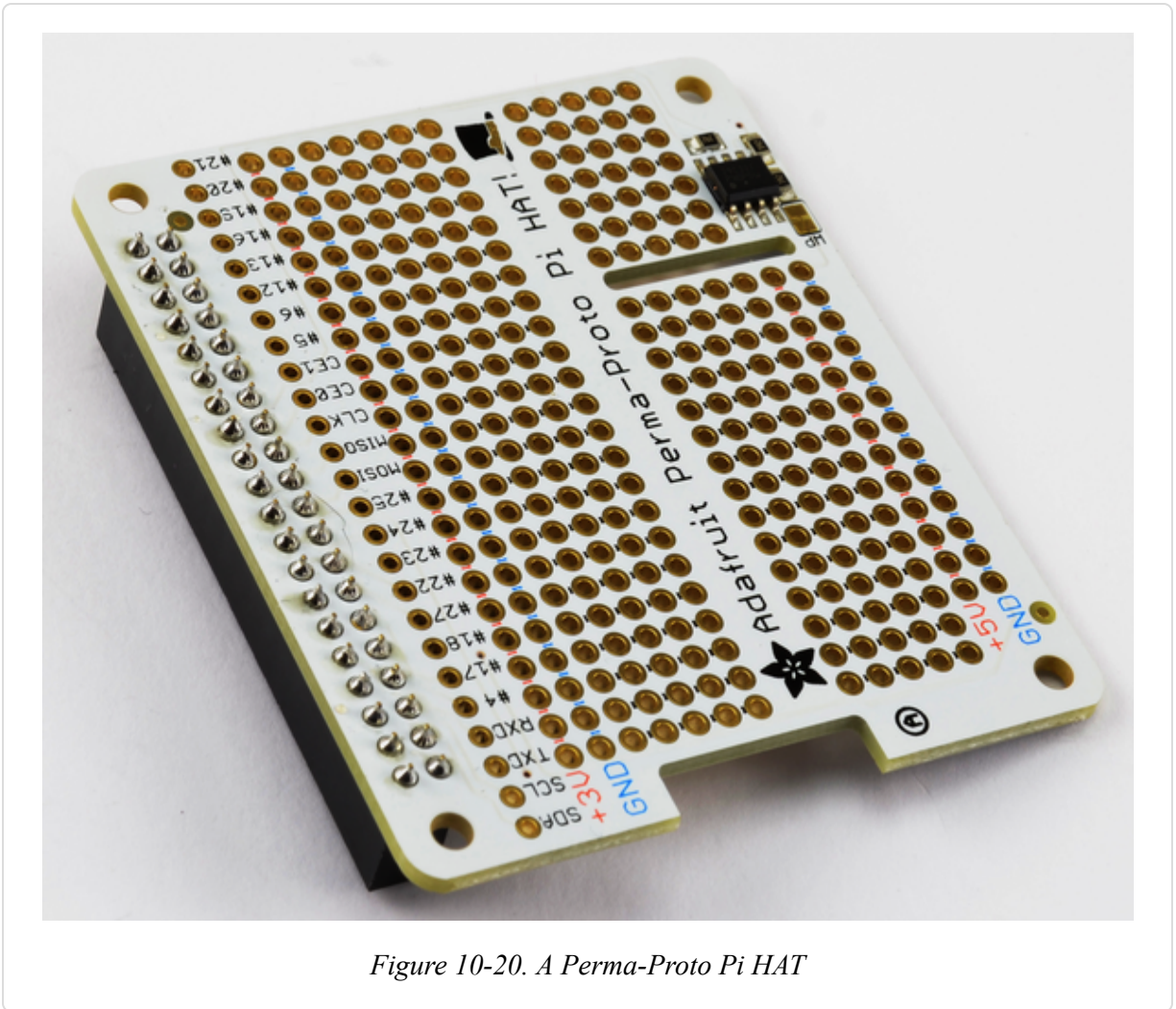


Figure 10-20. A Perma-Proto Pi HAT

With the advent of the Raspberry Pi B+ with a 40-pin GPIO header, a new standard for add-on boards for the Raspberry Pi was defined, called HAT (hardware attached on top). You do not need to stick to this standard, especially if you are just making a one-off product for yourself, but if you are designing a product to sell, it might make sense for you to conform to the HAT standard.

The HAT standard defines the size and shape of the PCB (printed circuit board) and also mandates that the PCB have an EEPROM chip soldered onto the board. This chip is connected to the ID\_SD and ID\_SC pins of the GPIO header and in the future will allow some configuration of the Pi and

even automatic loading of software to occur when the Raspberry Pi is booted up with a HAT attached.

The prototyping area of the board is made up of a breadboard format layout of two rows of five holes plus power rails down both sides of the board.

If you don't care about programming the EEPROM, you can stop here. However, if you want to add your own custom information onto the HAT's EEPROM, read on to the Discussion.

## Discussion

The HAT standard makes a lot of sense. However, as of this writing, Raspberry Pi OS does not make use of any information written in the HAT's EEPROM. This is likely to change in the future and leads to the exciting possibility of HATs automatically doing things like enabling I2C and installing Python libraries for their hardware, just by being present on the Raspberry Pi.

To write data into the EEPROM, you first need to enable the hidden I2C port used by the ID\_SD and ID\_SC pins, which are used to read and write to the EEPROM. To do that, you will need to edit */boot/config.txt* by adding in or uncommenting the following line:

```
dtoverlay=i2c-vc=on
```

After you do this, reboot your Raspberry Pi; you should then be able to detect that the I2C EEPROM is attached to the I2C bus using `i2c-tools` ([Recipe 10.5](#)):

```
$ i2cdetect -y 0
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50: 50  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

```
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

You can see from the result of the `i2cdetect` command that the EEPROM has an I2C address of 50. Note that the option `-y 0`, rather than the usual `-y 1`, is used because this is not the normal I2C bus on pins 2 and 3, but rather the I2C bus dedicated to the HAT EEPROM.

To read and write the EEPROM, you need to download some tools using the following commands:

```
$ git clone https://github.com/raspberrypi/hats.git
$ cd hats/epromutils
$ make
```

Writing to the EEPROM is a three-step process. First, you must edit the file `eprom_settings.txt`. Change at least the `product_id`, `product_version`, `vendor`, and `product` fields to be your company name and product name. Note that lots of other options are in this file, which is well documented. They include specifying back-powering options, GPIO pins used, and so on. Second, after editing the file, run the following command to convert the text file into a file suitable for writing to the EEPROM (`rom_file.eep`):

```
$ ./eepmake eprom_settings.txt rom_file.eep
Opening file eprom_settings.txt for read
UUID=7aa8b587-9c11-4177-bf14-00e601c5025e
Done reading
Writing out...
Done.
```

Finally, copy `rom_file.eep` onto the EEPROM by running the following command:

```
sudo ./eepflash.sh -w -f=rom_file.eep -t=24c32
This will disable the camera so you will need to REBOOT after
this...
This will attempt to write to i2c address 0x50. Make sure there
```

```
is...
This script comes with ABSOLUTELY no warranty. Continue only if
you...
Do you wish to continue? (yes/no): yes
Writing...
0+1 records in
0+1 records out
127 bytes (127 B) copied, 2.52071 s, 0.1 kB/s
Done.
pi@raspberrypi ~/hats/EEPROMutils $
```

When writing is complete, you can read the ROM back using these commands:

```
$ sudo ./eepflash.sh -r -f=read_back.eep -t=24c32
$ ./eepdump read_back.eep read_back.txt
$ more read_back.txt
```

## See Also

See the [Raspberry Pi HAT design guide](#).

Many ready-made HATs are on the market, including the Stepper Motor ([Recipe 12.8](#)), Capacitive Touch ([Recipe 14.21](#)), and 16-Channel PWM ([Recipe 12.3](#)) HATs from Adafruit, as well as the Pimoroni Explorer HAT Pro ([Recipe 10.16](#)).

## 10.18 Using the Raspberry Pi Zero 2 and Pi Zero 2 W

### Problem

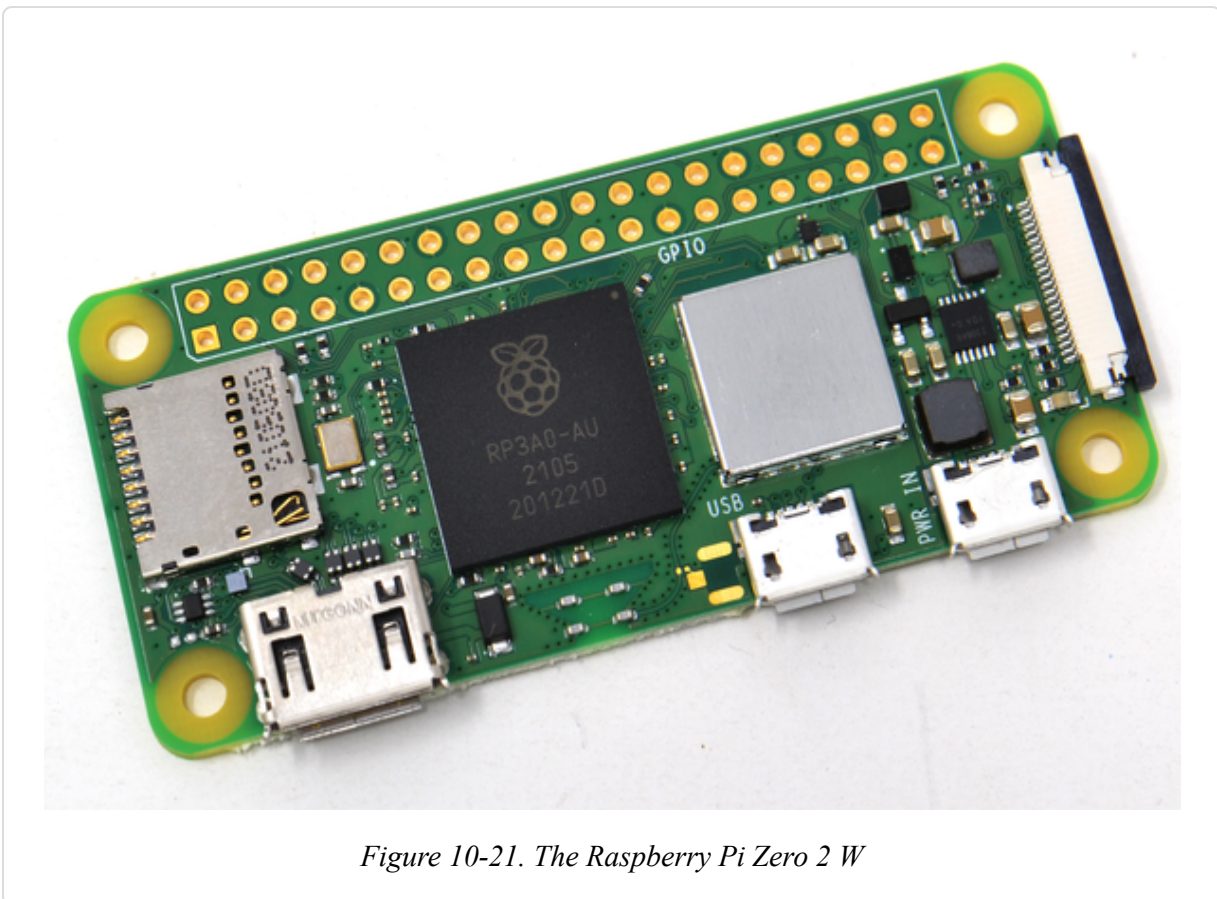
You want to learn more about the Pi Zero 2 and Pi Zero 2 W and how to make use of them in electronics projects.

### Solution



The small size and low cost of the Pi Zero 2 make it the ideal choice for embedding in electronics projects. The Pi Zero 2 W adds WiFi and Bluetooth capabilities to the Pi Zero 2, making it great for small Internet of Things (IoT) projects.

**Figure 10-21** shows a Raspberry Pi Zero 2 W.



*Figure 10-21. The Raspberry Pi Zero 2 W*

The Pi Zero 2 and Pi Zero 2 W are supplied without header pins attached, so your first job is likely to be to solder pins onto it. Suitable header pins are available in Pi Zero starter kits, such as the one supplied by Pi Hut.

It is also possible to buy the Pi Zero 2 W with header pins presoldered, but that is more expensive than the DIY version.

You can also find so-called *hammer* pins that are tight fitting and do not require soldering.

## **Discussion**

With only one USB connector—and a micro-USB OTG (on-the-go) connector at that—you'll need a USB adapter and USB hub to be able to plug in a USB WiFi dongle, keyboard, and mouse in order to set up the Pi Zero 2.

Alternatively, you can use a console cable as described in [Recipe 2.6](#) to set up WiFi by editing `/etc/network/interfaces`, as described in [Recipe 2.5](#). After that is set up, you can connect to the Pi Zero wirelessly using SSH ([Recipe 2.7](#)).

## **See Also**

For a comparison of the Raspberry Pi models available, see [Recipe 1.1](#).

# Chapter 11. Controlling Hardware

---

## 11.0 Introduction

In this chapter, you come to grips with the control of electronics through the Raspberry Pi’s general-purpose input/output (GPIO) connector.

Most of the recipes require the use of a solderless breadboard and male-to-female and male-to-male jumper wires (see [Recipe 10.9](#)). To maintain compatibility with older 26-pin Raspberry Pi models, all the breadboard examples here use only the top 26 pins common to both GPIO layouts (see [Recipe 10.1](#)).

For a kit of parts, and breadboard, that are suitable for many of the recipes in this chapter, take a look at the [Project Box 1 kit for Raspberry Pi](#).

## 11.1 Connecting an LED

### Problem

You want to know how to connect an LED to the Raspberry Pi.

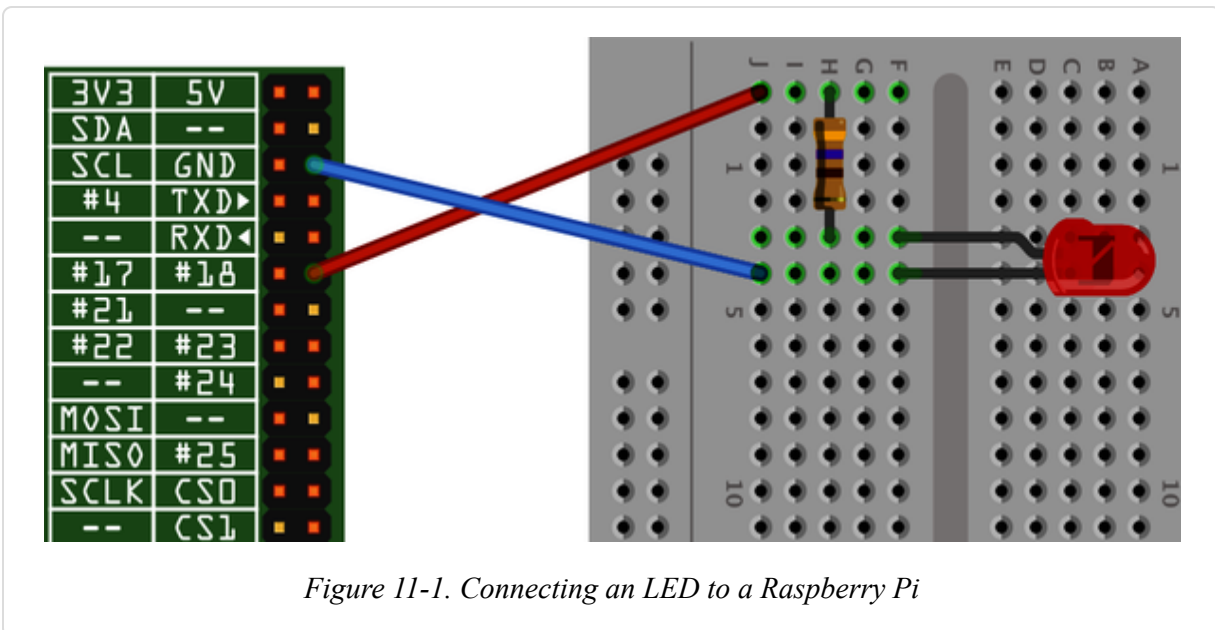
### Solution

Connect an LED to one of the GPIO pins using a 470 $\Omega$  or 1k $\Omega$  series resistor to limit the current. To make this recipe, you will need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- 470 $\Omega$  resistor (see [“Resistors and Capacitors”](#))
- LED (see [“OptoElectronics”](#))

Figure 11-1 shows how you can wire this LED using a solderless breadboard and male-to-female jumper leads. The LED has positive and negative leads. The positive lead is the longer and will be the one on the same row as the resistor in Figure 11-1.

The resistor shown is a 470Ω resistor, which will make the LED shine brightly without drawing so much current that the Raspberry Pi could be damaged.



Having connected the LED, we need to be able to turn it on and off using commands from Python.

Start a Python console from the Terminal and enter these commands:

```
$ sudo python3
>>> from gpiozero import LED
>>> led = LED(18)
>>> led.on()
>>> led.off()
>>>
```

This will turn your LED on after the `led.on()` command, and off again after the `led.off()` command.

## Discussion

LEDs are a very useful, cheap, and efficient way of producing light, but you do have to be careful how you use them. If they are connected directly to a voltage source (such as a GPIO output) that is greater than about 1.7 volts, they will draw a very large current. This can often be enough to destroy the LED or whatever is providing the current—which is not good if your Raspberry Pi is providing the current.

You should always use a current-limiting resistor with an LED. The series resistor is placed *between* the LED and the voltage source, which limits the amount of current flowing through the LED to a level that is safe for both the LED and the GPIO pin driving it.

Raspberry Pi GPIO pins are guaranteed to provide only about 3mA or 16mA of current (depending on the board and number of pins in use)—see [Recipe 10.3](#). LEDs will generally illuminate with any current greater than 1mA, but they will be brighter with more current. Use [Table 11-1](#) as a guide to selecting a series resistor based on the type of LED; the table also indicates the approximate current that will be drawn from the GPIO pin.

*Table 11-1. Selecting series resistors for LEDs and a 3.3V GPIO pin*

LED type	Resistor	Current (mA)
Red	470Ω	3.5
Red	1kΩ	1.5
Orange, yellow, green	470Ω	2
Orange, yellow, green	1kΩ	1
Blue, white	100Ω	3
Blue, white	270Ω	1

As you can see, in all cases it is safe to use a 470Ω resistor. If you are using a blue or white LED, you can reduce the value of the series resistor

considerably without risk of damaging your Raspberry Pi.

If you want to extend the experiments that you made in the Python console into a program that makes the LED blink on and off repeatedly, you could paste the code that you'll find in *ch\_11\_led\_blink.py* into an editor (as with all the program examples in this book, you can download this program [see [Recipe 3.22](#)]):

```
from gpiozero import LED
from time import sleep

led = LED(18)

while True:
    led.on()
    sleep(0.5)
    led.off()
    sleep(0.5)
```

To run the command, enter the following:

```
$ python3 ch_11_led_blink.py
```

The sleep period of 0.5 seconds between turning the LED on and turning it off again makes the LED blink once a second.

The LED class also has a built-in method for blinking, as illustrated by this example:

```
from gpiozero import LED

led = LED(18)
led.blink(0.5, 0.5, background=False)
```

The first two parameters to `blink` are the on time and off time, respectively. The optional `background` parameter is interesting because if you set this to `True`, your program will be able to continue running other commands in the background while the LED is blinking.

When you are ready to stop the LED blinking in the background, you can just use `led.off()`. This technique can greatly simplify your programs. The example in `ch_11_led_blink_2.py` shows this in action:

```
from gpiozero import LED

led = LED(18)
led.blink(0.5, 0.5, background=True)
print("Notice that control has moved away - hit Enter to
continue")
input()
print("Control is now back")
led.off()
input()
```

When the program starts, the LED will be set blinking in the background and the program is free to move onto the next command and print “Notice that control has moved away - hit Enter to continue.” The `input()` command will cause the program to halt and wait for input (you can just press the Enter key). But notice that before you press Enter, the LED is still blinking even though the program has moved on to wait for input.

When you do press Enter again, the `led.off()` command stops the LED’s background blinking.

## See Also

Check out this [series resistor calculator](#).

For more information on using a breadboard and jumper wires with the Raspberry Pi, see [Recipe 10.9](#).

See the [gpiozero documentation on LEDs](#).

## 11.2 Leaving the GPIO Pins in a Safe State

### Problem

You want all the GPIO pins to be set to inputs whenever your program exits to reduce the chance of an accidental short on the GPIO header, which could damage your Raspberry Pi.

## Solution

Whenever you exit a program that uses `gpiozero`, it will automatically set all the GPIO pins into a safe *input* state.

## Discussion

Earlier methods of accessing the GPIO pins, such as the `RPi.GPIO` library, did not automatically set the GPIO pins to be in a safe input state. Instead, they required you to call a `cleanup` function before exiting the program.

If `cleanup` was not called or the Pi was not rebooted, pins set to be outputs would remain as outputs after the program has finished. If you were to start wiring up a new project, unaware of this problem, your new circuit might accidentally short a GPIO output to one of the supply voltages or another GPIO pin in the opposite state. A typical scenario in which this might happen would be if you were to connect a push switch, connecting a GPIO pin that you had configured as an output and HIGH to GND.

Fortunately for us, the `gpiozero` library now takes care of this.

## See Also

For more information on exception handling in Python, see [Recipe 7.10](#).

## 11.3 Controlling the Brightness of an LED

### Problem

You want to vary the brightness of an LED from a Python program.

### Solution



The `gpiozero` library has a pulse-width modulation (PWM) feature that enables you to control the power to an LED and its brightness. To try it out, connect an LED as described in [Recipe 11.1](#) and run this test program (`ch_11_led_brightness.py`):

```
from gpiozero import PWMLED

led = PWMLED(18)

while True:
    brightness_s = input("Enter Brightness (0.0 to 1.0):")
    brightness = float(brightness_s)
    led.value = brightness
```

The program is included in the code download (see [Recipe 3.22](#)).

Run the Python program, and you will be able to change the brightness by entering a number between 0.0 (off) and 1.0 (full brightness):

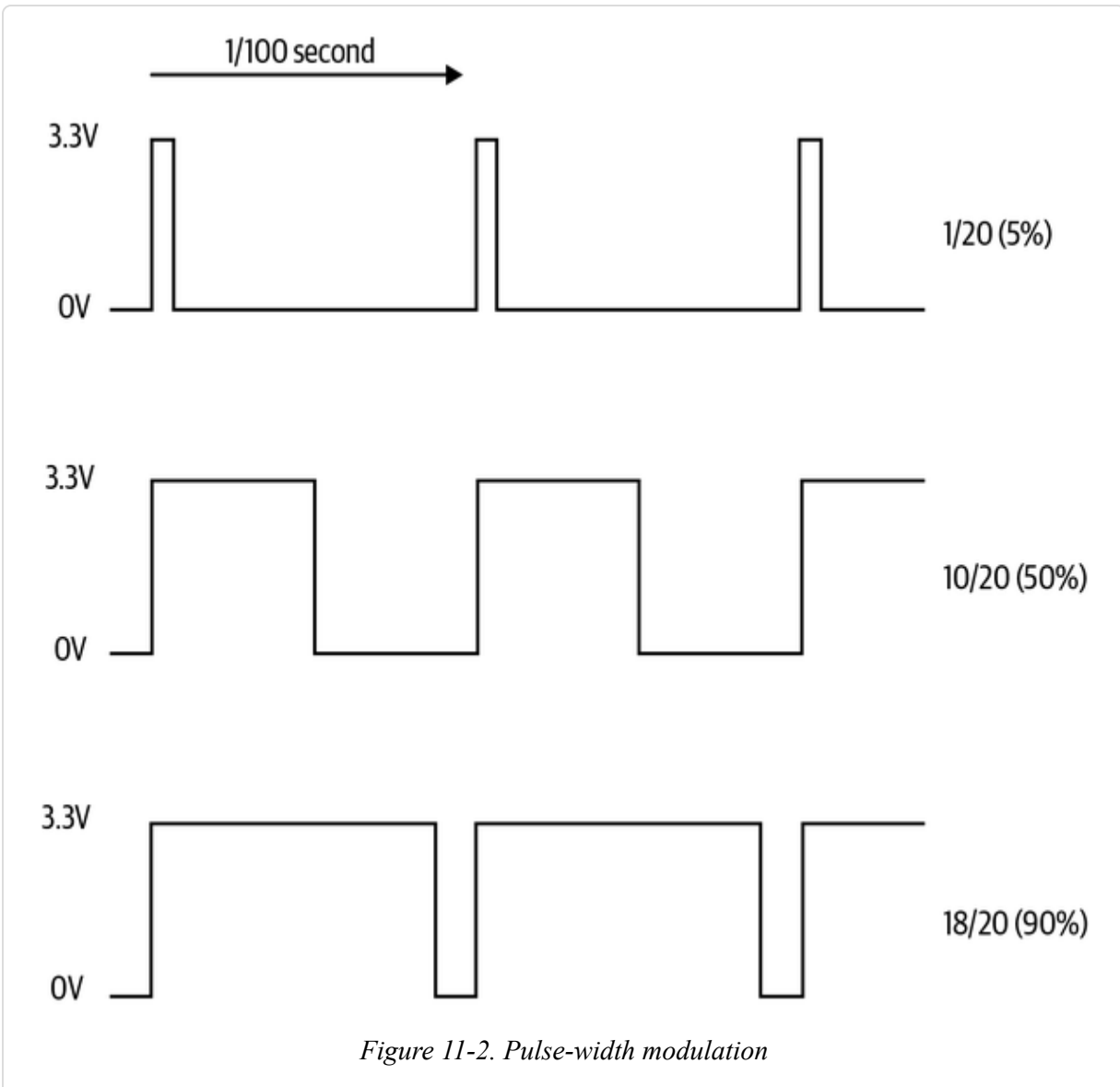
```
$ python ch_11_led_brightness.py
Enter Brightness (0.0 to 1.0):0.5
Enter Brightness (0.0 to 1.0):1
Enter Brightness (0.0 to 1.0):0
```

Exit the program by pressing Ctrl-C. Ctrl-C is command line for *stop what you are doing*; in many situations, it will stop a program entirely.

Note that when controlling an LED's brightness like this, you must define the LED as being `PWMLED` and not just `LED`.

## Discussion

PWM is a clever technique by which you vary the length of pulses while keeping the overall number of pulses per second (the frequency in Hz) constant. [Figure 11-2](#) illustrates the basic principle of PWM.



If the pulses are only high for a short amount of time, the LED will appear dim, whereas if the pulse is high a much higher proportion of the time, the LED will appear brighter.

By default, the PWM frequency is 100Hz; that is, the LED flashes 100 times per second. You can change this where you define `PWMLED` by supplying the optional `frequency` parameter:

```
led = PWMLED(18, frequency=1000)
```

The value is in Hz, so in this case, the frequency is set to 1,000 Hz (1 kHz).

**Table 11-2** compares frequencies specified in the parameter to the actual frequencies on the pin measured with an oscilloscope.

*Table 11-2. Requested frequency  
against measured frequency*

Requested frequency	Measured frequency
50 Hz	50 Hz
100 Hz	98.7 Hz
200 Hz	195 Hz
500 Hz	470 Hz
1 kHz	880 Hz
10 kHz	4.2 kHz

You can see that the frequency becomes less accurate as it increases. This means that this PWM feature is no good for audio (20 Hz to 20 kHz), but plenty fast enough for controlling the brightness of LEDs or the speed of motors. If you want to experiment with this yourself, the program is in the code download and called `ch_11_pwm_f_test.py`.

## See Also

For more information on PWM, see [Wikipedia](#).

**Recipe 11.11** uses PWM to change the color of an RGB LED, and **Recipe 12.4** uses PWM to control the speed of a DC motor.

For more information on using a breadboard and jumper wires with the Raspberry Pi, see **Recipe 10.9**. You can also control the brightness of the LED with a slider control—see **Recipe 11.10**.

## 11.4 Switching a High-Power DC Device Using a Transistor

### Problem

You want to control the current to a high-power, low-voltage DC device such as a 12V LED module.

### Solution

These high-power LEDs use far too much current to power directly from a GPIO pin. They also require 12V rather than 3.3V. To control such a high-power load, you need to use a transistor.

In this case, you'll use a high-power type of transistor called a metal–oxide–semiconductor field-effect transistor (MOSFET), which costs less than a dollar but can handle loads up to 30 amps—many times more than is required for the high-power LEDs. The MOSFET used is a FQP30N06L (see “[Transistors and Diodes](#)”).

**Figure 11-3** shows how you can connect a MOSFET on a breadboard. Make sure that you correctly identify the positive and negative supply leads for the LED module.

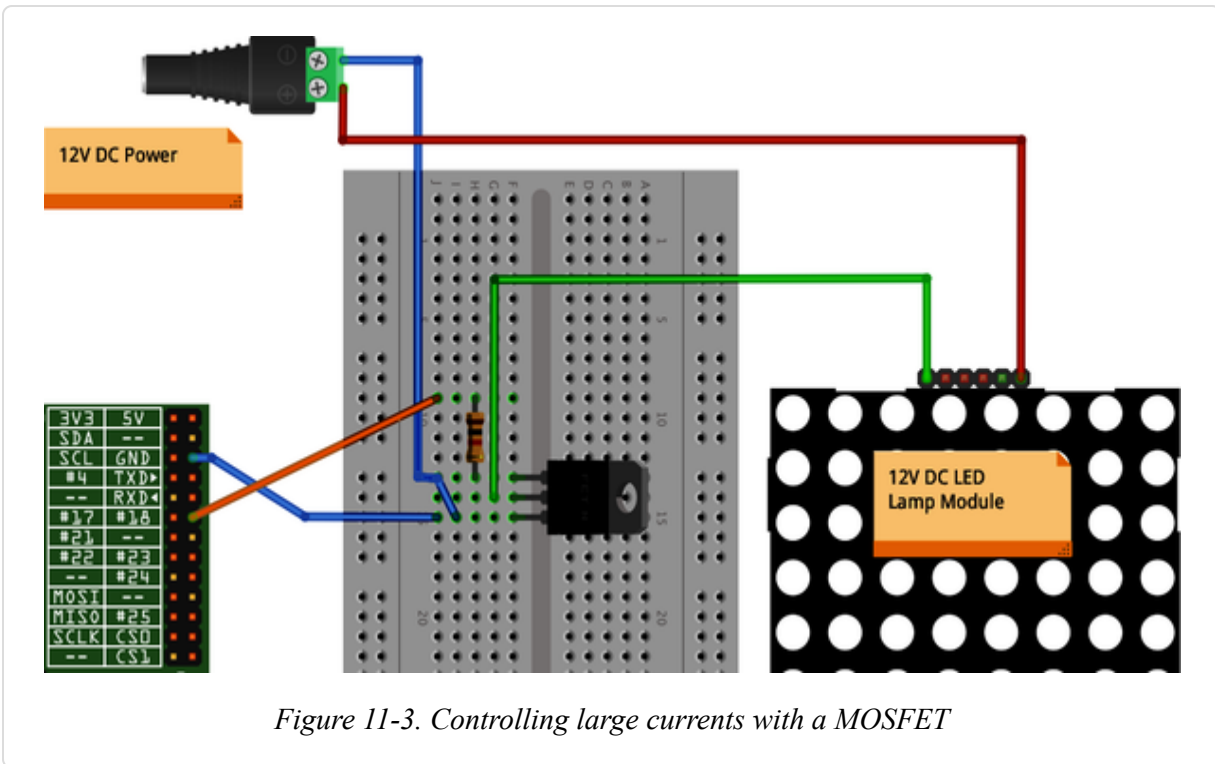


Figure 11-3. Controlling large currents with a MOSFET

To make this recipe, you will need the following:

- Breadboard and jumper wires (see “[Prototyping Equipment and Kits](#)”)
- 1kΩ resistor (see “[Resistors and Capacitors](#)”)
- FQP30N06L N-Channel MOSFET or TIP120 Darlington transistor (see “[Transistors and Diodes](#)”)
- 12V power adapter
- 12V DC LED module

The Python code to turn the LED panel on and off is exactly the same as if we were controlling a single low-power LED without the MOSFET (see [Recipe 11.1](#)).

You can also use PWM with the MOSFET to control the brightness of the LED module (see [Recipe 11.3](#)).

## Discussion

Whenever you need to power anything significant using the GPIO connector, use batteries or an external power adapter. The GPIO connector can supply only relatively low currents ([Recipe 10.3](#)). In this case, you'll use a 12V DC power adapter to provide the power to the LED panel. Pick a power adapter that has sufficient power handling. Thus, if the LED module is 5W, you need at least a 12V 5W power supply (6W would be better). If the power supply specifies a maximum current rather than power, you can calculate its power by multiplying the voltage by the maximum current. For instance, a 500mA 12V power supply can provide 6W of power.

The resistor is necessary to ensure that the peak currents that occur as the MOSFET switches from off to on, and vice versa, do not overload the GPIO pin. The MOSFET switches the negative side of the LED panel, so the positive supply is connected directly to the positive side of the LED panel, and the negative side of the LED panel is connected to the *drain* of the MOSFET. The *source* connection of the MOSFET is connected to GND, and the MOSFET's *gate* pin controls the flow of current from the drain to the source. If gate voltage is above 2V or so, the MOSFET will turn on, and current flows through both it and the LED module.

### **Not Suitable for AC**

*Do not* try to use this circuit for switching 110 or 220V AC. It won't work and is extremely dangerous to try. Instead, use [Recipe 11.7](#).

The MOSFET used here is an FQP30N06L. The L at the end means that it is a logic-level MOSFET whose gate *threshold* voltage is suitable for use with 3.3V digital outputs. The non-L version of this MOSFET is also likely to work just fine, but you can't guarantee that it will, as the specified range of gate threshold voltages is 2V to 4V. Therefore, if you were unlucky and got a MOSFET at the 4V end, it would not switch well.

An alternative to using a MOSFET is to use a power Darlington transistor like the TIP120. This has a pinout compatible with the FQP30N06L, so you can keep the same breadboard layout.

This circuit is suitable for controlling the power to other low-voltage DC devices. The only real exceptions are motors and relays, which require some extra treatment (see [Recipe 11.5](#)).

## See Also

Check out the [datasheet for the MOSFET](#).

If you would like to create a graphical user interface with which to control your LED module, see [Recipe 11.9](#) for a simple on/off control, and [Recipe 11.10](#) for variable control of the brightness with a slider.

## 11.5 Switching a High-Power Device Using a Relay

### Problem

You want to turn devices on and off that might not be suitable for switching with a MOSFET.

### Solution

Use a relay and small transistor.

Using a transistor by itself ([Recipe 11.4](#)) works well for medium loads of a few hundred mA or a bit more. But for higher currents or for situations when the controlling electronics need to be electrically isolated from the device being switched, it is more convenient to use a relay.

[Figure 11-4](#) shows how you can connect a transistor and relay on a breadboard. Make sure that both the transistor and diode are placed the right way. The diode has a stripe at one end, and the transistor used here has one flat side and one curved side.

To make this recipe, you will need the following:

- Breadboard and jumper wires (see “[Prototyping Equipment and Kits](#)”)
- 1k $\Omega$  resistor (see “[Resistors and Capacitors](#)”)
- Transistor 2N3904 (see “[Transistors and Diodes](#)”)
- 1N4001 diode (see “[Transistors and Diodes](#)”)
- 5V relay (see “[Miscellaneous](#)”)
- Multimeter

You can use the same LED blink program that you used in [Recipe 11.1](#). If all is well, you’ll hear a click from the relay and a beep from the multimeter each time the contacts are closed. However, relays are slow mechanical devices, so don’t try to use them with pulse-width modulation (PWM): it can damage the relay.



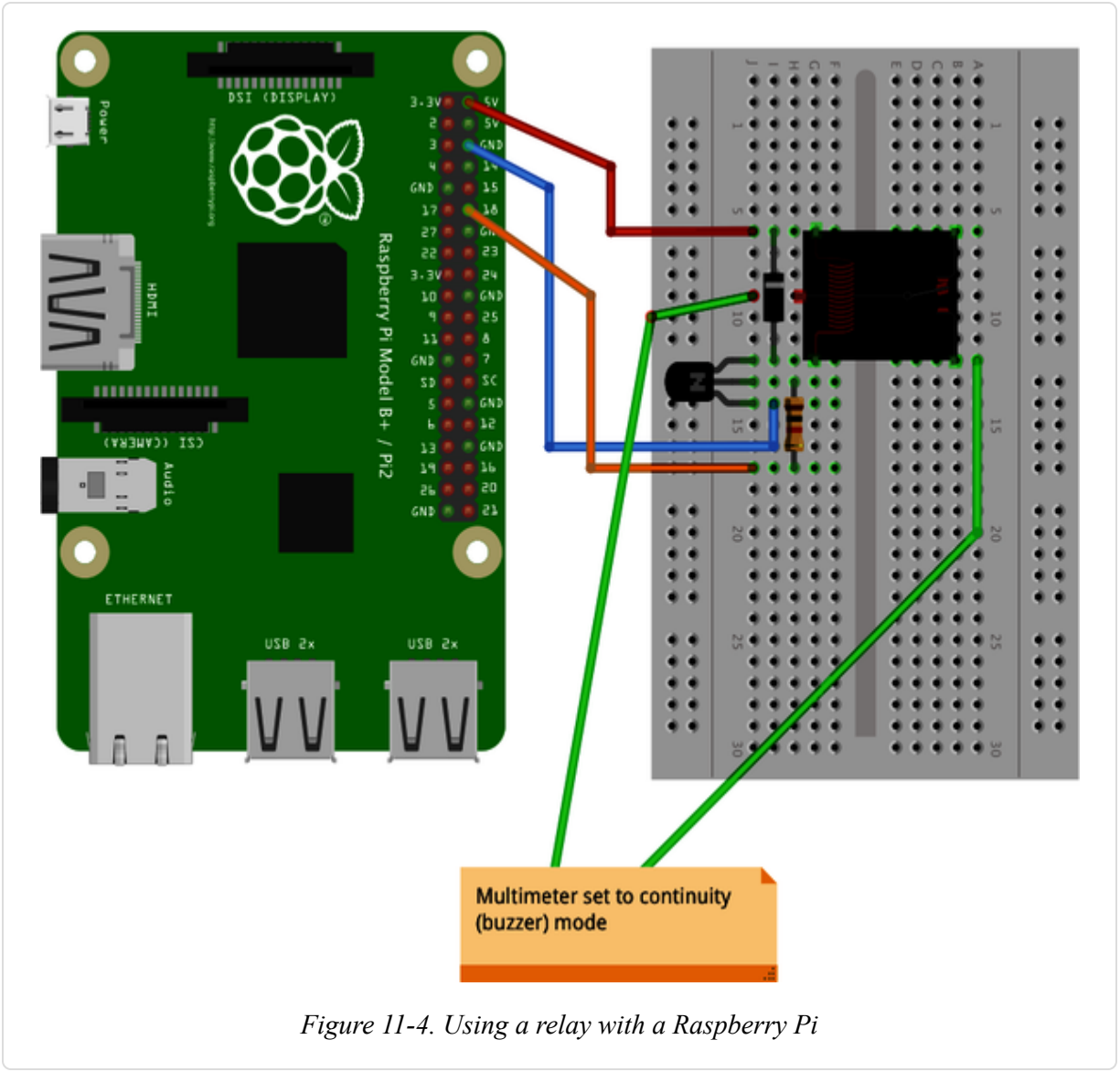


Figure 11-4. Using a relay with a Raspberry Pi

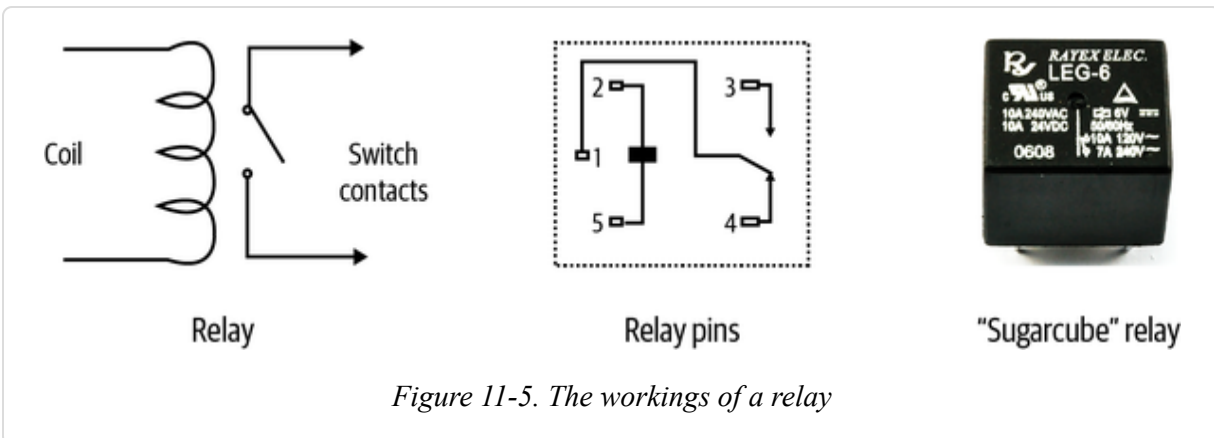
### Discussion

Relays have been around since the early days of electronics and have the great advantage of being easy to use, plus they'll work in any situation in which a switch would normally work—for example, when you're switching AC (alternating current), or in situations for which the exact wiring of the device being switched is unknown.

If the relay contacts are asked to exceed their specifications, the relay's life will be shortened. There will be arcing, and the contacts can eventually fuse

together. There is also the possibility of the relay becoming dangerously hot. When in doubt, overspecify the relay contacts.

**Figure 11-5** shows the schematic symbol, pin layout, and package of a typical relay.



A relay is essentially a switch whose contacts are closed when an electromagnet pulls them together. Because the electromagnet and switch are not connected electrically in any way, this protects the circuit driving the relay coil from any high voltages on the switch side.

The downside of relays is that they are slow to operate and will eventually wear out after many hundreds of thousands of operations. This means they are suitable only for slow on/off control, and not for fast switching like PWM.

The coil of a relay requires about 50mA to close the connection. Because a Raspberry Pi GPIO pin is capable of supplying only about 3mA, you need to use a small transistor as a switch. You don't need to use a high-power MOSFET like you did in **Recipe 11.4**; you can just use a small transistor instead. This has three connections. The base (middle lead) is connected to the GPIO pin via a 1k $\Omega$  resistor to limit the current. The *emitter* is connected to GND, and the *collector* is connected to one side of the relay. The other side of the relay is connected to 5V on the GPIO connector. The diode is used to suppress any high-voltage pulses that occur when the transistor rapidly switches the power to the relay's coil.

## WARNING

Although relays can be used to switch 110V or 220V AC, this voltage is very dangerous and should not be used on a breadboard. If you want to switch high voltages, use [Recipe 11.7](#) instead.

## See Also

For switching direct current (DC) using a power MOSFET, see [Recipe 11.4](#).

# 11.6 Switching Using a Solid-State Relay

## Problem

You want to use a solid-state (no moving parts) relay with your Raspberry Pi.

## Solution

### Not Suitable for High Voltage

The solution detailed here is only suitable for switching low-voltage (less than 16V) AC or DC. *You must not* use this with the domestic 110V or 220V AC supply.

Connect a GPIO pin directly to the input of a solid-state relay (SSR) such as the MonkMakes SSR shown in [Figure 11-6](#).

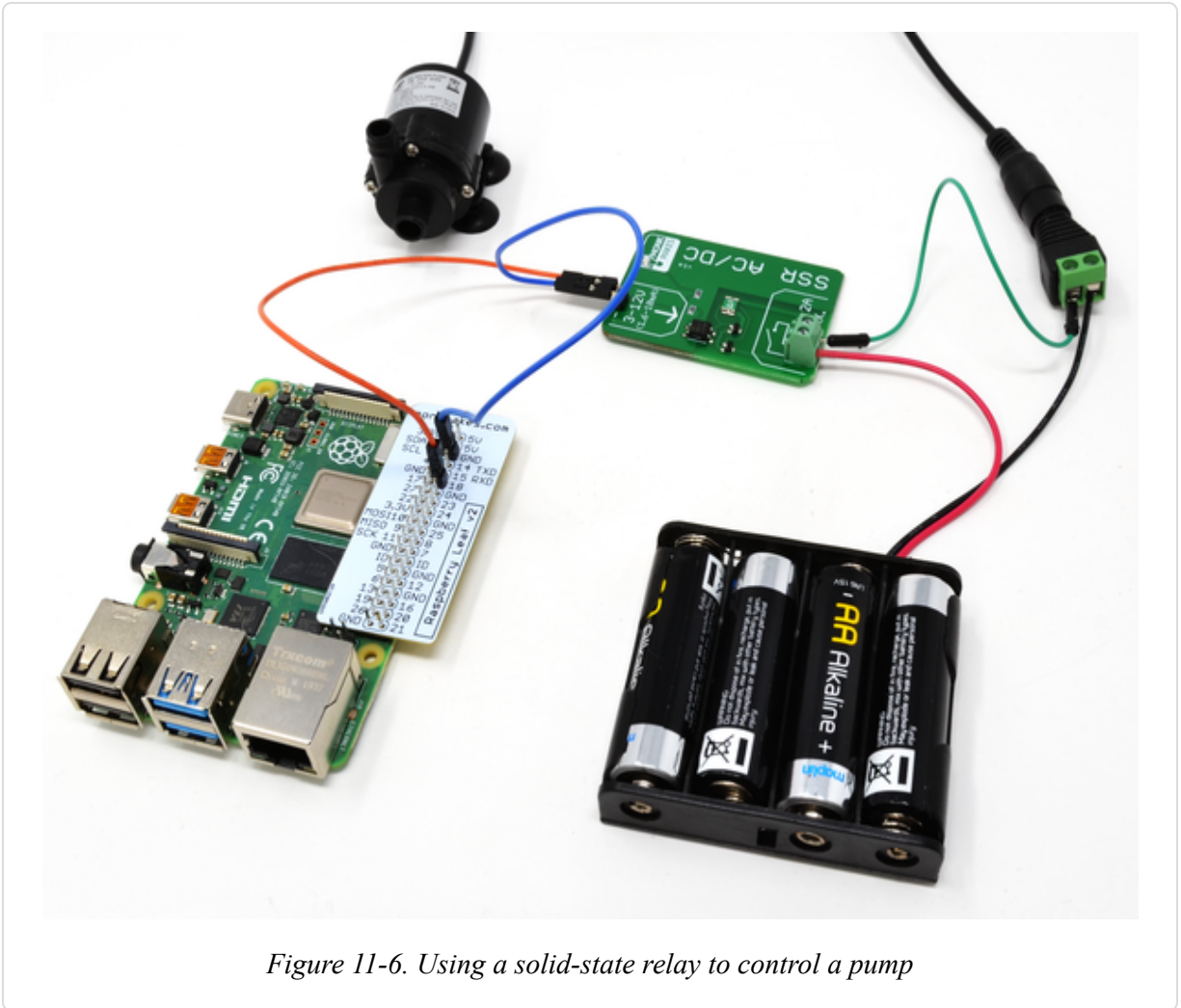


Figure 11-6. Using a solid-state relay to control a pump

The input to the SSR can be directly switched from a Raspberry Pi GPIO pin acting as a digital output. Then the Raspberry Pi GPIO pin connected to the SSR's input goes to 3.3V, and the output switches on, just like an electromechanical relay.

## Discussion

Whereas electromechanical relays as described in [Recipe 11.5](#) electrically isolate their input and output using an electromagnet and switch, SSRs use optoelectronics to isolate the input from the output. [Figure 11-7](#) shows the schematic diagram for the MonkMakes SSR. The part labeled IC1 is effectively an LED and series of photocells in a sealed and light-proof integrated circuit package. When the LED lights, it generates a voltage that

is used to control the two MOSFET transistors labeled Q1 and Q2. Two are required so that both DC and AC (where the voltage reverses) can be switched.

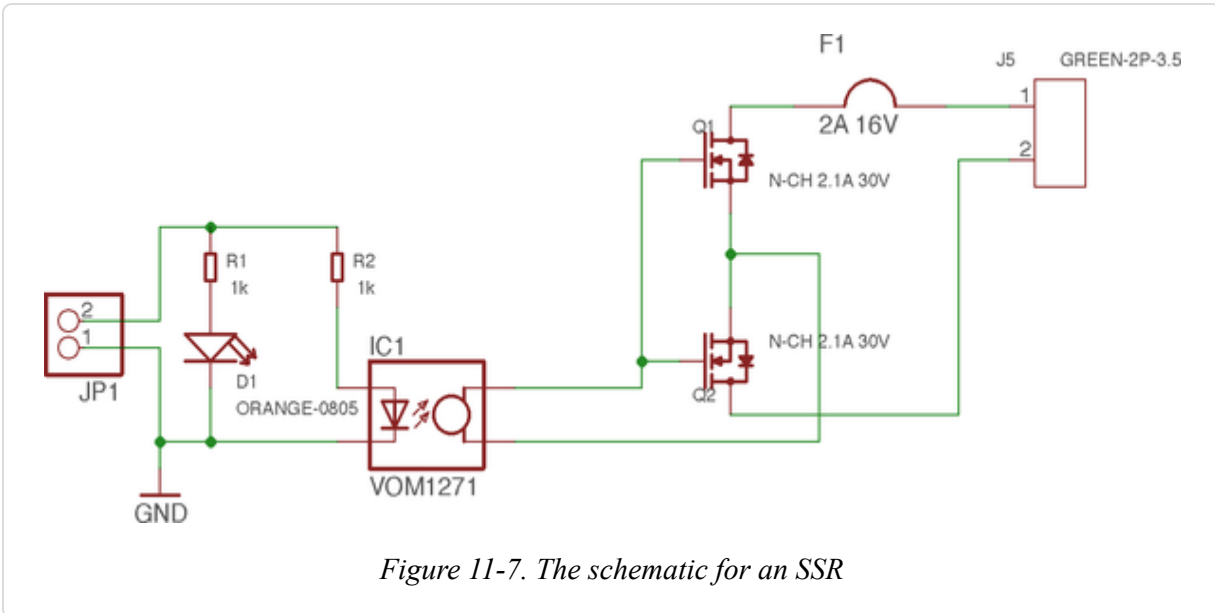


Figure 11-7. The schematic for an SSR

## See Also

To use an electromechanical relay, see [Recipe 11.5](#).

## 11.7 Controlling High-Voltage AC Devices

### Problem

You want to switch 110 or 220V alternating current (AC) on and off, using a Raspberry Pi.

### Solution

Use a PowerSwitch Tail II (see [Figure 11-8](#)) or [Four Output Power Relay](#). These handy devices make it really safe and easy to switch AC equipment on and off from a Raspberry Pi. They have an AC socket on one end and a plug (or plugs) on the other, like an extension cable; the only difference is

that the control box in the middle of the lead has three screw terminals. By attaching terminal 2 to GND and terminal 1 to a GPIO pin, the device acts like a switch to turn the appliance on and off.

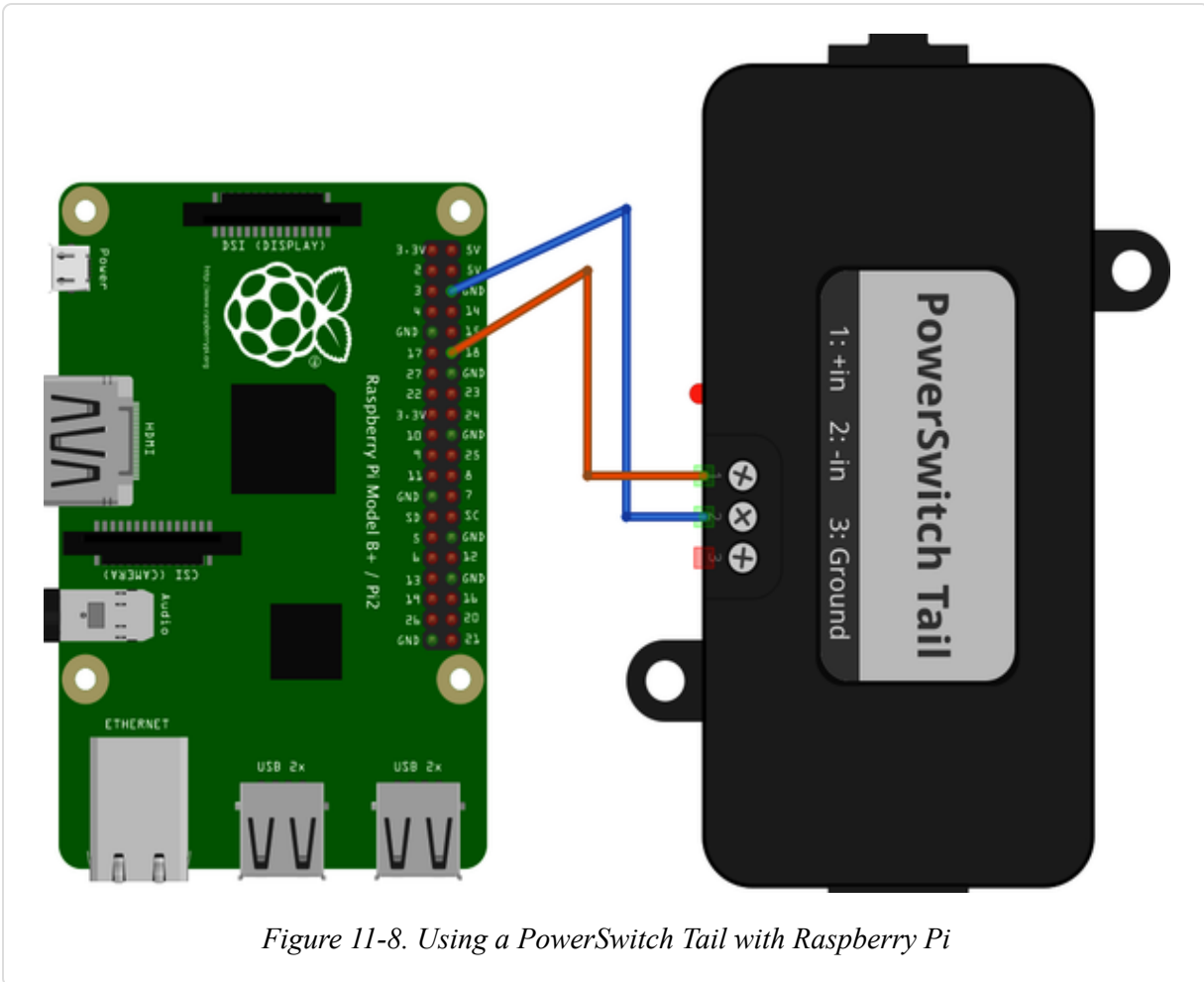


Figure 11-8. Using a PowerSwitch Tail with Raspberry Pi

You can use the same Python code that you did in [Recipe 11.1](#) to use the PowerSwitch Tail, as shown in [Figure 11-8](#).

## Discussion

The PowerSwitch Tail uses a relay, but to switch the relay, it uses a component called an *opto-isolator*, which has an LED shining onto a photo-TRIAC (a high-voltage, light-sensitive switch); when the LED is illuminated, the photo-TRIAC conducts, supplying current to the relay coil.

The LED inside the opto-isolator has its current limited by a resistor so that only 3mA flows through it when you supply it with 3.3V from a GPIO pin.

You will also find devices similar to, but less expensive than, the PowerSwitch Tail for sale on eBay and Amazon.

## See Also

For switching DC using a power MOSFET, see [Recipe 11.4](#); for switching using a relay on a breadboard, see [Recipe 11.5](#).

# 11.8 Controlling Hardware with Android and Bluetooth

## Problem

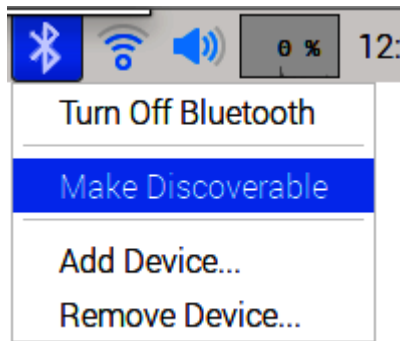
You want to use your Android mobile phone and Bluetooth to interact with your Raspberry Pi.

## Solution

Use the free Blue Dot Android app and Python library:

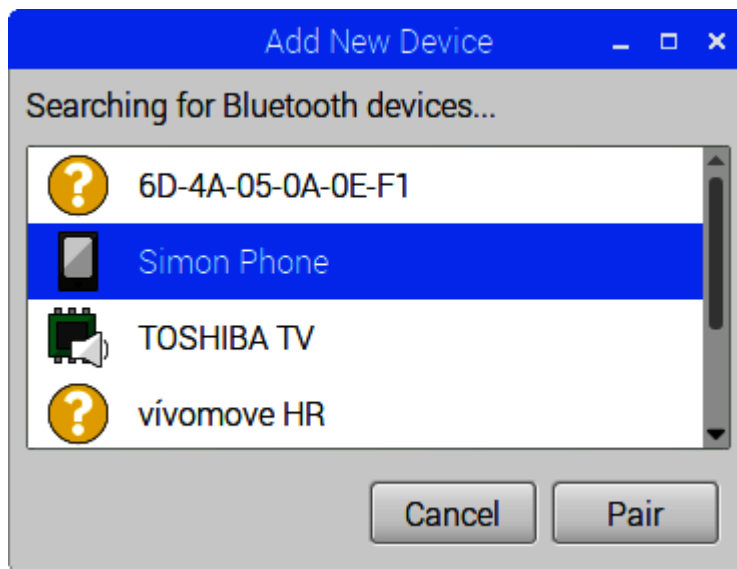
```
$ sudo pip3 install bluedot
```

Next, you need to make sure that your Raspberry Pi is *discoverable*. In the upper-right corner of the Raspberry Pi's screen, click the Bluetooth icon, and then click Make Discoverable ([Figure 11-9](#)).



*Figure 11-9. Making your Raspberry Pi discoverable in Bluetooth*

Next, you need to pair your Raspberry Pi and phone. Make sure that your phone has Bluetooth turned on, and then click Add New Device on your Raspberry Pi's Bluetooth menu ([Figure 11-10](#)).



*Figure 11-10. Pairing your Raspberry Pi and phone*

Find your phone in the list and then click Pair. You then are prompted on your phone to confirm a code to complete the pairing.

When the pairing is complete, go to the Play Store app on your phone. Search for and install the Blue Dot app. The app won't be able to work with your phone until you run a Python program that uses the Python Blue Dot

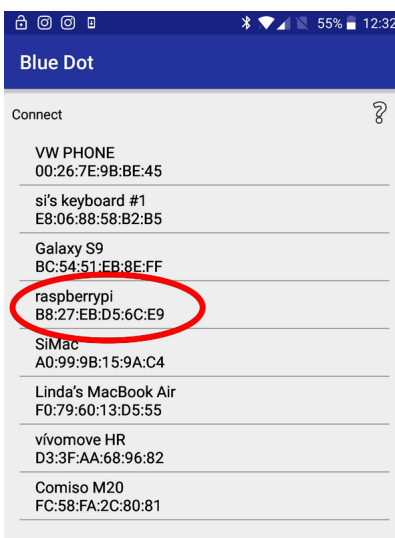


code to listen for commands, so run the following program (*ch\_11\_bluetodot.py*) on your Raspberry Pi:

```
from bluedot import BlueDot
bd = BlueDot()
while True:
    bd.wait_for_press()
    print("You pressed the blue dot!")
```

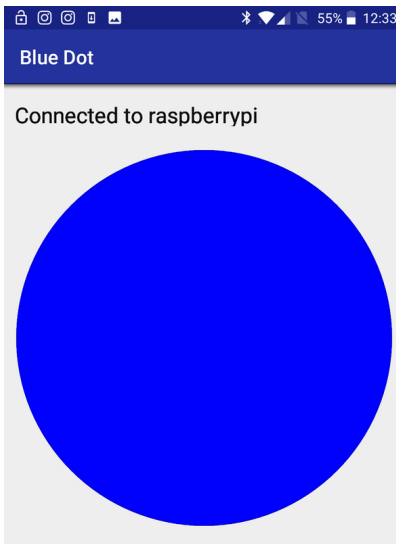
As with all the program examples in this book, you can download this program (see [Recipe 3.22](#)).

Now it's time to open the Blue Dot app on your phone. When you do this, it provides a list of Blue Dot devices ([Figure 11-11](#)).



*Figure 11-11. Connecting to your Raspberry Pi with Blue Dot*

After you're connected, the eponymous blue dot will appear, as shown in [Figure 11-12](#).



*Figure 11-12. The blue dot*

When you tap on the Blue Dot, your Python program will print out the message: “You pressed the blue dot!”:

```
$ python3 ch_11_bluedot.py
Server started B8:27:EB:D5:6C:E9
Waiting for connection
Client connected C0:EE:FB:F0:94:8F
You pressed the blue dot!
You pressed the blue dot!
You pressed the blue dot!
```

## Discussion

The big blue dot isn’t just a button; you can also use it as a joystick. You can slide, swipe, and rotate the dot. The Blue Dot library allows you to link handler functions to events such as swiping and rotating. For more information on this, take a look at [the documentation](#).

## See Also

For full information on Blue Dot, see [the Blue Dot website](#).

There is also a [Blue Dot Python module](#) that lets you use a second Raspberry Pi as the Blue Dot remote.

See [Recipe 1.17](#) for more information on using Bluetooth with a Raspberry Pi.

## 11.9 Making a User Interface to Turn Things On and Off

### Problem

You want to make an application to run on the Raspberry Pi that has a button for turning things on and off.

### Solution

Use `guizero` to provide the user interface for `gpiozero` to turn the pin on and off ([Figure 11-13](#)).



*Figure 11-13. An on/off switch in guizero*

If you haven't already done so, install `guizero` using the following command:

```
$ sudo pip3 install guizero
```

You'll need to connect an LED or some other kind of output device to GPIO pin 18. Using an LED ([Recipe 11.1](#)) is the easiest option for getting started.

As with all the program examples in this book, you can download the code for this recipe (see [Recipe 3.22](#)). The file is called `ch_11_gui_switch.py` and creates the switch shown in [Figure 11-13](#):

```
from gpiozero import DigitalOutputDevice
from guizero import App, PushButton

pin = DigitalOutputDevice(18)

def start():
    start_button.disable()
    stop_button.enable()
    pin.on()

def stop():
    start_button.enable()
    stop_button.disable()
    pin.off()

app = App(width=100, height=150)
start_button = PushButton(app, command=start, text="On")
start_button.text_size = 30
stop_button = PushButton(app, command=stop, text="Off",
enabled=False)
stop_button.text_size = 30
app.display()
```

## Discussion

The example uses a pair of buttons, and when you press one, it disables itself and enables its counterpart. It also uses `gpiozero` to change the state of the output pin using the `on()` and `off()` methods. This example would work just the same if we used the line `pin = LED(18)` rather than `pin = DigitalOutputDevice(18)`, but using `DigitalOutputDevice` keeps things more generic. After all, you could be controlling anything from pin 18, not just an LED.

## See Also

You can also use this program to control a high-power DC device ([Recipe 11.4](#)), a relay ([Recipe 11.5](#)), or a high-voltage AC device ([Recipe 11.7](#)).

For more information on `guizero`, see [Recipe 7.22](#).

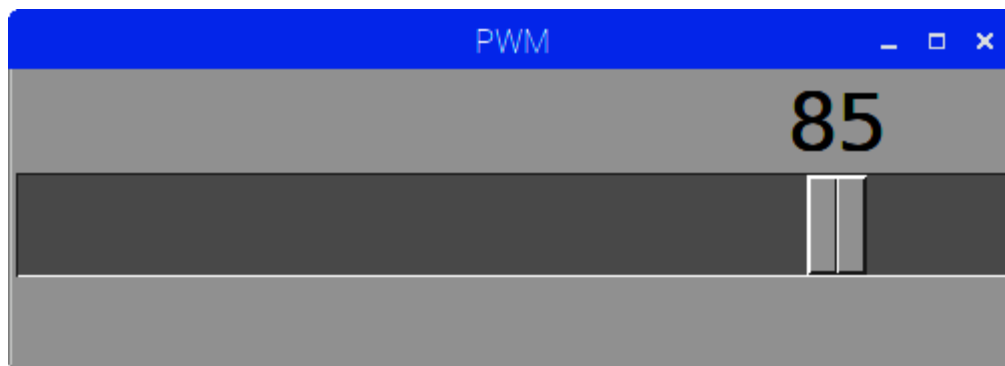
## 11.10 Making a User Interface to Control PWM Power for LEDs and Motors

### Problem

You want to make an application to run on the Raspberry Pi that has a slider to control power to a device using pulse-width modulation (PWM).

### Solution

Using the `gpiozero` and `guizero` user interface framework, write a Python program that uses a slider to change the PWM duty cycle between 0 and 100% ([Figure 11-14](#)).



*Figure 11-14. User interface for controlling PWM power*

You'll need to connect an LED or some other kind of output device to GPIO pin 18 that is capable of responding to a PWM signal. Using an LED ([Recipe 11.1](#)) is the easiest option to start with.

Open an editor and paste in the following code (the name of the file is `ch_11_gui_slider.py`):

```
from gpiozero import PWMOutputDevice
from guizero import App, Slider

pin = PWMOutputDevice(18)

def slider_changed(percent):
    pin.value = int(percent) / 100

app = App(title='PWM', width=500, height=150)
slider = Slider(app, command=slider_changed, width='fill',
height=50)
slider.text_size = 30
app.display()
```

As with all the program examples in this book, you can download the code for this recipe (see [Recipe 3.22](#)).

Run the program using the following command:

```
$ python3 gui_slider.py
```

## Discussion

The example program uses the `Slider` class. The `command` option runs the `slider_changed` command every time the value of the slider is changed. This updates the value of the output pin. The parameter to the `slider_changed` function is a string, even though it contains a number between 0 and 100, so `int` is used to convert it into a number, and then the percent value has to be divided by 100 to give a value between 0 and 1 for the PWM output.

## See Also

You can use this program to control an LED ([Recipe 11.1](#)), a DC motor ([Recipe 12.4](#)), or a high-power DC device ([Recipe 11.4](#)).

## 11.11 Making a User Interface to Change the Color of an RGB LED

### Problem

You want to control the color of an RGB LED.

### Solution

Use PWM to control the power to each of the red, green, and blue channels of an RGB LED.

To make this recipe, you will need the following:

- Breadboard and jumper wires (see “[Prototyping Equipment and Kits](#)”)
- Three 470Ω resistors (see “[Resistors and Capacitors](#)”)
- RGB common cathode LED (see “[OptoElectronics](#)”)

**Figure 11-15** shows how you can connect your RGB LED on a breadboard. Make sure that the LED is the correct way around; the longest lead should be the second lead from the top of the breadboard. This connection is called the *common cathode* because the negative connections (cathodes) of the red, green, and blue LEDs within the LED case have all their negative sides connected together to reduce the number of pins needed in the package.

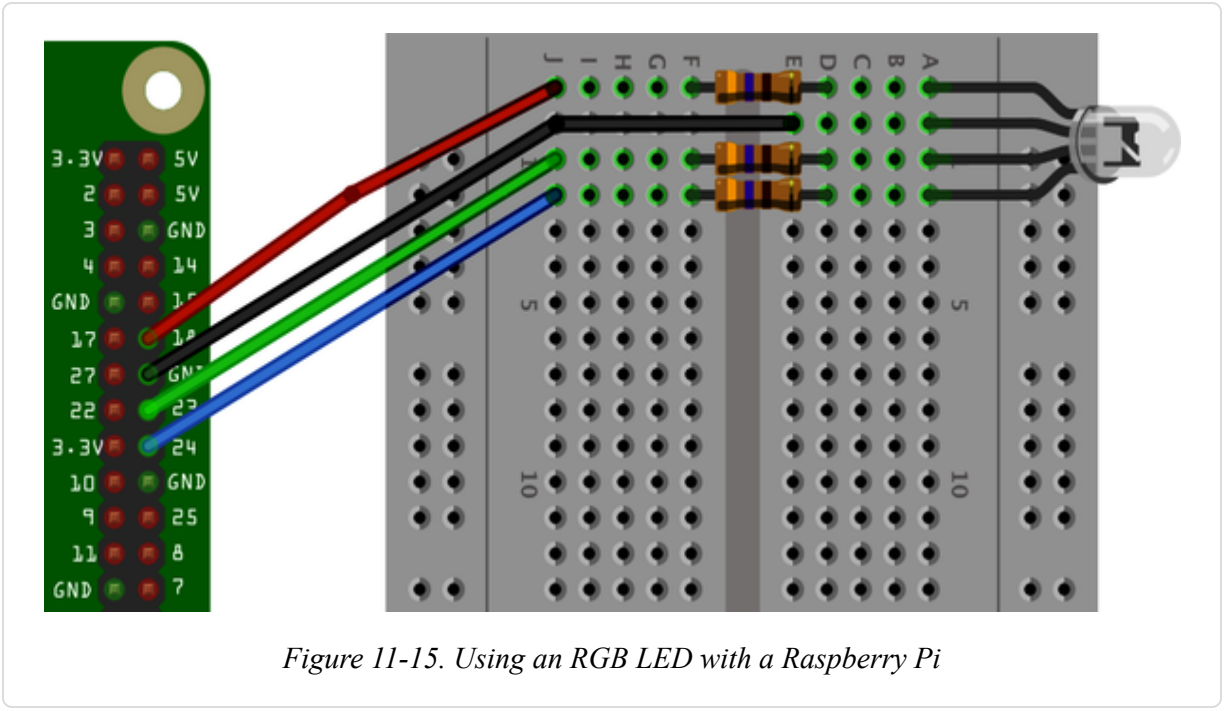
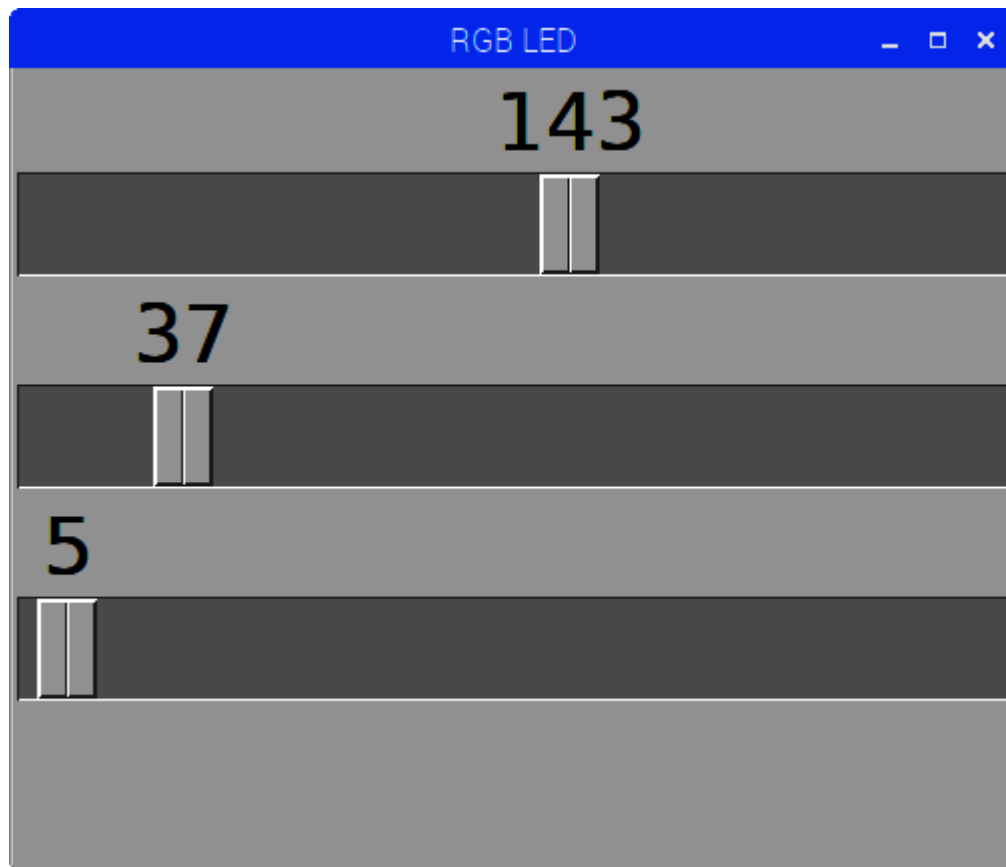


Figure 11-15. Using an RGB LED with a Raspberry Pi

An alternative to using a breadboard (avoiding fiddly resistors) is to use a Raspberry Squid (see [Recipe 10.10](#)).

The upcoming program has three sliders to control the red, green, and blue channels of the LED ([Figure 11-16](#)).





*Figure 11-16. User interface for controlling an RGB LED*

Open an editor and paste in the following code from the file `ch_11_gui_slider_RGB.py`:

```
from gpiozero import RGBLED
from guizero import App, Slider
from colorzero import Color

rgb_led = RGBLED(18, 23, 24)

red = 0
green = 0
blue = 0

def red_changed(value):
    global red
    red = int(value)
    rgb_led.color = Color(red, green, blue)
```

```

def green_changed(value):
    global green
    green = int(value)
    rgb_led.color = Color(red, green, blue)

def blue_changed(value):
    global blue
    blue = int(value)
    rgb_led.color = Color(red, green, blue)

app = App(title='RGB LED', width=500, height=400)

Slider(app, command=red_changed, end=255, width='fill',
height=50).text_size = 30
Slider(app, command=green_changed, end=255,
width='fill', height=50).text_size = 30
Slider(app, command=blue_changed, end=255,
width='fill', height=50).text_size = 30

app.display()

```

As with all the program examples in this book, you can download the code for this recipe (see [Recipe 3.22](#)).

## Discussion

The code is similar in operation to the control for a single PWM channel, described in [Recipe 11.10](#). However, in this case, you need three PWM channels and three sliders, one for each color.

The type of RGB LED used here is a common cathode. If you have the common anode type, you can still use it, but connect the common anode to the 3.3V pin on the GPIO connector. You will find that the slider is reversed, so a setting of 255 becomes *off* and 0 becomes full *on*.

When you are selecting an LED for this project, LEDs labeled “diffused” are best because they allow the colors to be mixed better.

## See Also

If you want to control just one PWM channel, see [Recipe 11.10](#).

## 11.12 Using an Analog Meter as a Display

### Problem

You want to connect an analog panel voltmeter to a Raspberry Pi.

### Solution

Assuming you have a 5V voltmeter, you can use a PWM output to drive the meter directly, connecting the negative side of the meter to GND and the positive side to a GPIO pin ([Figure 11-17](#)). If the meter is the common 5V kind, you'll only be able to display voltages up to 3.3V.

If you want to use almost the full range of a 5V voltmeter, you'll need a transistor to act as a switch for the PWM signal and a 1k $\Omega$  resistor to limit the current to the base of the transistor.

To make this recipe, you'll need the following:

- 5V panel meter (see [“Miscellaneous”](#))
- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- Two 1k $\Omega$  resistors (see [“Resistors and Capacitors”](#))
- Transistor 2N3904 (see [“Transistors and Diodes”](#))

[Figure 11-18](#) shows the breadboard layout for this.

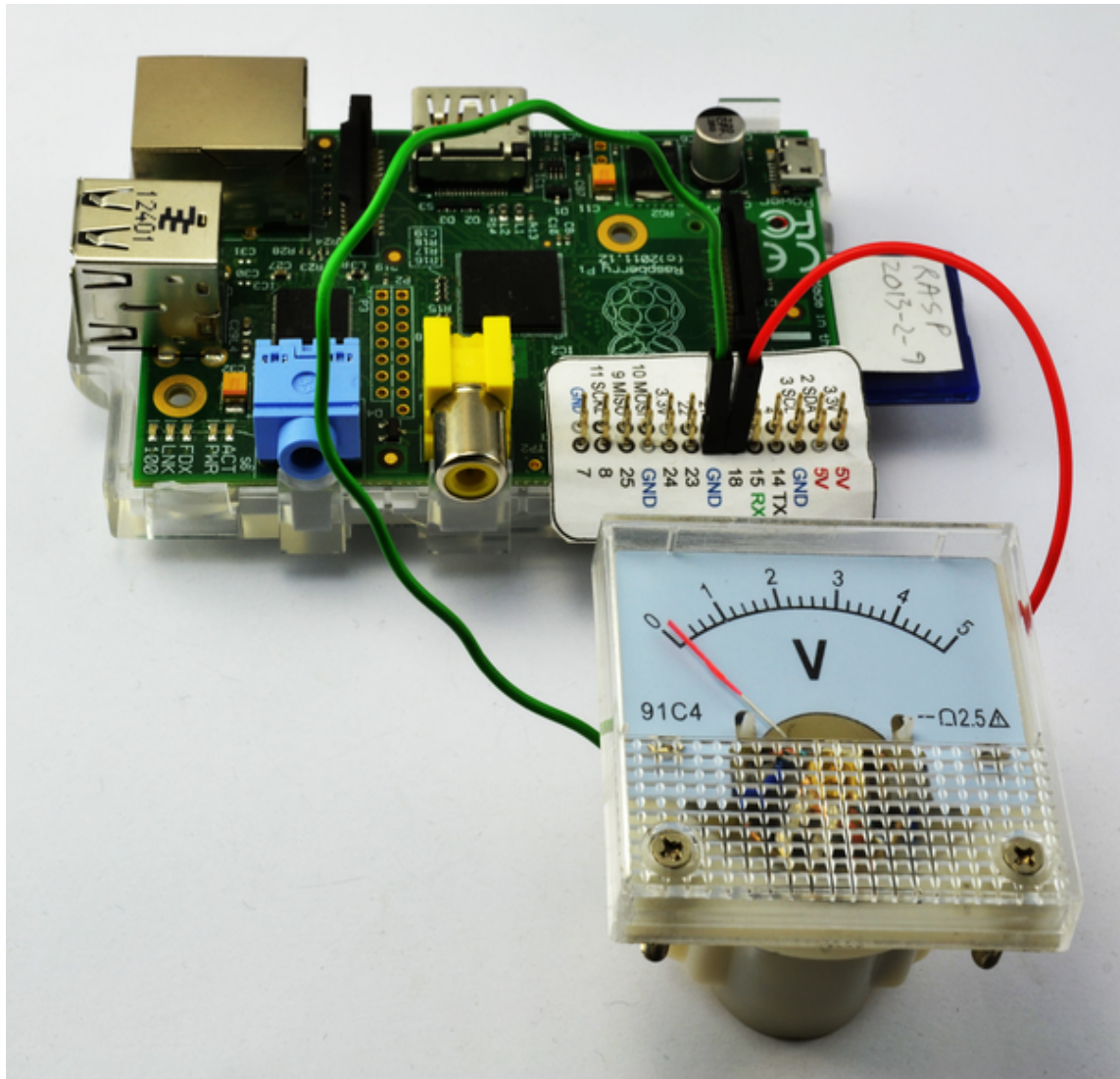
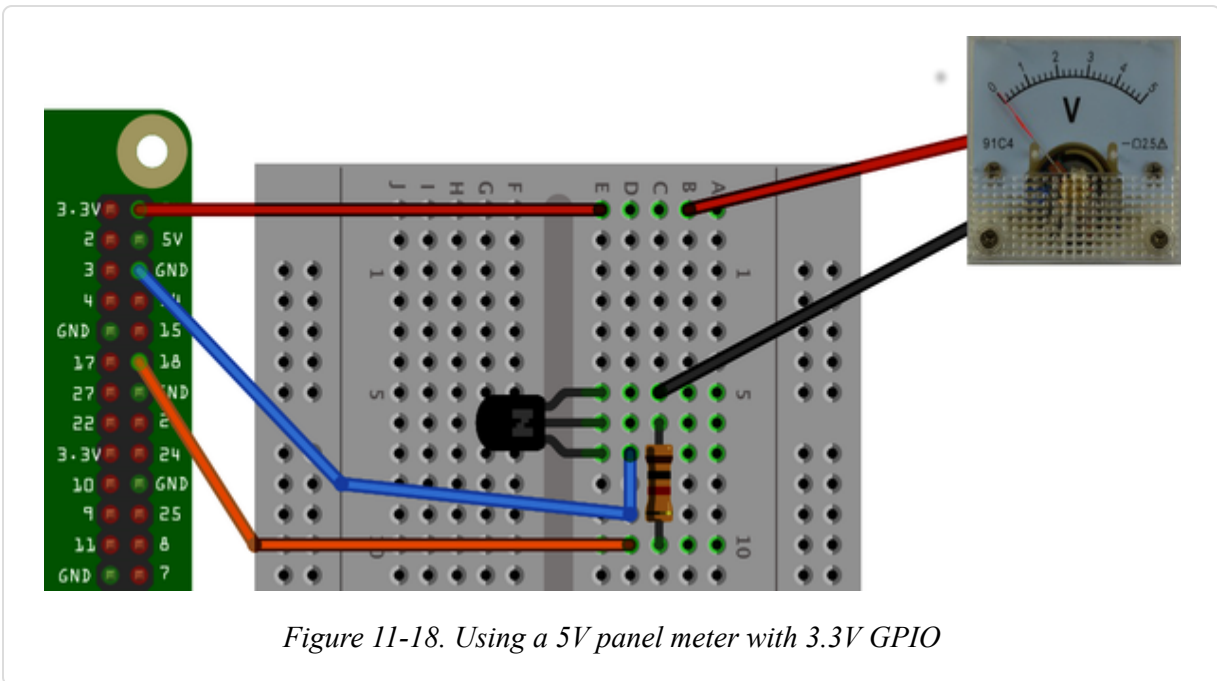


Figure 11-17. Connecting a voltmeter directly to a GPIO pin



## Discussion

To test the voltmeter, use the same program as you did for controlling the brightness of the LED in [Recipe 11.10](#).

You will probably notice that the needle gives a steady reading at either end of the scale, but everywhere else it jitters a bit. This is a side effect of the way the PWM signals are generated. For a steadier result, you can use external PWM hardware like the 16-channel module used in [Recipe 12.3](#).

## See Also

For more information about how old-fashioned voltmeters work, see [Wikipedia](#).

For more information on using a breadboard and jumper wires with the Raspberry Pi, see [Recipe 10.9](#).

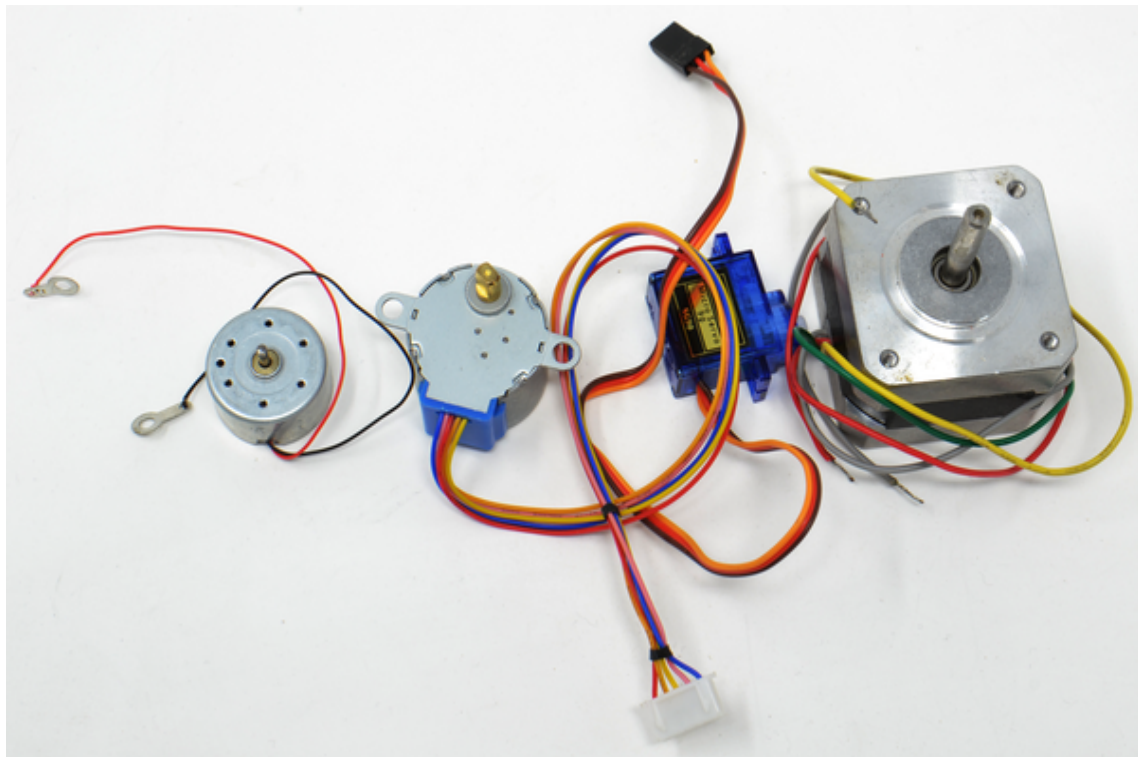
# Chapter 12. Motors

---

## 12.0 Introduction

In this chapter, you will investigate the use of different types of motors with the Raspberry Pi. This includes DC motors, servomotors, and stepper motors.

Motors come in all shapes and sizes (Figure 12-1). The most common is the simple brushed DC motor that you might find in a toy car. In this chapter, we will also look at servomotors, where the position of the motor's shaft is set using pulses generated by the Raspberry Pi, and at stepper motors, which don't rotate smoothly, but, as the name suggests, in tiny steps, as their coils are energized in sequence.



*Figure 12-1. A selection of motors*

## 12.1 Controlling Servomotors

### Problem

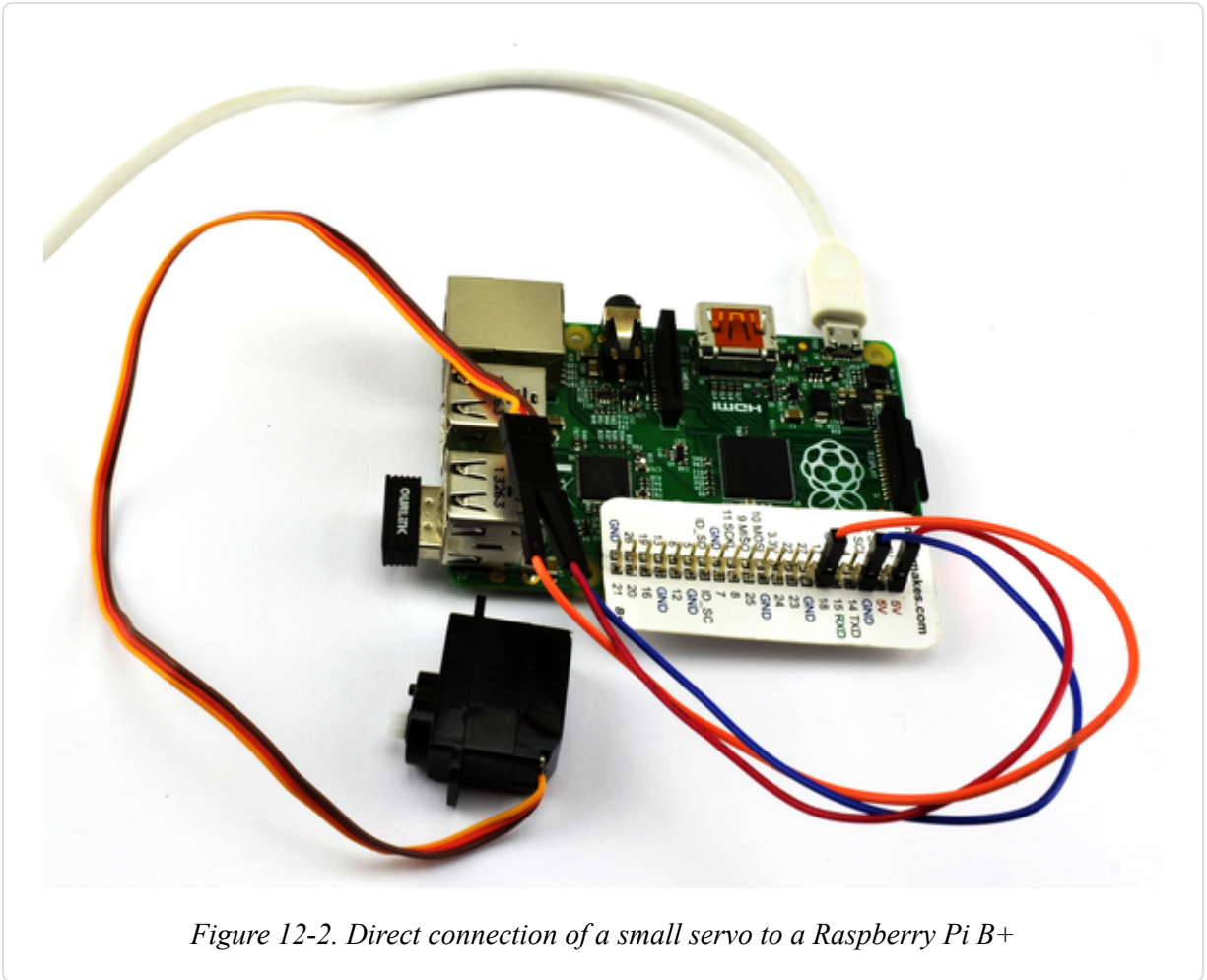
You want to use a Raspberry Pi to control the position of a servomotor.

### Solution

Use pulse-width modulation (PWM) to control the width of pulses to a servomotor to change its angle. Although this will work, the PWM generated is not completely stable, so there will be a little bit of jitter with the servo. For alternative solutions that produce much more stable pulse timing, see Recipes [12.2](#) and [12.3](#).

If you have a Raspberry Pi 1, you should power the servo from a separate 5V power supply because peaks in the load current are very likely to crash or overload the Raspberry Pi. If you have a Raspberry Pi B+ or newer, improvements in the onboard voltage regulation mean that you might get away with powering small servos directly from the 5V pin on the general-purpose input/output (GPIO) port.

[Figure 12-2](#) shows a small 9g servo (see “[Miscellaneous](#)”) working quite happily with a Raspberry Pi B+.



*Figure 12-2. Direct connection of a small servo to a Raspberry Pi B+*

The leads of the servo are usually colored so that the 5V wire is red, the ground is brown, and the control lead is orange. The 5V and ground leads are connected to the GPIO header 5V and GND pins, and the control lead is connected to pin 18. The connections are made with female-to-male header leads.

If you are using a separate power supply, a breadboard is a good way of keeping all the leads together.

In this case, you will need the following:

- 5V servomotor (see [“Miscellaneous”](#))
- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- 1k $\Omega$  resistor (see [“Resistors and Capacitors”](#))
- 5V 1A power supply or 4.8V battery pack (see [“Miscellaneous”](#))



Figure 12-3 shows the breadboard layout for this.

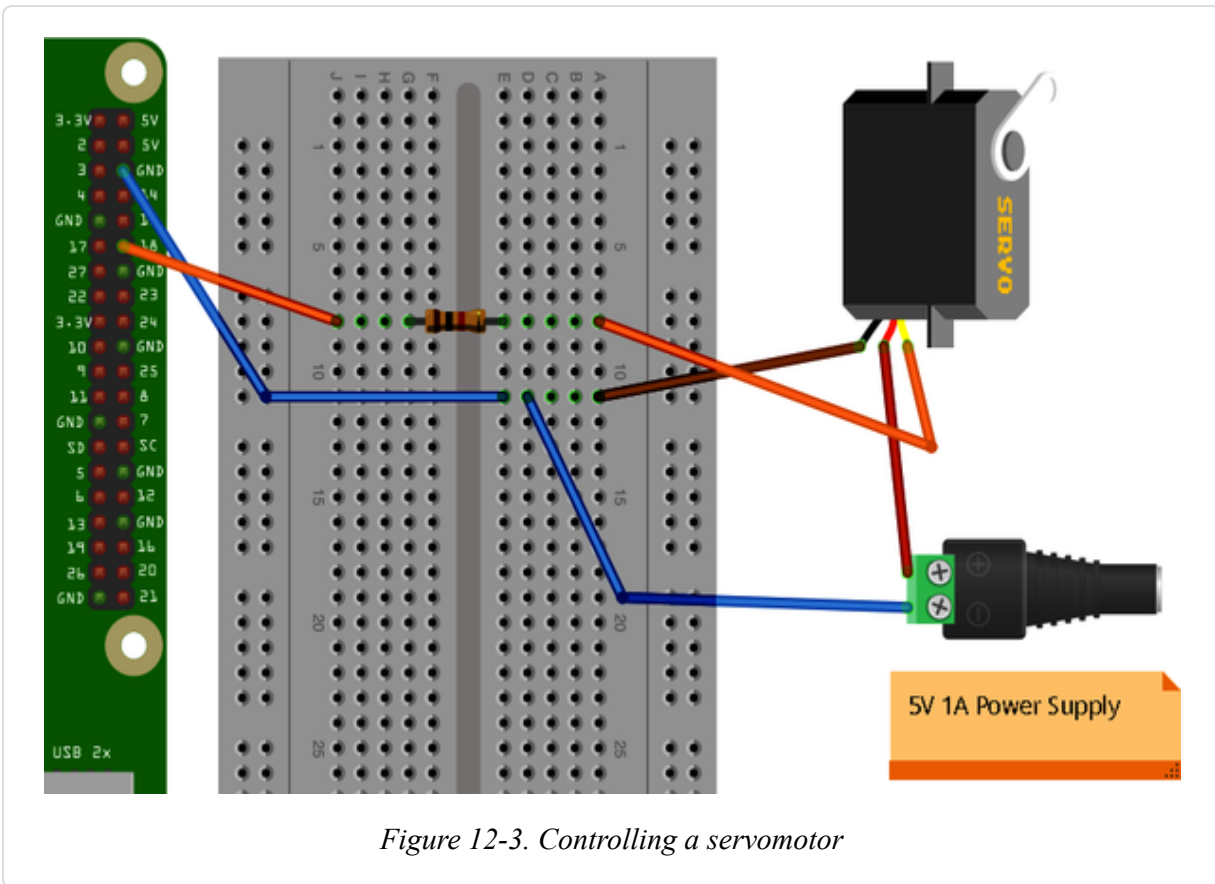


Figure 12-3. Controlling a servomotor

The 1kΩ resistor is not essential, but it does protect the GPIO pin from unexpectedly high currents in the control signal, which could occur if a fault developed on the servo.

You can, if you prefer, power the servo from a battery pack rather than a power supply. Using a four-cell AA battery holder with rechargeable batteries will provide around 4.8V and work well with a servo. Using four alkali AA cells to provide 6V will be fine for many servos, but check the datasheet of your servo to make sure it is OK with 6V.

The user interface for setting the angle of the servo is based on the `ch_11_gui_slider.py` program, which is intended for controlling the brightness of an LED (Recipe 11.10). However, you can modify it so that the slider sets the angle to be between -90 and 90 degrees (Figure 12-4).

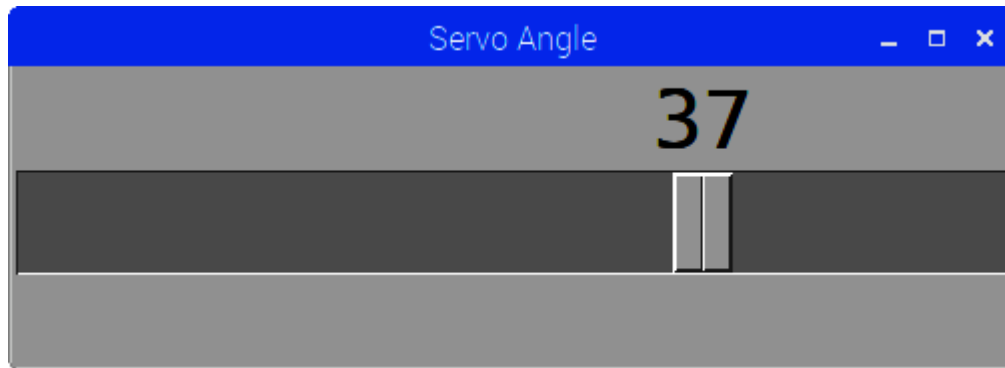


Figure 12-4. User interface for controlling a servomotor

Open an editor and paste in the following code (the file is called `ch_12_servo.py`):

```
from gpiozero import AngularServo
from guizero import App, Slider

servo = AngularServo(18, min_pulse_width=0.5/1000,
max_pulse_width=2.5/1000)

def slider_changed(angle):
    servo.angle = int(angle)

app = App(title='Servo Angle', width=500, height=150)
slider = Slider(app, start=-90, end=90, command=slider_changed,
width='fill',
                height=50)
slider.text_size = 30
app.display()
```

As with all the program examples in this book, you can download this code (see [Recipe 3.22](#)).

Note that this program uses a graphical user interface, so you can't run it from SSH or the Terminal. You must run it from the windowing environment on the Pi itself or via remote control using virtual network computing (VNC) ([Recipe 2.8](#)).

The `gpiozero` class `AngularServo` takes care of all the pulse generation. It just leaves us to specify the angle to which we want the servo

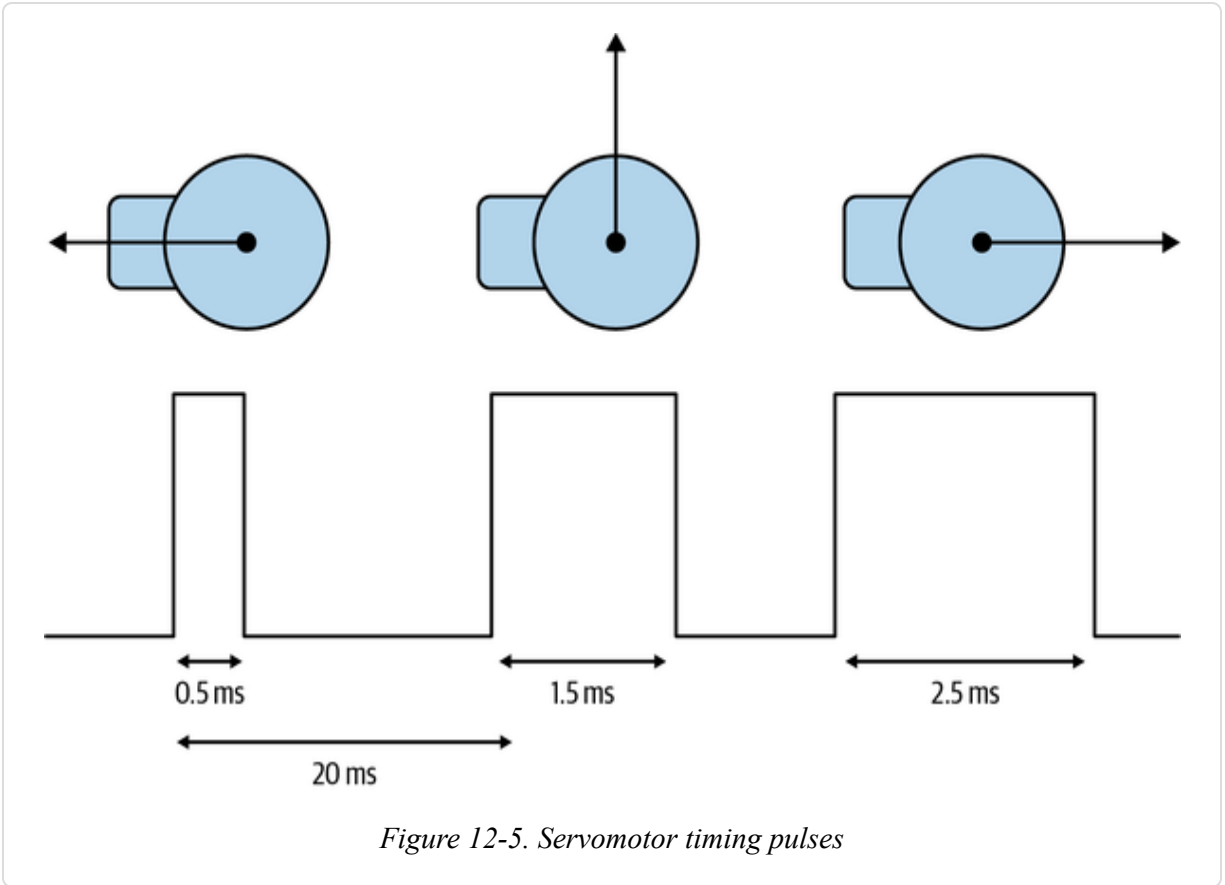
arm to position itself. Pretty much all other software that works with servomotors specifies angles of between 0 and 180 degrees, where 0 is as far as the servo arm can go to one side, 90 is in the middle, and 180 is as far as it can go to the other side. The `gpiozero` library does things differently, and probably more logically, by referring to the center position as 0 and to the angles on one side as negative and on the other side as positive.

When defining the servo, the first parameter (18 in this case) specifies the control pin for the servomotor. The optional parameters of `min_pulse_width` and `max_pulse_width` set the minimum and maximum pulse lengths in seconds. For a typical servo, these values should be 0.5 milliseconds and 2.5 milliseconds. For some reason, the default values in `gpiozero` are 1 and 2 milliseconds; thus the servomotor has a very restricted range unless you set these values as we have here.

## Discussion

Servomotors are used in remote control vehicles and robotics. Most servomotors are not *continuous*; that is, they cannot rotate all the way around but rather can rotate only over an angle range of about 180 degrees.

The position of the servomotor is set by the length of a pulse. The servo expects to receive a pulse at least every 20 milliseconds. If that pulse is high for 0.5 milliseconds, the servo angle will be  $-90$  degrees; if it's high for 1.5 milliseconds, the motor will be at its center position (0 degrees); and if the pulse is high for 2.5 milliseconds, the servo angle will be 90 degrees (Figure 12-5).



If you have a few servomotors to connect, the MonkMakes Servo Six board (Figure 12-6) makes the wiring easier.

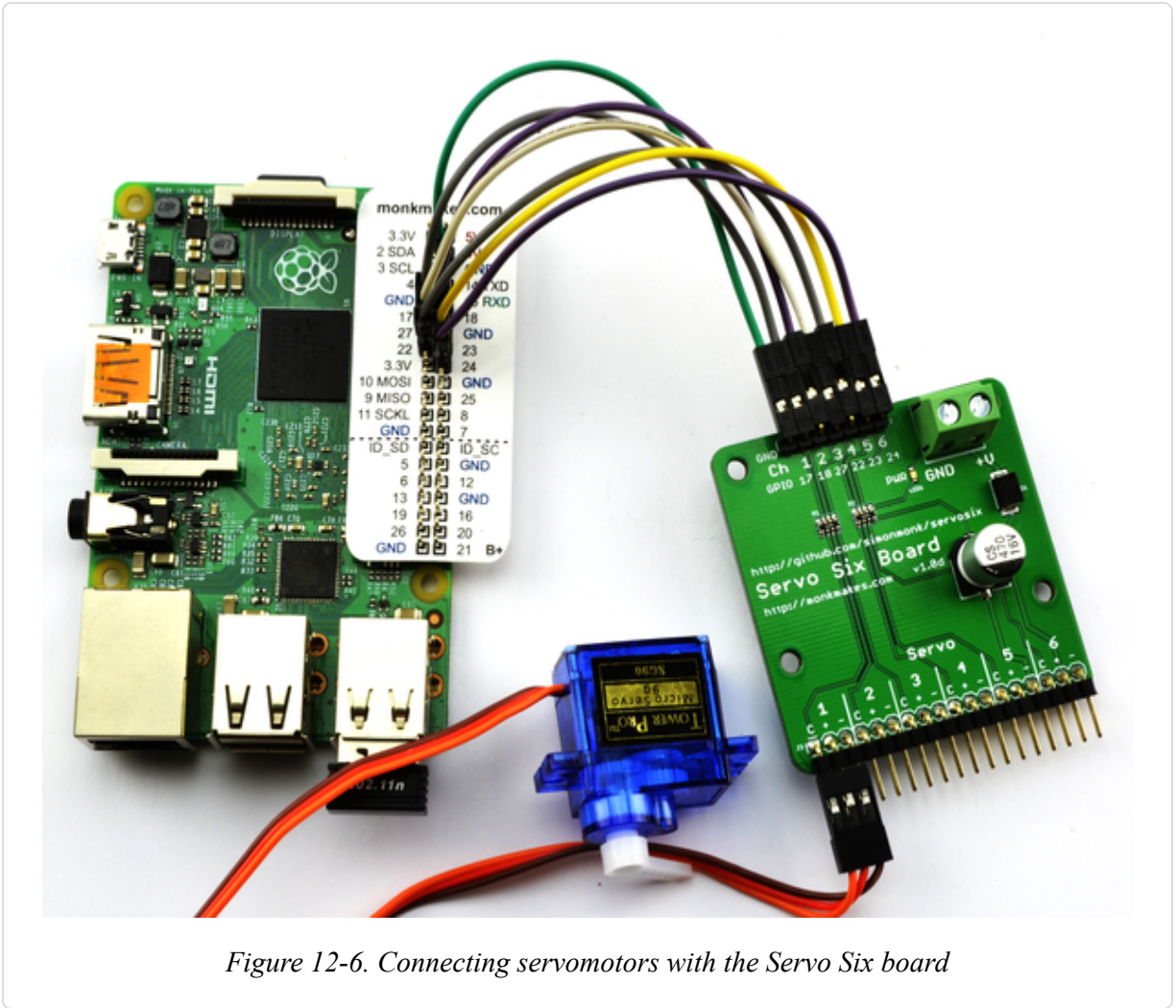


Figure 12-6. Connecting servomotors with the Servo Six board

## See Also

If you have a lot of servos to control or require greater stability and precision, you can use a dedicated servo controller module, as described in [Recipe 12.3](#).

See more information in the [full documentation of the Servo Six board](#).

Adafruit has developed [another method of servo control](#).

For an alternative solution that produces much more stable pulse timing using the ServoBlaster device driver software, see [Recipe 12.2](#).

## 12.2 Controlling Servomotors Precisely

## Problem

The pulse generation function of the `gpiozero` library is not precise or jitter-free enough for your servo application.

## Solution

Install the ServoBlaster device driver.

### ServoBlaster and Sound

The ServoBlaster software uses Raspberry Pi hardware that is also involved in generating sound. Thus, you won't be able to play audio through your Raspberry Pi's audio jack or HDMI while using ServoBlaster.

The ServoBlaster software created by Richard Hirst uses Raspberry Pi CPU hardware to generate pulses with much more accurate timings than are possible using `gpiozero`. Install ServoBlaster using the following commands and then reboot your Raspberry Pi:

```
$ git clone https://github.com/srcshelton/servoblaster.git
$ cd servoblaster
$ sudo make
$ sudo make install
```

You can modify the program from [Recipe 12.1](#) to use the ServoBlaster code. You can find the modified program in the file `ch_12_servo_blaster.py`. As with all the program examples in this book, you can download it (see [Recipe 3.22](#)). This program assumes that the servo control pin is connected to GPIO 18:

```
import os
from guizero import App, Slider
```

```

servo_min = 500 # uS
servo_max = 2500 # uS
servo = 2 # GPIO 18

def map(value, from_low, from_high, to_low, to_high):
    from_range = from_high - from_low
    to_range = to_high - to_low
    scale_factor = float(from_range) / float(to_range)
    return to_low + (value / scale_factor)

def set_angle(angle):
    pulse = int(map(angle+90, 0, 180, servo_min, servo_max))
    command = "echo {}={{}us > /dev/servoblaster".format(servo,
pulse)
    os.system(command)

def slider_changed(angle):
    set_angle(int(angle))

app = App(title='Servo Angle', width=500, height=150)
slider = Slider(app, start=-90, end=90, command=slider_changed,
width='fill',
                height=50)
slider.text_size = 30
app.display()

```

The user interface code is almost the same as [Recipe 12.1](#). The differences are in the `set_angle` function. This function first uses a utility function called `map` that converts the angle into a pulse duration using the constants `servo_min` and `servo_max`. Then it constructs a command that will be run as if from the command line. The format of this line starts with the `echo` command, followed by the servo number to be controlled, an equal sign, and then a pulse duration in microseconds. This string part of the command will be directed to the device `/dev/servoblaster`. The servo will then adjust its angle.

## Killing ServoBlaster

When ServoBlaster, or more specifically, the service `servo.d`, is running, you will not be able to use the servo pins for anything, and audio on the Raspberry Pi will not work. So when you need to use the pins for something else, use the following commands to disable ServoBlaster and then reboot your Pi:

```
$ sudo update-rc.d servoblaster disable
$ sudo reboot
```

When your Raspberry Pi restarts, ServoBlaster will no longer have control of your pins, and you'll be able to use sound again on your Pi. You can always turn ServoBlaster back on again using:

```
$ sudo update-rc.d servoblaster enable
$ sudo reboot
```

## Discussion

The ServoBlaster driver is very powerful, and you can configure it to allow you to use pretty much all the GPIO pins to control servos. Its default setup defines eight GPIO pins to act as servo control pins. These are each given a channel number, as shown in [Table 12-1](#).

*Table 12-1. Servo channel default pin allocation for ServoBlaster*

Servo channel	GPIO pin
0	4
1	17
2	18
3	27



Servo channel	GPIO pin
4	22
5	23
6	24
7	25

Connecting so many servos can result in jumper-lead spaghetti. A board like the MonkMakes Servo Six ([Figure 12-6](#)) greatly simplifies the wiring of the servos to your Raspberry Pi.

## See Also

More information is available in the [full documentation for ServoBlaster](#).

If you don't need the precise timing of ServoBlaster, the `gpiozero` library can also generate pulses for your servo, as described in [Recipe 12.1](#).

## 12.3 Controlling Multiple Servomotors Precisely

### Problem

You need to control lots of servos with high precision and without the loss of sound that comes with using ServoBlaster.

### Solution

Although the ServoBlaster code (see [Recipe 12.2](#)) allows you to control up to eight servos accurately, it does rather take over your Raspberry Pi's hardware and disables sound generation.

The alternative to ServoBlaster is to use a servomotor HAT like the one shown in [Figure 12-7](#) that has its own servo controlling hardware, relieving the Raspberry Pi's hardware.

This Adafruit HAT allows you to control up to 16 servos or PWM channels using the I2C interface of the Raspberry Pi. The servos just plug straight into the HAT.

Power is supplied to the logic circuits of the module from the 3.3V connection of the Raspberry Pi. This is entirely separate from the power supply for the servomotors, which comes from an external 5V power adapter.

You can, if you prefer, power the servos from a battery pack rather than a power supply. Using a four-cell AA battery holder with rechargeable batteries provides around 4.8V and works well with most servos. Using four alkali AA cells to provide 6V will be fine for many servos, but check the datasheet of your servo to make sure it is OK with 6V.

The pin headers for connecting servos are conveniently arranged so that the servo lead fits directly onto the pins. Be careful to get them facing the correct way.

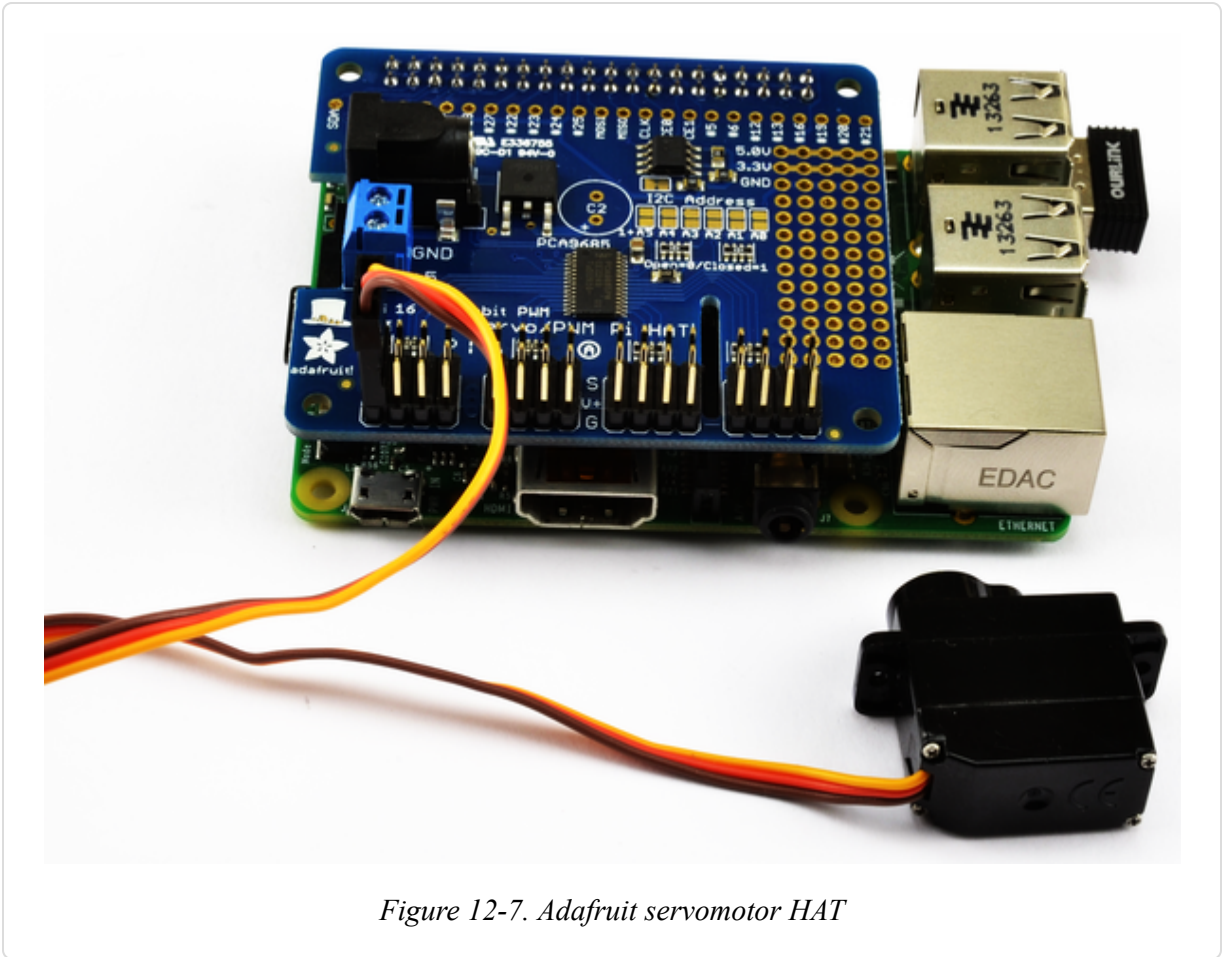


Figure 12-7. Adafruit servomotor HAT

To use the Adafruit software for this module, you will need to set up I2C on the Raspberry Pi ([Recipe 10.4](#)).

The software for this board uses some helpful software from Adafruit that allows you to use an entire range of their accessory add-on boards.

To install the Adafruit *blinka* code needed for this board, run the following commands:

```
$ pip3 install adafruit-blinka
$ sudo pip3 install adafruit-circuitpython-servokit
```

Open an editor and paste in the following code (the name of the file is *ch\_12\_servo\_adafruit.py*):

```

from adafruit_servokit import ServoKit
from guizero import App, Slider

servo_kit = ServoKit(channels=16)

def slider_changed(angle):
    servo_kit.servo[0].angle = int(angle) + 90

app = App(title='Servo Angle', width=500, height=150)
slider = Slider(app, start=-90, end=90, command=slider_changed,
width='fill',
                    height=50)
slider.text_size = 30
app.display()

```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

When you run the program, you will get the same window containing a slider as shown in [Figure 12-4](#). Use this to move the servo arm about. Adafruit's software is not compatible with Python 2, so you need to run any programs that use the Adafruit software via the `python3` command.

Note that this program uses a graphical user interface (GUI), so you can't run it from SSH or the Terminal; you must run it from the windowing environment on the Pi itself or via remote control using VNC ([Recipe 2.8](#)):

```
$ python3 ch_12_servo_adafruit.py
```

The Adafruit software uses the servomotor angle range of 0 to 180, rather than `gpiozero`'s -90 to 90 degrees, so to keep the user interface the same, 90 is added to the angle supplied by the slider.

To address a specific servo channel among the 16 available channels, the channel number (between 0 and 15) is specified inside the square brackets in the command `servo_kit.servo[0].angle`.

## Discussion

When selecting a power supply for this module, remember that a standard remote control servo can easily draw 400mA while it's moving, and more if it's under load. So if you plan to have a lot of large servos moving at the same time, you will need a big power adapter.

## See Also

Visit Adafruit for [information on this Adafruit product](#).

A Servo HAT is great if your Raspberry Pi is close to the servomotors, but if your servos are distant from where you want the Raspberry Pi to be, Adafruit also sells a servo module (product ID 815) that has the same servo controller hardware as the servo HAT but just four pins to connect the I2C interface of the board to the Raspberry Pi's I2C interface.

## 12.4 Controlling the Speed of a DC Motor

### Problem

You want to control the speed of a DC motor using your Raspberry Pi.

### Solution

You can use the same design as [Recipe 11.4](#). However, you should place a diode across the motor to prevent voltage spikes from damaging the transistor or even the Raspberry Pi. The 1N4001 is a suitable diode for this. The diode has a stripe at one end, so make sure that this is facing the proper direction ([Figure 12-8](#)).

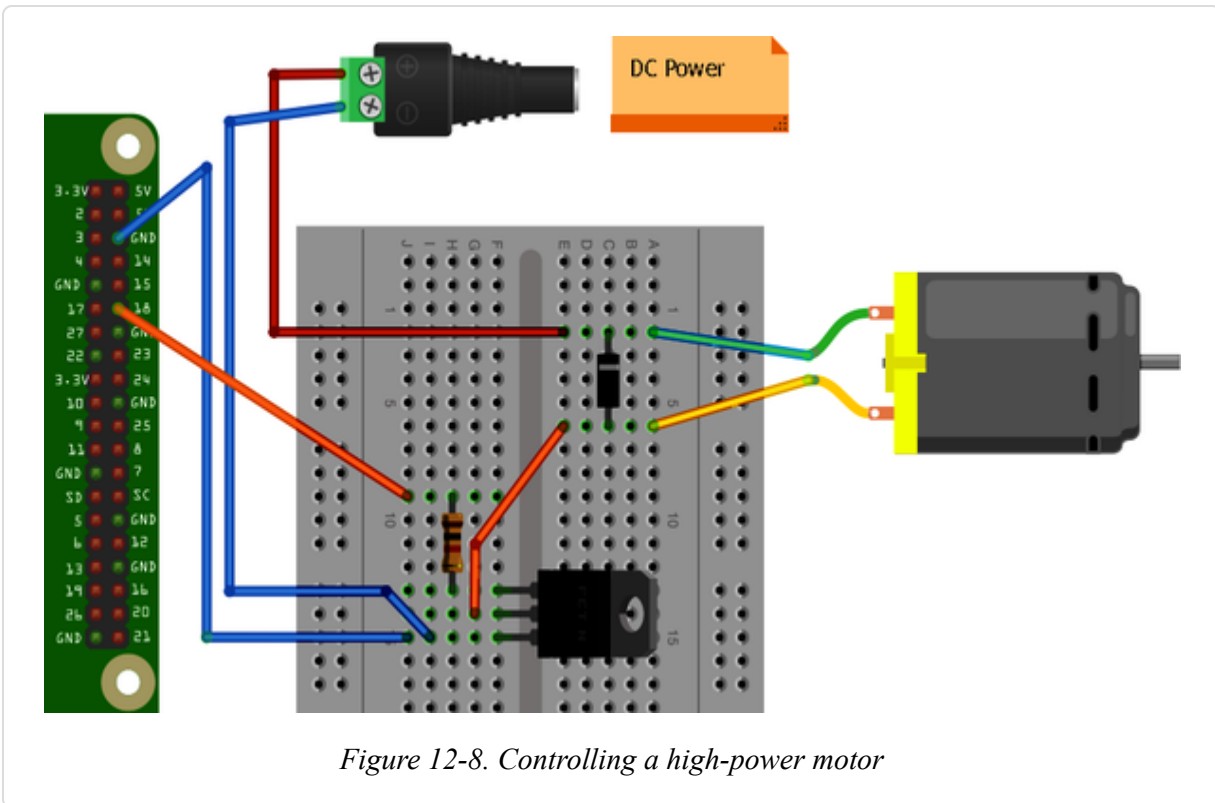


Figure 12-8. Controlling a high-power motor

You'll need the following:

- 3V to 12V DC motor
- Breadboard and jumper wires (see “[Prototyping Equipment and Kits](#)”)
- 1k $\Omega$  resistor (see “[Resistors and Capacitors](#)”)
- MOSFET transistor FQP30N06L (see “[Transistors and Diodes](#)”)
- Diode 1N4001 (see “[Transistors and Diodes](#)”)
- Power supply with voltage to match the motor

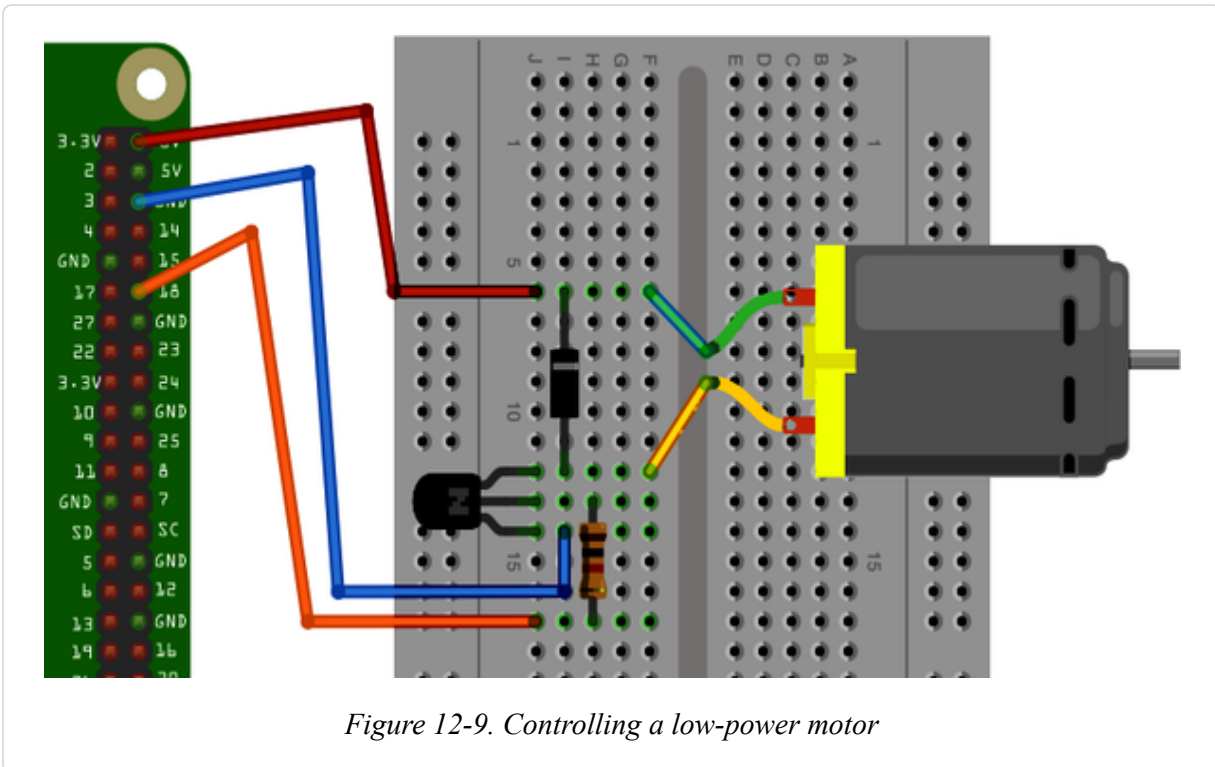
As with all the program examples in this book, you can download this program (see [Recipe 3.22](#)). The file is called `ch_12_gui_slider.py`.

Note that this program uses a GUI, so you can't run it from SSH. You must run it from the windowing environment on the Pi itself or via remote control using VNC ([Recipe 2.8](#)).

## Discussion

If you are using only a low-power DC motor (less than 200mA), you can use a smaller (and cheaper) transistor such as the 2N3904 (see “[Transistors and Diodes](#)”). [Figure 12-9](#) shows the breadboard layout to use a 2N3904.

You can probably get away with powering a small motor from the 5V supply line on the GPIO connector. If you find that the Raspberry Pi crashes, use an external power supply, as shown in [Figure 12-8](#).



*Figure 12-9. Controlling a low-power motor*

## See Also

For more information on using a breadboard and jumper wires with the Raspberry Pi, see [Recipe 10.9](#).

This design controls only the motor’s speed. It can’t control its direction. For that, you need to see [Recipe 12.5](#).

## 12.5 Controlling the Direction of a DC Motor

### Problem

You want to control both the speed and direction of a small DC motor.

## **Solution**

Use an H-bridge chip or module, the most common chip being the L293D. These are low cost and easy to use. Other H-bridge chips or modules usually use the same pair of control pins for the direction of each motor (see the discussion).

The L293D chip is actually capable of driving two motors without any extra hardware. The Discussion also mentions a few other options for controlling DC motors. To try out the L293D to control a motor, you'll need the following:

- 3V to 12V DC motor
- Breadboard and jumper wires (male-to-female; see “[Prototyping Equipment and Kits](#)”)
- L293D chip (see “[Integrated Circuits](#)”)
- Power supply with voltage to match the motor

[Figure 12-10](#) shows the breadboard layout.



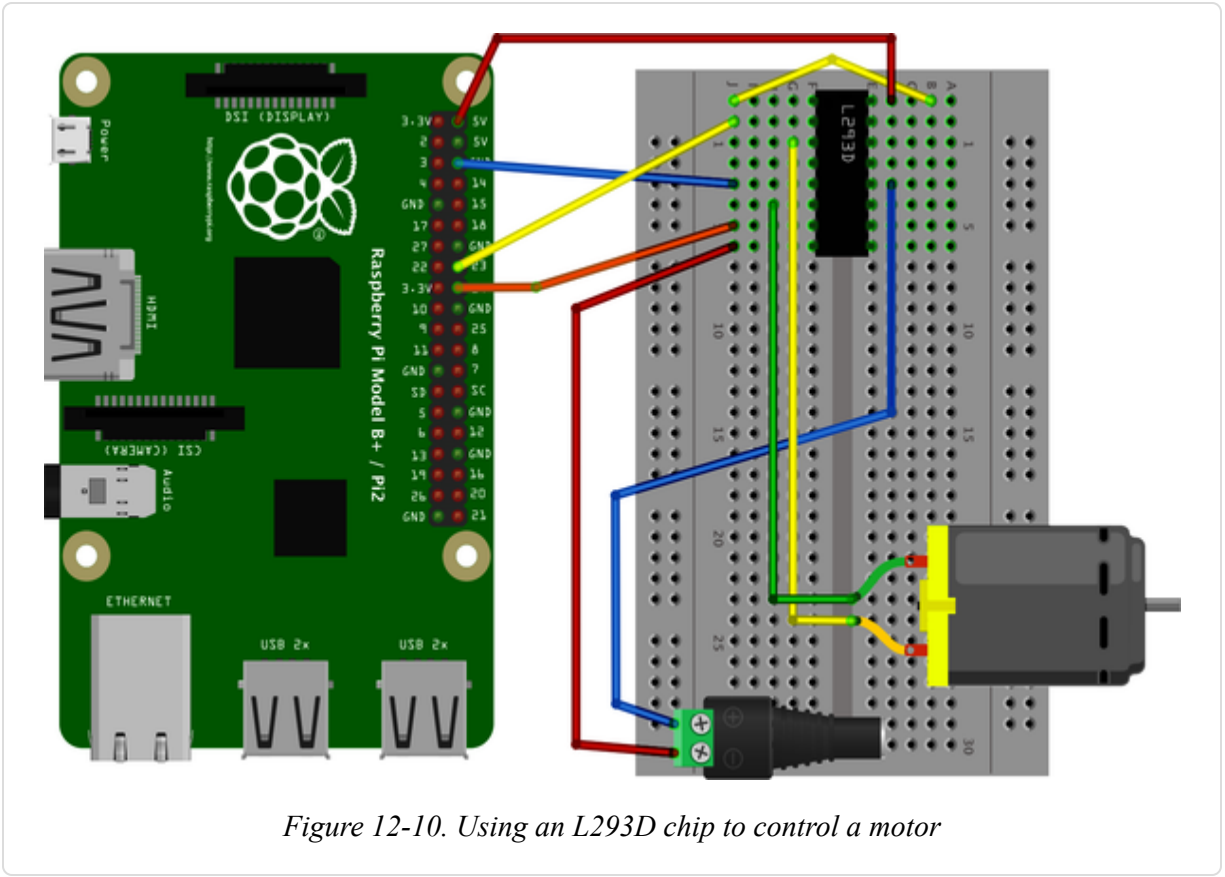


Figure 12-10. Using an L293D chip to control a motor

Make sure the chip is facing the proper direction: it has a notch at the top, which is the end that should be at the top of the breadboard.

The test program for this recipe (*ch\_12\_motor\_control.py*) allows you to enter the letter *f* or *r* and then a single digit between 0 and 9. The motor will then go either forward or backward at a speed specified by the digit—0 for stopped, 9 for full speed:

```
$ python3 ch_12_motor_control.py
Command, f/r 0..9, E.g. f5 :f5
Command, f/r 0..9, E.g. f5 :f1
Command, f/r 0..9, E.g. f5 :f2
Command, f/r 0..9, E.g. f5 :r2
```

Open an editor and paste in the following code. As with all the program examples in this book, you can also download it (see [Recipe 3.22](#)).

This program uses the command line, so you can run it from SSH or the Terminal:

```
from gpiozero import Motor

motor = Motor(forward=23, backward=24)

while True:
    cmd = input("Command, f/r 0..9, E.g. f5 :")
    direction = cmd[0]
    speed = float(cmd[1]) / 10.0
    if direction == "f":
        motor.forward(speed=speed)
    else:
        motor.backward(speed=speed)
```

gpiozero conveniently has a class called `Motor` that we can use to control both the speed and direction of a single DC motor. When you create an instance of the class, you need to specify the `forward` and `backward` control pins.

The `forward` and `backward` methods of `Motor` take an optional parameter of speed between 0 and 1, where 1 is full speed.

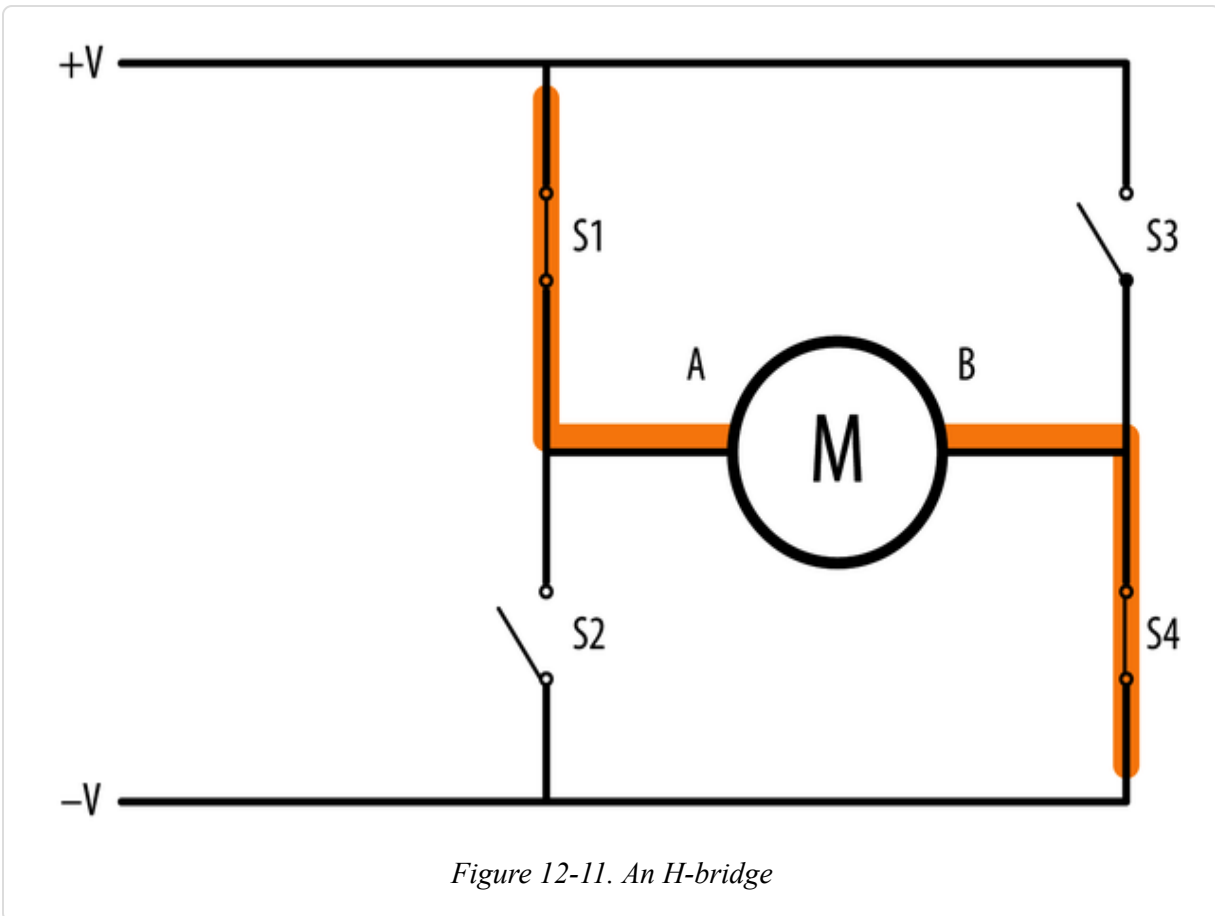
## Discussion

The `Motor` class of `gpiozero` hides the complexity of the H-bridge's hardware.

**Figure 12-11** shows how an H-bridge works, using switches rather than transistors or a chip. By reversing the polarity across the motor, an H-bridge also reverses the direction in which the motor turns.

In **Figure 12-11**, S1 and S4 are closed and S2 and S3 are open. This allows current to flow through the motor, with terminal A being positive and terminal B being negative. If we were to reverse the switches, so that S2 and S3 are closed and S1 and S4 are open, B would be positive and A would be negative, and the motor would turn in the opposite direction.

However, you might have spotted a danger with this circuit. If by some chance S1 and S2 are both closed, the positive supply will be directly connected to the negative supply, and you will have a short circuit. The same is true if S3 and S4 are both closed at the same time.



Although you can use individual transistors to make an H-bridge, it is simpler to use an H-bridge integrated circuit (IC) such as the L293D. This chip actually has two H-bridges in it, so you can use it to control two motors. It also has logic to ensure that the equivalent of closing both S1 and S2 cannot happen.

The L293D uses two control pins for each of the two motor control channels: a *forward* pin and a *backward* pin. If the forward pin (23) is high and the backward pin (24) is low, the motor will turn in one direction. If those two pins are reversed, the motor will turn in the opposite direction.

As an alternative to using an L293D on a breadboard, very low-cost modules are available from eBay that include a L293D on a printed circuit board (PCB) with screw terminals to attach motors and header pins to link directly to the Raspberry Pi GPIO connector. You can find high-power motor controller modules that operate on the same principles but at much higher currents, even up to 20A or more. [Pololu](#) has an impressive range of such motor controller boards.

## See Also

You also can use the Adafruit Stepper Motor HAT ([Recipe 12.8](#)) to control the speed and direction of a DC motor.

Check out the [L293D datasheet](#) and the [SparkFun Motor Driver module product page](#).

For more information on using a breadboard and jumper wires with the Raspberry Pi, see [Recipe 10.9](#).

## 12.6 Using a Unipolar Stepper Motor

### Problem

You want to drive a five-lead unipolar stepper motor using a Raspberry Pi.

### Solution

Use a ULN2803 Darlington driver chip on a breadboard.

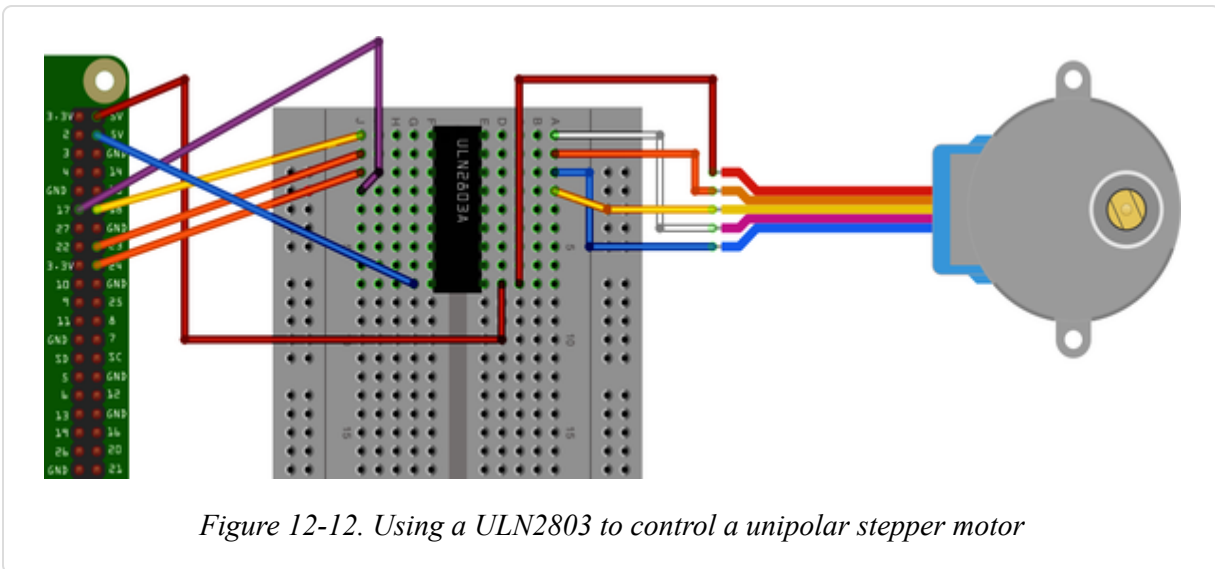
Stepper motors fit somewhere between DC motors and servomotors in the world of motor technologies. Like a regular DC motor, they can rotate continuously, but you can also very accurately position them by moving them a step at a time in either direction.

To make this recipe, you'll need the following:

- 5V, five-pin unipolar stepper motor (see [“Miscellaneous”](#))

- ULN2803 Darlington driver IC (see “Integrated Circuits”)
- Breadboard and jumper wires (see “Prototyping Equipment and Kits”)

Figure 12-12 shows the wiring diagram for using a ULN2803. Note that you can use the chip to drive two such motors. To drive a second stepper motor, you need to connect four more control pins from the GPIO connector to pins 5 to 8 of the ULN2803, and then you connect the second motor’s four pins to pins 11 to 14 of the ULN2803.



The 5V supply from the GPIO connector can work acceptably with a small stepper motor. If you experience problems with the Raspberry Pi crashing or need to use a bigger stepper motor, use a separate supply for the power to the motor (pin 10 of the ULN2803).

Open an editor and paste in the following code (*ch\_12\_stepper.py*). This program uses the command line, so you can run it from SSH:

```
from gpiozero import Motor
import time

coil1 = Motor(forward=18, backward=23, pwm=False)
coil2 = Motor(forward=24, backward=17, pwm=False)

forward_seq = ['FF', 'BF', 'BB', 'FB']
reverse_seq = list(forward_seq) # to copy the list
reverse_seq.reverse()
```

```

def forward(delay, steps):
    for i in range(steps):
        for step in forward_seq:
            set_step(step)
            time.sleep(delay)

def backwards(delay, steps):
    for i in range(steps):
        for step in reverse_seq:
            set_step(step)
            time.sleep(delay)

def set_step(step):
    if step == 'S':
        coil1.stop()
        coil2.stop()
    else:
        if step[0] == 'F':
            coil1.forward()
        else:
            coil1.backward()
        if step[1] == 'F':
            coil2.forward()
        else:
            coil2.backward()

while True:
    set_step('S')
    delay = input("Delay between steps (milliseconds)?")
    steps = input("How many steps forward? ")
    forward(int(delay) / 1000.0, int(steps))
    set_step('S')
    steps = input("How many steps backwards? ")
    backwards(int(delay) / 1000.0, int(steps))

```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

When you run the program, you are prompted for a delay between steps. This should be 2 or more. You are then prompted for the number of steps in each direction:

```

$ python3 ch_12_stepper.py
Delay between steps (milliseconds)?2
How many steps forward? 100

```

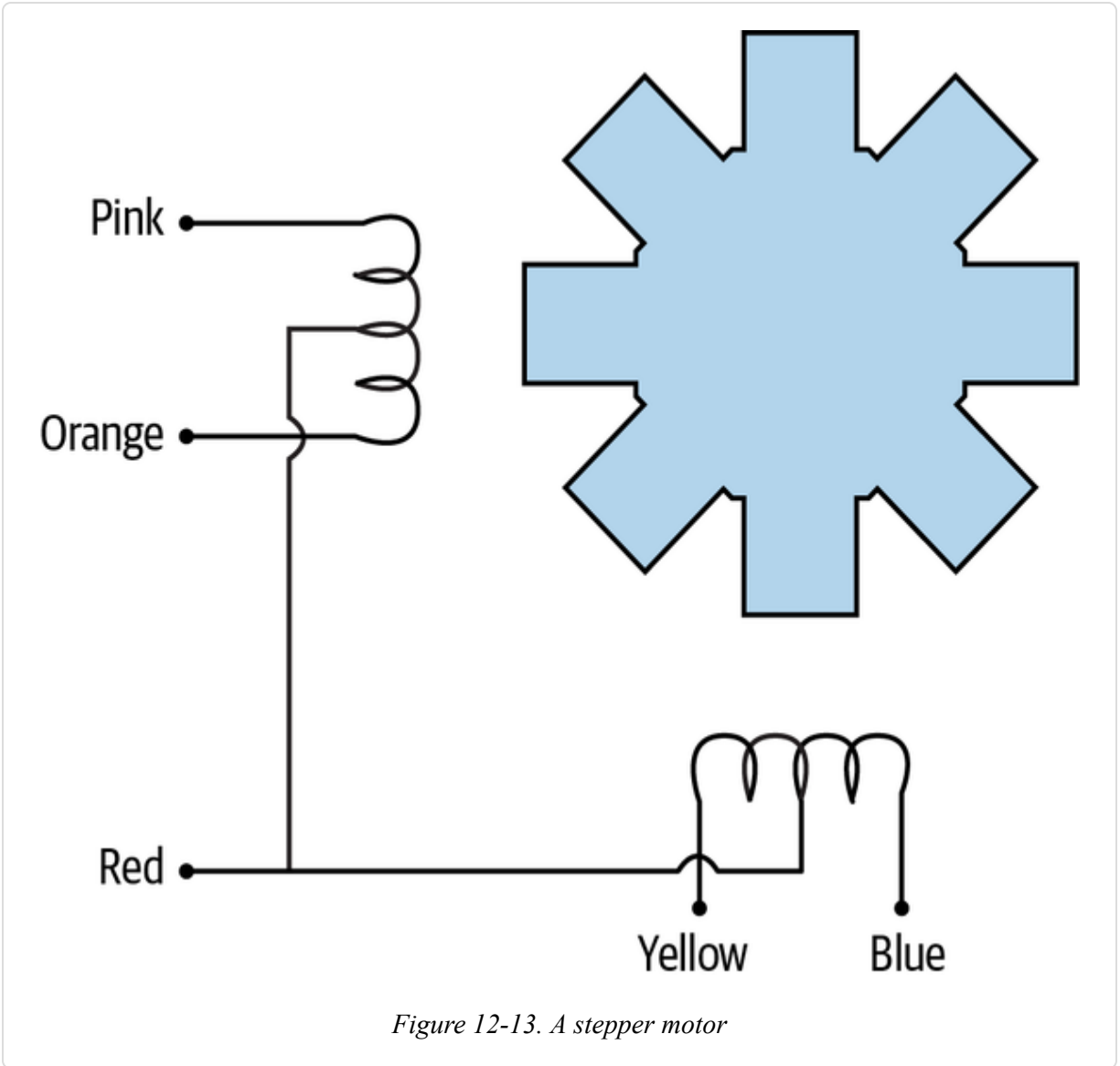
```
How many steps backwards? 100  
Delay between steps (milliseconds)?10  
How many steps forward? 50  
How many steps backwards? 50  
Delay between steps (milliseconds)?
```

This code is explained in the Discussion section that follows, because it helps to know a bit more about how stepper motors work to follow the code.

## Discussion

Stepper motors use a cogged rotor with alternating north and south poles and electromagnets to nudge the wheel around a *step* at a time (Figure 12-13). Note that the colors of the leads will vary.

Energizing the coils in a certain order drives the motor around. The number of steps that the stepper motor has in a 360-degree rotation is actually the number of teeth on the rotor.



The two coils are each controlled by an instance of the `gpiozero` class `Motor` and are called `coil1` and `coil2`.

The program uses a list of strings to represent each of the four energization stages that make up a single step:

```
forward_seq = ['FF', 'BF', 'BB', 'FB']
```

Each pair of letters indicates the current direction for `coil1` and `coil2`: either forward or backward. So, looking for a moment at [Figure 12-13](#) and



assuming that the common Red connection is to GND, the letter F for the Pink-Orange coil might make Pink high and Orange low, whereas B would reverse this.

The sequence for rotating the motor in the opposite direction is just the reverse of the sequence for moving forward.

You can use the `forward` and `backward` functions in your programs to step the motor back and forth. The first argument to either function is the delay in milliseconds between each part of the step sequence. The minimum value for this depends on the motor you use. If it's too small, the motor will not turn. Typically, two milliseconds or more will be fine. The second parameter is the number of steps to take:

```
def forward(delay, steps):
    for i in range(steps):
        for step in forward_seq:
            set_step(step)
            time.sleep(delay)
```

The `forward` function has two nested `for` loops. The outer loop repeats for the number of steps, and the inner one iterates over the sequence of motor activations, calling `set_step` for each in sequence:

```
def set_step(step):
    if step == 'S':
        coil1.stop()
        coil2.stop()
    else:
        if step[0] == 'F':
            coil1.forward()
        else:
            coil1.backward()
        if step[1] == 'F':
            coil2.forward()
        else:
            coil2.backward()
```

The `set_step` function sets each coil's polarity, depending on the message supplied as its `step` argument. The command S stops the power

to both coils so that we can avoid using current when the motor isn't moving. If the first letter is an F, then `coil1` is set to `forward`; otherwise it is set to `backward`. `coil2` is set in the same way but using the second letter of `step`.

The main loop sets the `step` to S between moving forward and backward, to deactivate both coils when the motor is not actually turning. Otherwise, one of the coils might be left on, causing the motor to unnecessarily draw current.

## See Also

If you have a four-wire bipolar stepper motor, see [Recipe 12.7](#).

For more information, see the [Wikipedia article on stepper motors](#), where you will also see the different types and how they work, as well as find a nice, animated explanation of the activation pattern for driving the motor.

For information on using servomotors, see [Recipe 12.1](#); for information on controlling DC motors, see [Recipes 12.4](#) and [12.5](#).

For more information on using a breadboard and jumper wires with the Raspberry Pi, see [Recipe 10.9](#).

## 12.7 Using a Bipolar Stepper Motor

### Problem

You want to drive a four-lead bipolar stepper motor using a Raspberry Pi.

### Solution

Use an L293D H-bridge driver chip. An H-bridge is required to drive a bipolar stepper motor because, as the word *bipolar* suggests, the direction of current across the windings needs to be reversed, rather like driving a DC motor in both directions (see [Recipe 12.5](#)).

To make this recipe, you'll need the following:

- 12V, four-pin bipolar stepper motor (see “Miscellaneous”)
- L293D H-bridge IC (see “Integrated Circuits”)
- Breadboard and jumper wires (see “Prototyping Equipment and Kits”)

The motor used here, a 12V one, is somewhat larger than the unipolar stepper motor example in [Recipe 12.6](#). The power for the motor itself is therefore supplied from an external power supply rather than from the Raspberry Pi. See the wiring diagram in [Figure 12-14](#).

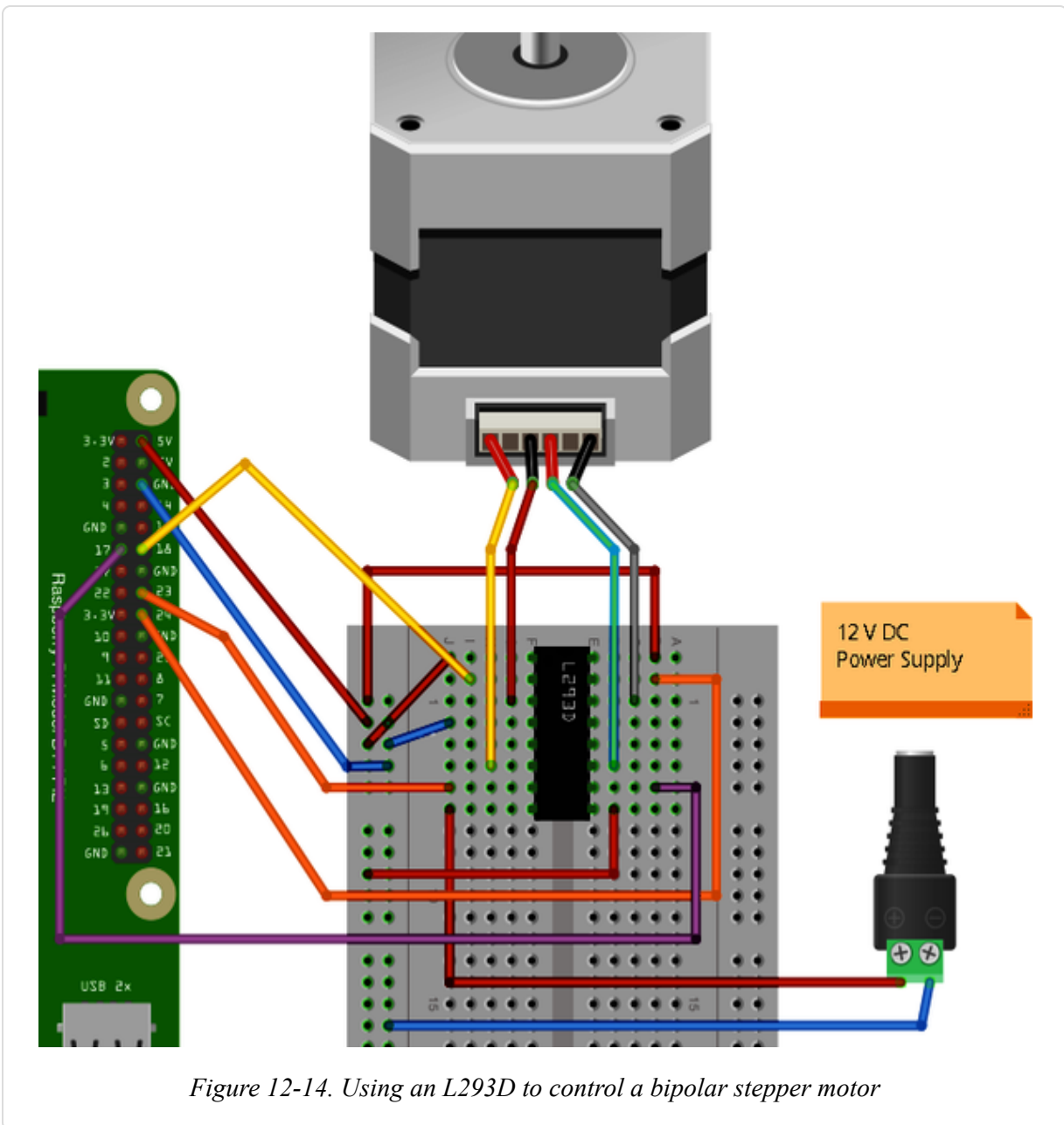


Figure 12-14. Using an L293D to control a bipolar stepper motor

## Discussion

A bipolar stepper motor is just like the unipolar version shown in [Figure 12-13](#), except that the red central tapping connections to the coils are not present. The same energization pattern will work just as well on both variants, but for a bipolar motor, the direction of the current in the whole of the coil must be reversible; hence, two H-bridges are necessary.

You can use the same `ch_12_stepper.py` program to control this stepper (see [Recipe 12.6](#)). The design uses both H-bridges of the L293D, so you need one of these chips for each motor that you want to control.

## See Also

If the type of stepper motor you have is a five-wire unipolar stepper motor, see [Recipe 12.6](#).

For more information on stepper motors—the different types and how they work—see [Wikipedia](#), where you will also find a nice animated explanation of the activation pattern for driving the motor.

For information on using servomotors, see [Recipe 12.1](#); for information on controlling DC motors, see [Recipes 12.4](#) and [12.5](#).

For more information on using a breadboard and jumper wires with the Raspberry Pi, see [Recipe 10.9](#).

## 12.8 Using a Stepper Motor HAT to Drive a Bipolar Stepper Motor

### Problem

You want to control multiple bipolar stepper motors using a single interface board.

### Solution

Use an Adafruit Stepper Motor HAT.

This board is capable of driving two bipolar stepper motors. [Figure 12-15](#) shows the board with one bipolar stepper motor, with one coil connected to the M1 terminals and the other coil to the M2 terminals. The power for the motors is supplied separately at the screw terminals on the right.

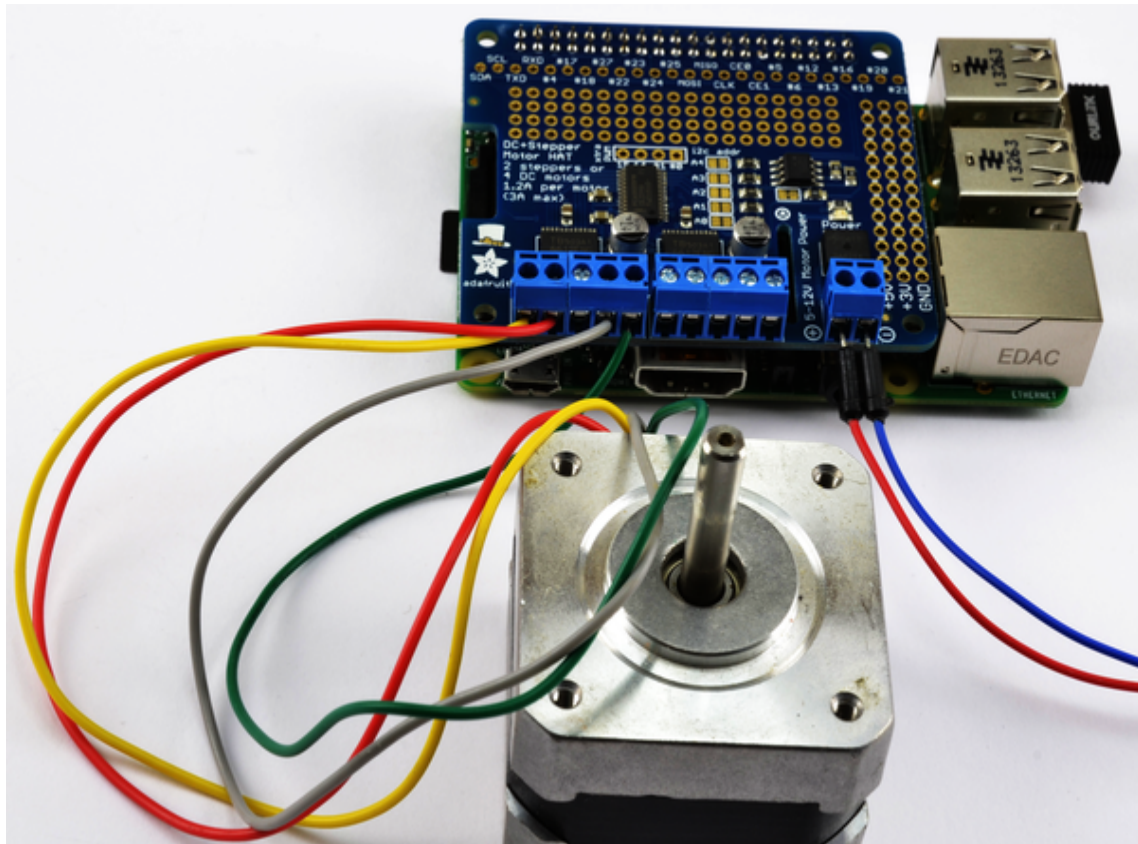
## I2C Buses

If you followed [Recipe 10.17](#) to make your own HAT and enabled the I2C bus 0 as described there, you will need to reverse the change to `/boot/config.txt` because the Adafruit autodetects the I2C bus to use and will detect the wrong one if bus 0 is enabled.

In `/boot/config.txt`, delete or comment out this line (by putting a # at the beginning of it):

```
dtparam=i2c_vc=on
```

After you do this, reboot your Raspberry Pi.



*Figure 12-15. Using an Adafruit Stepper Motor HAT to control a bipolar stepper motor*

This HAT uses I2C, so make sure that you have I2C enabled ([Recipe 10.4](#)). This board is supported by an [excellent Adafruit tutorial](#).

## Discussion

When you run the program supplied in the Adafruit tutorial, the motor begins to turn, and the program loops around four different modes of stepping.

## See Also

For a discussion of the HAT standard and how to make your own HAT, see [Recipe 10.17](#).

For more information on using this HAT and its accompanying library, see <https://oreil.ly/3a4Jm>.

To use an L293D to control a stepper motor, see [Recipe 12.7](#).

# Chapter 13. Digital Inputs

---

## 13.0 Introduction

In this chapter, we look at recipes for using digital components such as switches and keypads. This chapter also covers modules that have a digital output that can be connected to a Raspberry Pi general-purpose input/output (GPIO) acting as an input.

Many of the recipes require the use of a solderless breadboard and jumper wires (see [Recipe 10.9](#)).

## 13.1 Connecting a Push Switch

### Problem

You want to connect a switch to your Raspberry Pi so that when you press it, some Python code is run.

### Solution

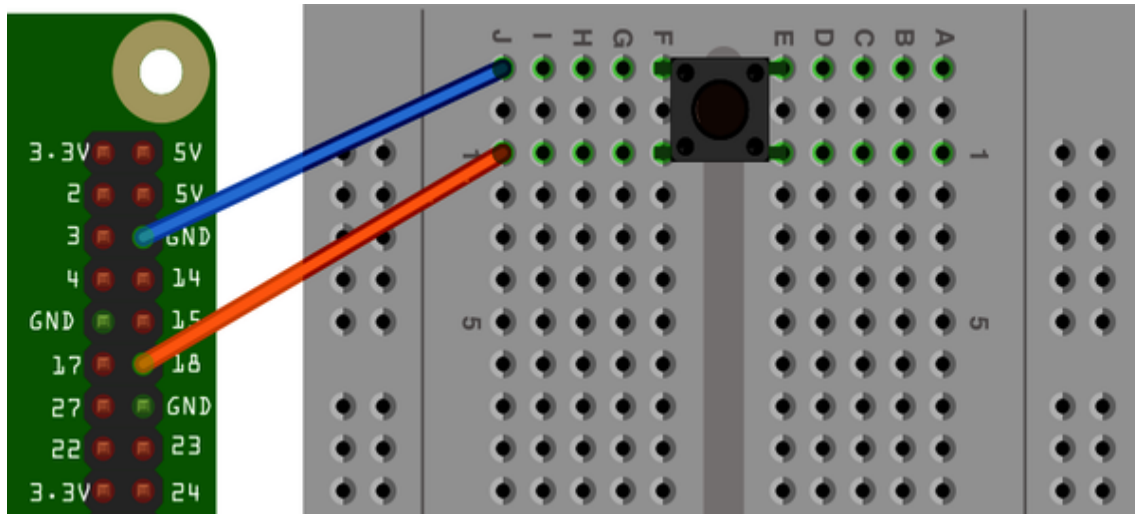
Connect a switch to a GPIO pin and use the `gpiozero` library in your Python program to detect the button press.

To make this recipe, you'll need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- Tactile push switch (see [“Miscellaneous”](#))

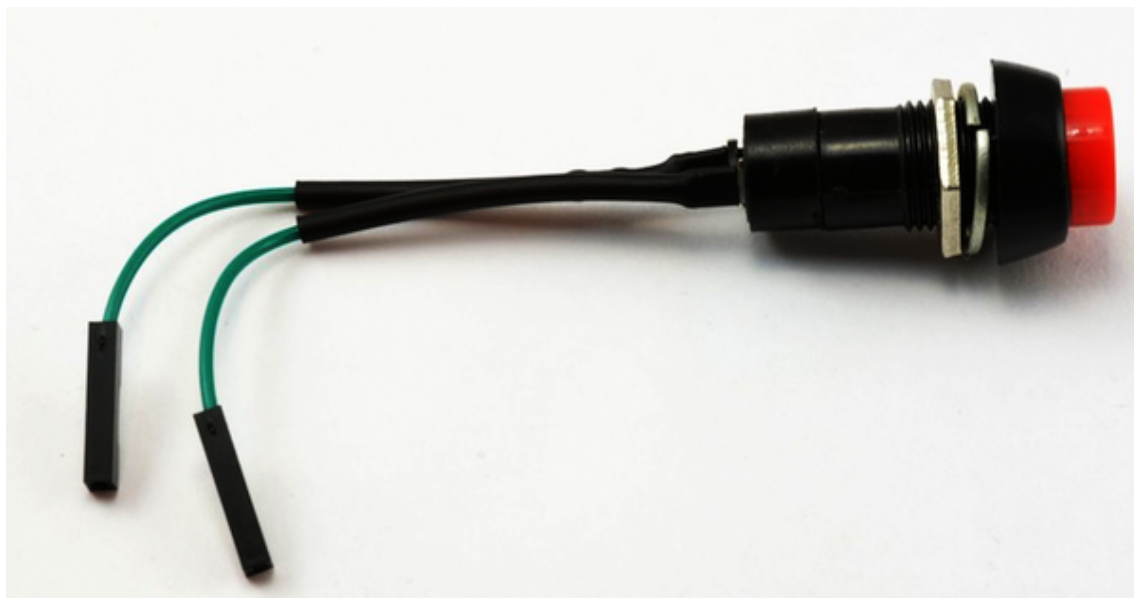
[Figure 13-1](#) shows how to connect a tactile push switch using a breadboard and jumper wires.





*Figure 13-1. Connecting a push switch to a Raspberry Pi*

An alternative to using a breadboard and tactile switch is to use a Squid Button (Figure 13-2). This is a push switch with female header leads soldered to the end, which you can directly connect to the GPIO connector (Recipe 10.11).



*Figure 13-2. A Squid Button*

Open an editor and paste in the following code (*ch\_13\_switch.py*):

```

from gpiozero import Button

button = Button(18)

while True:
    if button.is_pressed:
        print("Button Pressed")

```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

This is what you will see when you run the program and as you press the button:

```

pi@raspberrypi ~ $ python3 ch_13_switch.py
Button Pressed
Button Pressed
Button Pressed
Button Pressed

```

In fact, the “Button Pressed” messages will probably go shooting off the bottom of the screen. This is because the program checks very frequently for button presses. Another problem with this code is that while it is watching for button presses, it can’t be getting on with other things.

We can improve on this code, both to do something only once when pressed and to allow other things to be going on until a button is pressed. You will find these changes in the following code (*ch\_13\_switch\_2.py*):

```

from gpiozero import Button
from time import sleep

def do_stuff():
    print("Button Pressed")

button = Button(18)
button.when_pressed = do_stuff

while True:
    print("Busy doing other stuff")
    sleep(2)

```

When you run this program, you will see output like this:

```
$ python3 ch_13_switch_2.py
Busy doing other stuff
Busy doing other stuff
Button Pressed
Busy doing other stuff
Busy doing other stuff
```

When you press the button, the function `do_stuff` is run, irrespective of what the program is otherwise doing. This approach is called using *interrupts* and is often used in programs that need to trigger actions when a button is pressed, but also need to be doing other things.

Note that this line:

```
button.when_pressed = do_stuff
```

includes `do_stuff` without `()` on the end. This is because we are referring to the function, not actually calling it, until the interrupt occurs. In other words, we are telling the interrupt handler which function to call when there is an interrupt, not asking it to call the function immediately.

## Discussion

Notice that the switch is wired so that when it is pressed, it connects pin 18, which is configured as an input to GND.

You might expect the push switch to have just two connections, which are either open or closed. Although some of these tactile push switches do have just two connections, most have four to provide mechanical strength.

**Figure 13-3** shows how these connections are arranged.

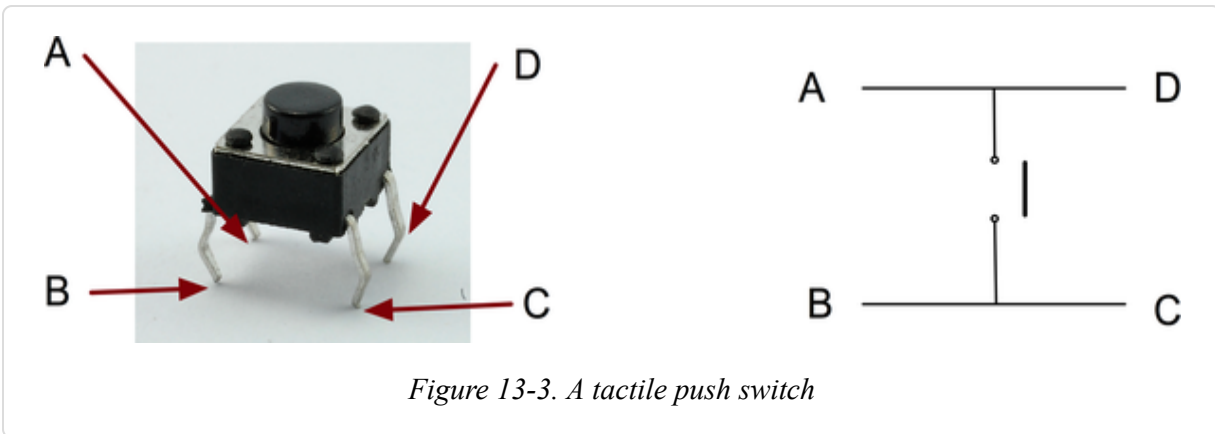


Figure 13-3. A tactile push switch

Actually, there are really only two electrical connections, because inside the switch package, pins B and C are connected together, as are A and D.

## See Also

For more information on using a breadboard and jumper wires with the Raspberry Pi, see [Recipe 10.9](#).

To debounce a switch, see [Recipe 13.5](#).

To use external pull-up or pull-down resistors, see [Recipe 13.6](#).

## 13.2 Toggling with a Push Switch

### Problem

You want to create a push switch that toggles something between on and off each time you press it.

### Solution

Read the last *state* of the button (that is, whether the button is on or off) and invert that value each time the button is pressed.

The following example toggles an LED on and off as you press the switch.

To make this recipe, you will need the following:

- Breadboard and jumper wires (see “Prototyping Equipment and Kits”)
- Tactile push switch (see “Miscellaneous”)
- LED (see “OptoElectronics”)
- 470Ω resistor (see “Resistors and Capacitors”)

Figure 13-4 shows how to connect a tactile push switch and LED, using a breadboard and jumper wires.

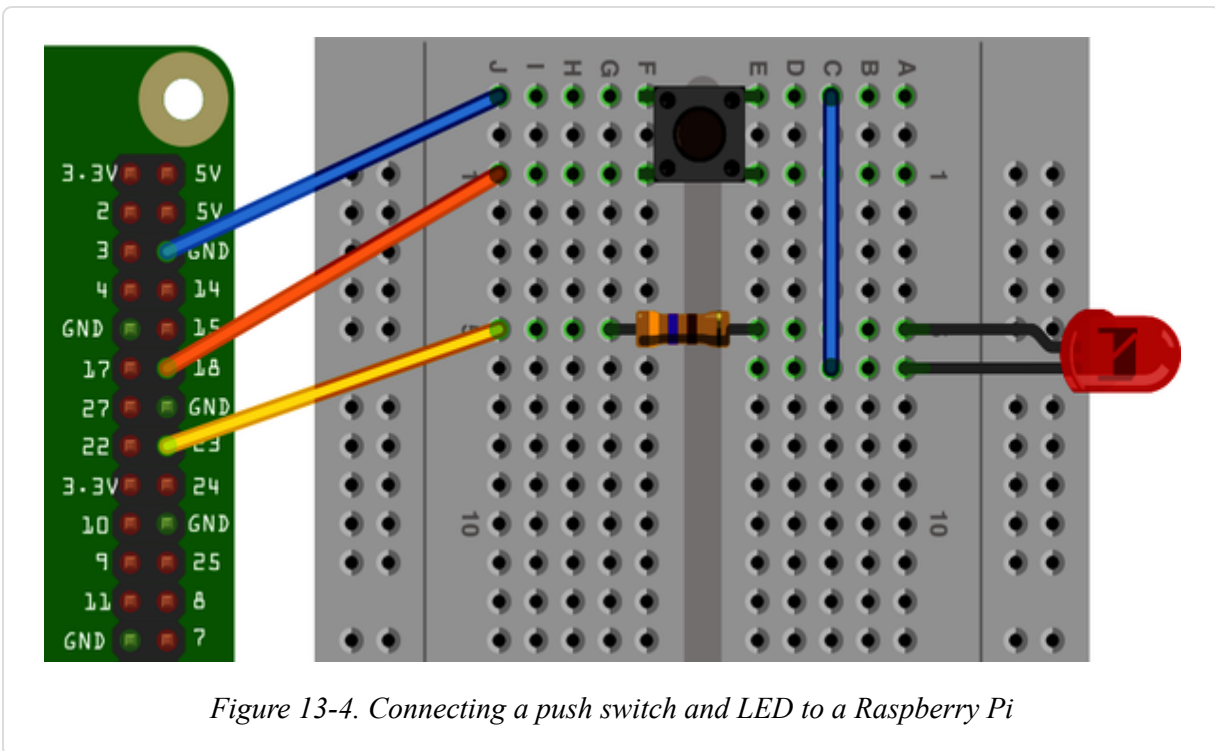


Figure 13-4. Connecting a push switch and LED to a Raspberry Pi

In addition to the male-to-female jumper wires connecting the Raspberry Pi to the breadboard, you also need one male-to-male jumper wire or solid core wire.

As an alternative to a breadboard and separate components, you could use a Raspberry Squid (Recipe 10.10) and a Squid Button (Recipe 13.1).

Open an editor and paste in the following code (*ch\_13\_switch\_on\_off.py*):

```
from gpiozero import Button, LED
from time import sleep

led = LED(23)
```

```
def toggle_led():
    print("togglng")
    led.toggle()

button = Button(18)
button.when_pressed = toggle_led

while True:
    print("Busy doing other stuff")
    sleep(2)
```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

The program is based on `ch_13_switch_2.py` and again uses interrupts so that the program can be getting on with other things until a button press happens.

When the button is pressed, the function `toggle_led` is called. This *toggles* the LED; that is, if it's *on*, it turns it *off*, and if it's *off*, it turns it *on*.

## Discussion

Depending on the quality of your switch, you may have noticed that sometimes the LED doesn't toggle, but that two or more "togglng" messages appear in the Terminal. This is due to something called *switch bounce*, which we will discuss further in [Recipe 13.5](#).

## See Also

For the documentation on the `gpiozero Button` class, see <https://oreil.ly/SXIZ9>.

# 13.3 Using a Two-Position Toggle or Slide Switch

## Problem

You want to connect a two-position toggle or slide switch to your Raspberry Pi and be able to find the position of the switch in your Python program.

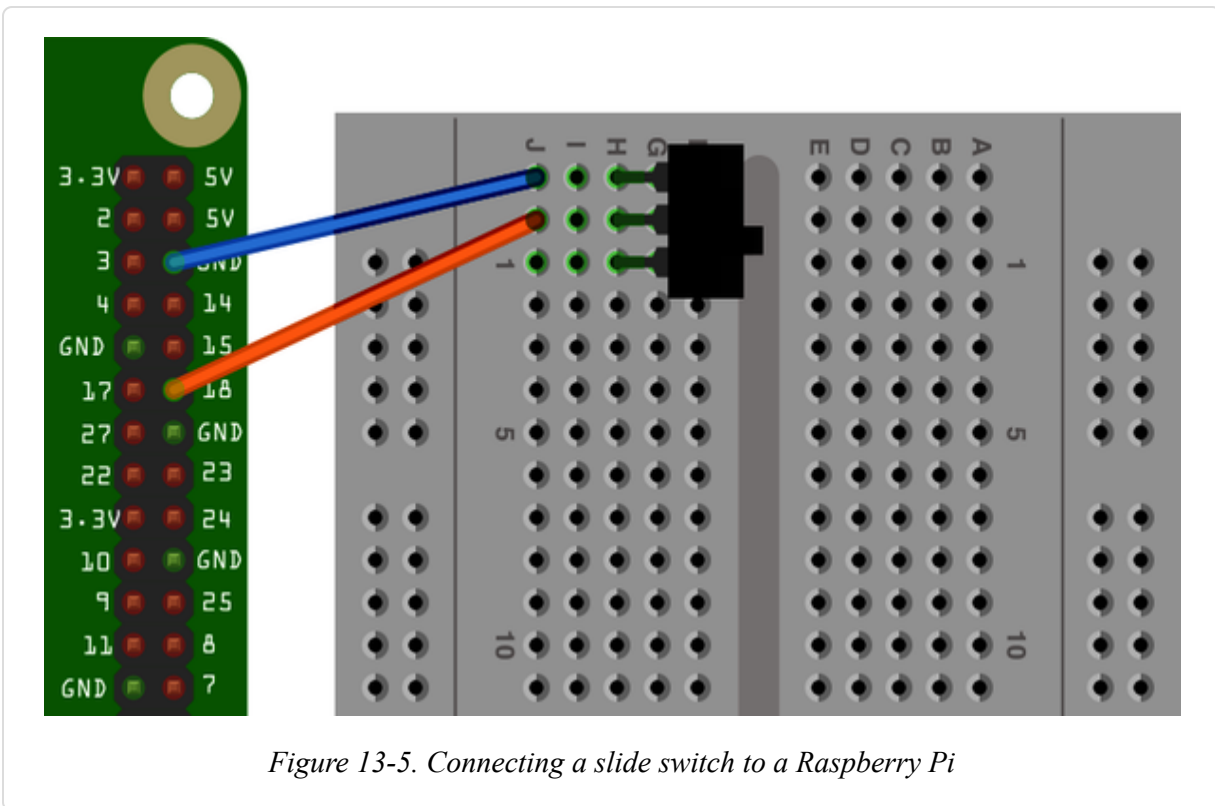
## Solution

Use the switch as you would a tactile push switch ([Recipe 13.1](#)): just connect the center and one end contact ([Figure 13-5](#)).

To make this recipe, you will need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- Miniature toggle or slide switch (see [“Miscellaneous”](#))

The same code you used in [Recipe 13.1](#) works with this arrangement.



*Figure 13-5. Connecting a slide switch to a Raspberry Pi*

## Discussion

These types of slide switches are useful because you can see the position they are set to without the need for some additional indicator like an LED. However, they are more fragile and a little more expensive than the tactile

push switches that are used more and more in consumer electronics because they can sit behind a nicer-looking plastic button.

## See Also

To use a three-position switch with a center-off position, see [Recipe 13.4](#).

# 13.4 Using a Center-Off Toggle or Slide Switch

## Problem

You want to connect a three-position (center-off) toggle switch to your Raspberry Pi and be able to find the position of the switch in your Python program.

## Solution

Connect the switch to two GPIO pins, as shown in [Figure 13-6](#), and use the `gpiozero` library in your Python program to detect the position of the switch.

To make this recipe, you will need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- Miniature center-off three-position toggle switch (see [“Miscellaneous”](#))



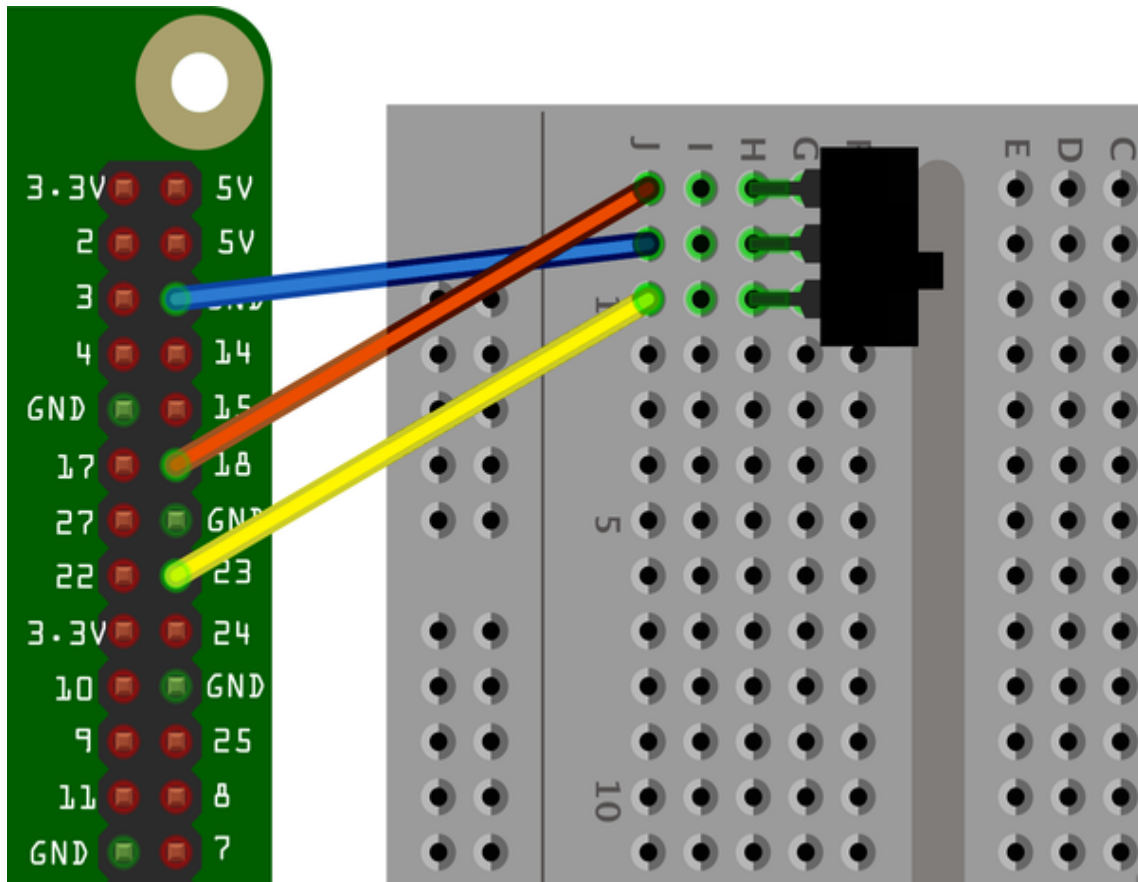


Figure 13-6. Connecting a three-position switch to a Raspberry Pi

The common (center) connection of the switch is connected to GND, and each of the two ends of the switch are connected to a GPIO pin.

Open an editor and paste in the following code (*ch\_13\_switch\_3\_pos.py*):

```
from gpiozero import Button

switch_top = Button(18)
switch_bottom = Button(23)

switch_position = "unknown"

while True:
    new_switch_position = "unknown"
    if switch_top.is_pressed:
        new_switch_position = "top"
    elif switch_bottom.is_pressed:
        new_switch_position = "bottom"
```

```
else:
    new_switch_position = "center"

if new_switch_position != switch_position:
    switch_position = new_switch_position
    print(switch_position)
```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

Run the program, and as you move the switch from top to center to bottom, the position of the switch is reported every time it changes:

```
$ python3 ch_13_switch_3_pos.py
center
top
center
bottom
```

## Discussion

The program sets up two inputs as separate buttons. Inside the loop, both button states are read, and the three conditions of the `if`, `elif`, and `else` structure determine the position of the switch, assigning the value to a variable called `new_switch_position`. If this differs from the previous value, the switch position is printed.

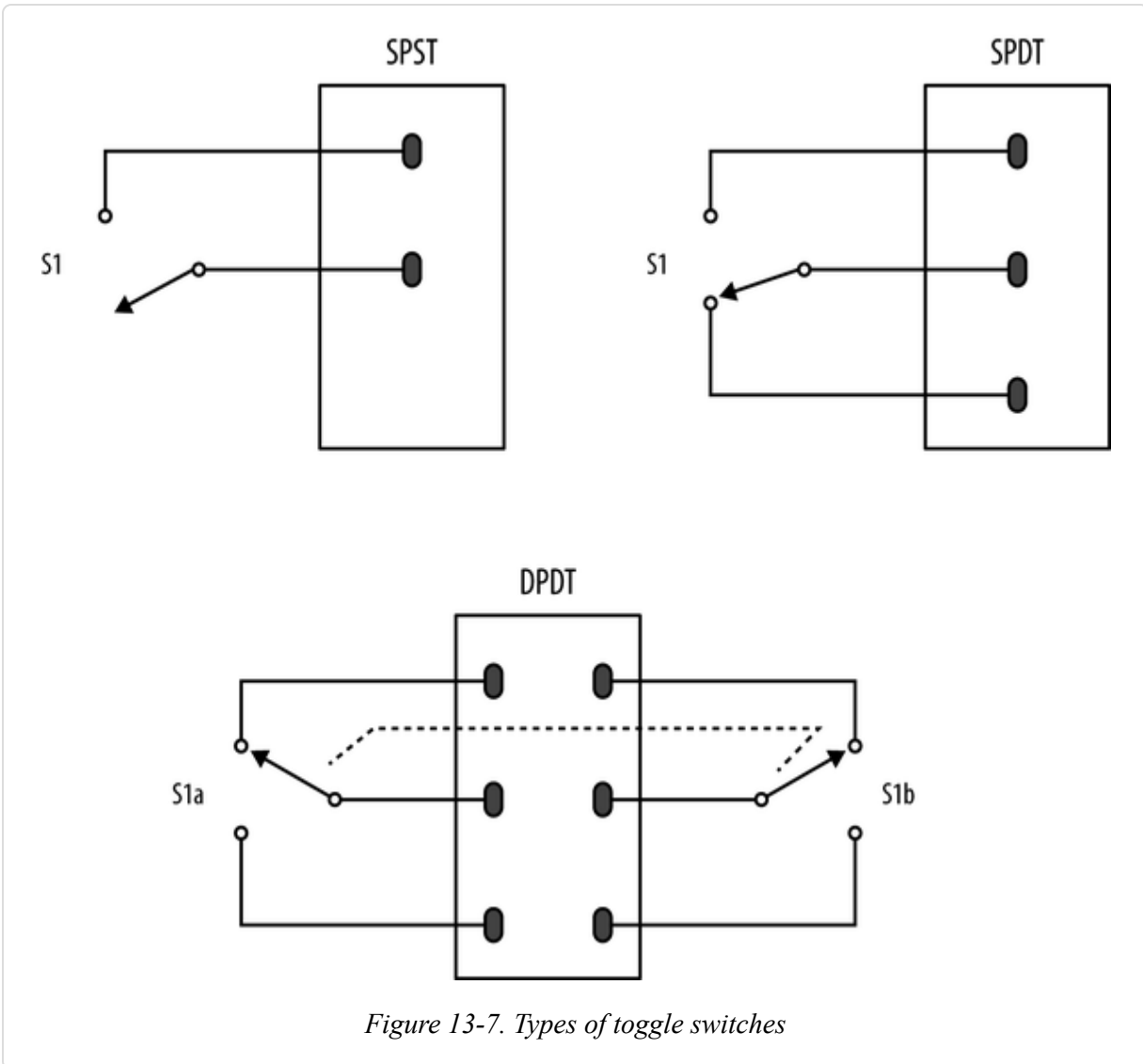
You will find a wide range of types of toggle switches. Some will be described as DPDT, SPDT, SPST, or SPST-momentary-on, and so on. The meaning of these letters is as follows:

- D: Double
- S: Single
- P: Pole
- T: Throw

Thus, a DPDT switch is double pole, double throw. The word *pole* refers to the number of separate switch contacts that are controlled from the one mechanical lever. Thus, a double pole switch can switch two things on and

off. A single throw switch can only open or close a single contact (or two contacts if it is double pole). However, a double throw switch can connect the common contact to one of two other contacts.

Figure 13-7 shows the most common types of switches.



## See Also

For more information on how `if` statements work, see [Recipe 5.20](#).

For more information on toggle switches, see <https://oreil.ly/uDigs>.

For the most basic switch recipe, see [Recipe 13.1](#).

## 13.5 Debouncing a Button Press

### Problem

Sometimes when you press the button on a switch, the expected action happens more than once because the switch contacts *bounce* (Figure 13-8). In that case, you want to write code that *debounces* the switch.

### Solution

The `gpiozero.Button` class includes code that deals with bouncing of switch contacts. However, by default this is turned off. You can change it when you create a button instance using the optional `bounce_time` parameter.

See the Discussion for more information on what is going on with switch bouncing. But the basic idea is that when a switch is pressed, the contacts can *bounce*, producing false readings of the switch position. The `bounce_time` parameter determines how long false changes of the switch state should be ignored for.

In [Recipe 13.2](#), for example, you might have noticed that pressing the button often doesn't seem to toggle the LED. This will happen if you get an even number of bounces when you press the button, so that one bounce turns the LED on and a second bounce immediately turns it off again (within a fraction of a second), with the result that it looks like nothing happened.

You can modify the program `ch_13_switch_on_off.py` to set the debouncing time by adding the `bounce_time` optional parameter where you define the button:

```
from gpiozero import Button, LED
from time import sleep

led = LED(23)

def toggle_led():
```

```
print("toggling")
led.toggle()

button = Button(18, bounce_time=0.1)
button.when_pressed = toggle_led

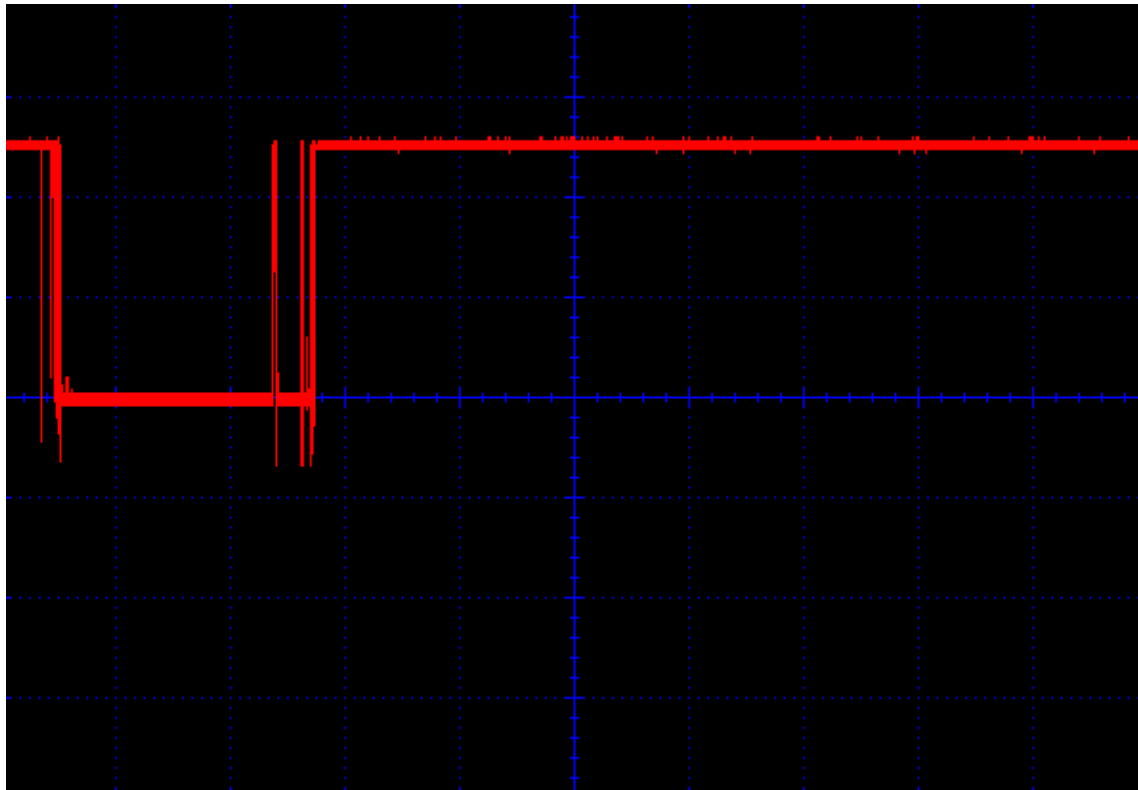
while True:
    print("Busy doing other stuff")
    sleep(2)
```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

In this example, the bounce time is set to 0.1 seconds, which should be more than enough time for the switch contacts to settle.

## Discussion

Switch bouncing occurs on most switches and can be quite severe on some switches, as the oscilloscope trace in [Figure 13-8](#) shows.



*Figure 13-8. Contact bounce with a poor switch*

You can see that there is contact bounce both as the switch closes and when it is released. Most switches are not as bad as this one.

## See Also

For the basics of connecting a button, see [Recipe 13.1](#).

## 13.6 Using an External Pull-Up Resistor

### Problem

You want to run a long wire from the Raspberry Pi to the switch, but you are getting some false readings on the input pin.

### Solution

The Raspberry Pi GPIO pins include *pull-up* resistors. When a GPIO pin is being used as a digital input, its pull-up resistor will keep the input high (3.3V) until the input is pulled down to GND, perhaps by a switch. In addition, the pull-up resistor can be turned on and off within your Python program.

These internal pull-up resistors are quite weak (about 40kΩ), which means that if you run a long lead to the switch, or operate in an electrically noisy environment, you might get false triggerings on the digital input. You can overcome this by turning off the internal pull-up and pull-down resistors and using an external pull-up resistor.

**Figure 13-9** shows the use of an external pull-up resistor.

To test out this hardware, you can use the program `ch_13_switch.py`; see **Recipe 13.1**.

## Discussion

The lower the resistance of the resistor, the longer the range of your switch. However, when you press the button, a current flows from 3.3V through the resistor to ground. A 100Ω resistor draws a current of  $3.3\text{V}/100\Omega = 33\text{mA}$ . This is within the safe limit for the 3.3V supply of 50mA for a Raspberry Pi 1, so don't use a lower value than this if you have an older Raspberry Pi. If you are using a newer 40-pin GPIO Raspberry Pi, you could drop this value even further, perhaps to 47Ω.

In almost all cases, a 1kΩ resistor will provide a long range with no problems.

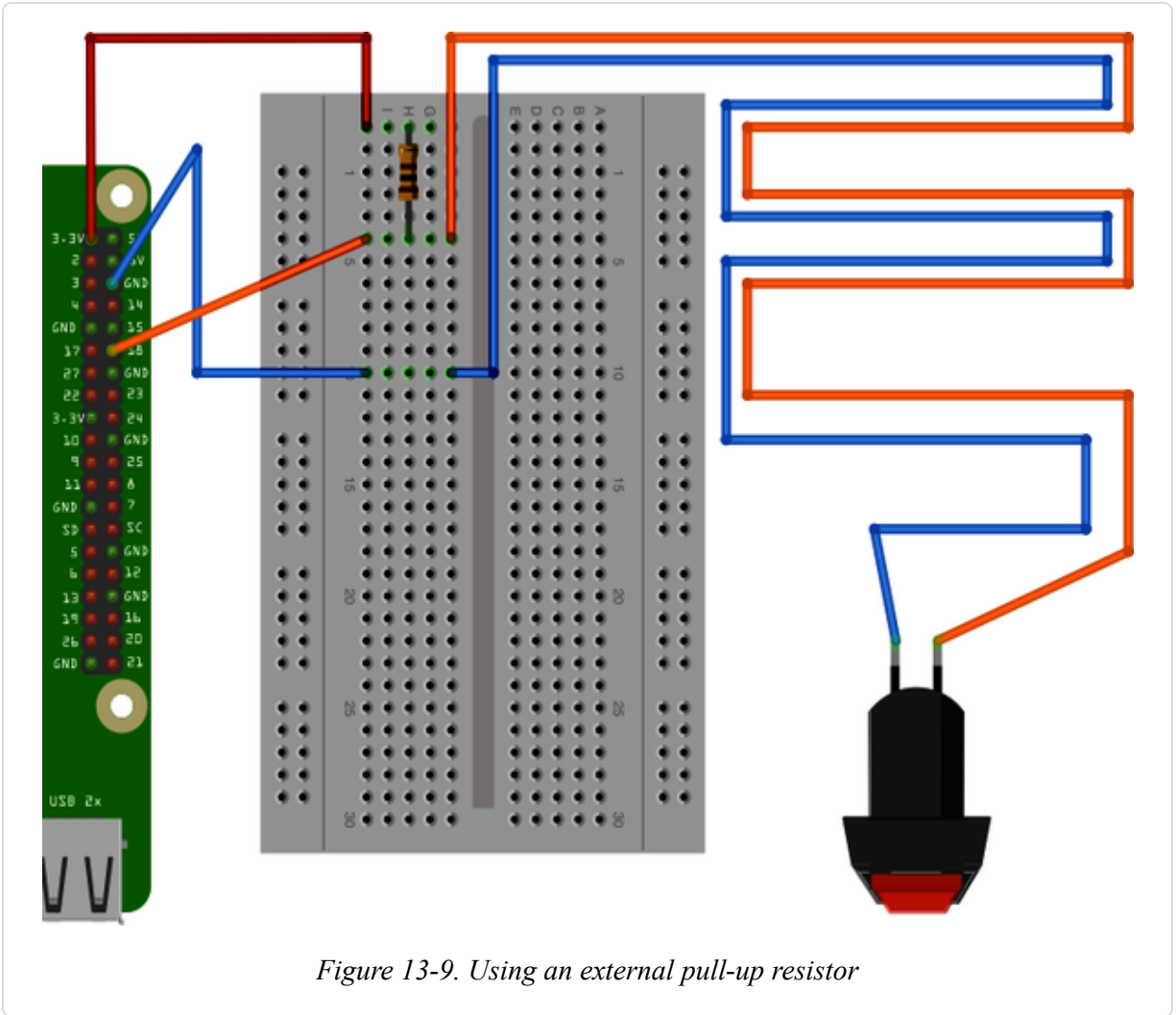


Figure 13-9. Using an external pull-up resistor

## See Also

For the basics of connecting a button, see [Recipe 13.1](#).

## 13.7 Using a Rotary (Quadrature) Encoder

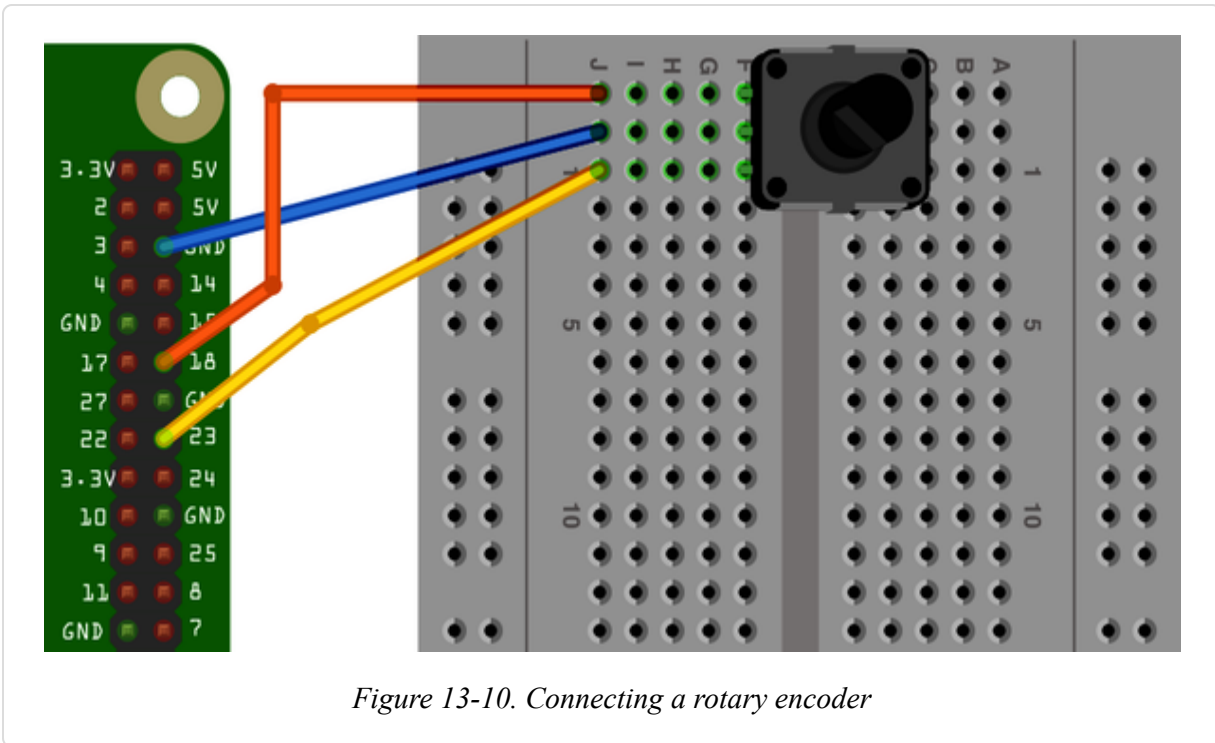
### Problem

You want to detect rotation by using a rotary encoder (a control that you rotate like a volume knob).

### Solution



Use a rotary (quadrature) encoder connected to two GPIO pins (see [Figure 13-10](#)).



*Figure 13-10. Connecting a rotary encoder*

To make this recipe, you will need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- Rotary encoder (quadrature type; see [“Miscellaneous”](#))

This type of rotary encoder is called a *quadrature encoder*, and it behaves like a pair of switches. The sequence in which they open and close as the rotary encoder’s shaft is turned determines the direction of rotation.

The rotary encoder shown in [Figure 13-10](#) has the center lead as the *common* lead and the two leads on either side as A and B. Not all rotary encoders use this layout, so check the pinout on the datasheet for the rotary encoder that you are using. The issue is often confused further because many rotary encoders include a push switch, which will have a separate pair of contacts.

Open an editor and paste in the following code (*ch\_13\_rotary\_encoder.py*):

```

from gpiozero import Button
import time

input_A = Button(18)
input_B = Button(23)

old_a = True
old_b = True

def get_encoder_turn():
    # return -1 (ccl), 0 (no movement), or +1 (cw)
    global old_a, old_b
    result = 0
    new_a = input_A.is_pressed
    new_b = input_B.is_pressed
    if new_a != old_a or new_b != old_b :
        if old_a == 0 and new_a == 1 :
            result = (old_b * 2 - 1)
        elif old_b == 0 and new_b == 1 :
            result = -(old_a * 2 - 1)
    old_a, old_b = new_a, new_b
    time.sleep(0.001)
    return result

x = 0

while True:
    change = get_encoder_turn()
    if change != 0 :
        x = x + change
        print(x)

```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

The test program simply counts up as you turn the rotary encoder clockwise, and down when you rotate it counterclockwise:

```

$ python3 ch_13_rotary_encoder.py
1
2
3
4
5
6
7

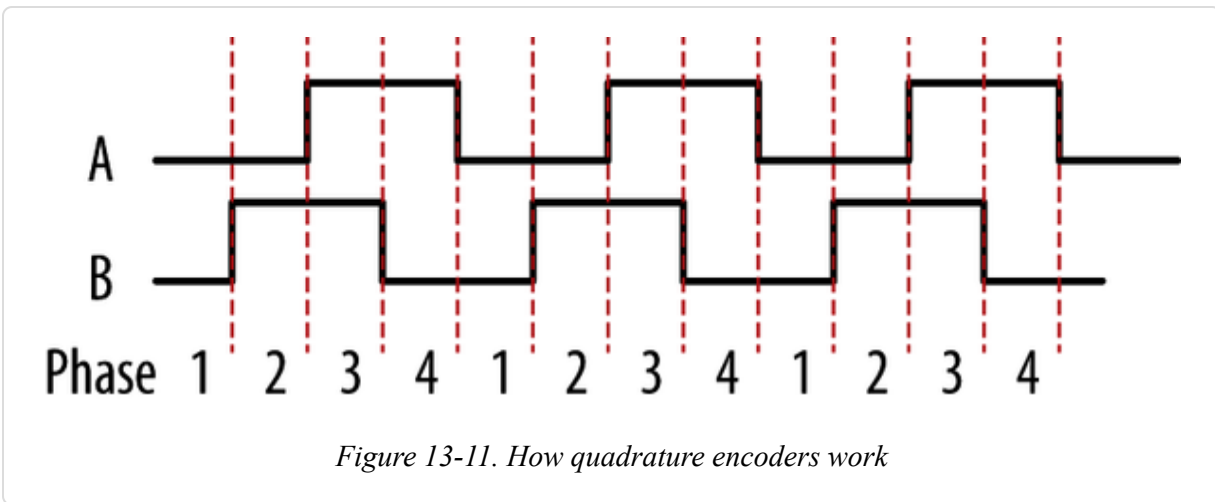
```

8  
9  
10  
9  
8  
7  
6  
5  
4

## Discussion

Rotary encoders have replaced variable resistors in many applications, as they are generally lower cost and do not suffer from track corrosion or wear.

**Figure 13-11** shows the sequence of pulses that you will get from the two contacts, A and B. You can see that the pattern repeats itself after four steps (hence the name *quadrature* encoder).



When rotating clockwise (left to right in **Figure 13-11**), the sequence will be:

Phase	A	B
1	0	0
2	0	1
3	1	1
4	1	0

When rotating in the opposite direction, the sequence of phases will be reversed:

Phase	A	B
4	1	0
3	1	1
2	0	1
1	0	0

The Python program listed previously implements the algorithm for determining the rotation direction in the function `get_encoder_turn`. The function will return 0 if there has been no movement, 1 for a rotation clockwise, or -1 for a rotation counterclockwise. It uses two global variables, `old_a` and `old_b`, to store the previous states of the switches A and B. By comparing them with the newly read values, it can determine (using a bit of clever logic) which direction the encoder is turning.

The sleep period of one millisecond is to ensure that the next new sample doesn't occur too soon after the previous sample; otherwise, the transitions can give false readings.

The test program should work reliably no matter how fast you turn the knob on the rotary encoder; however, try to avoid doing anything time consuming in the loop, or you might find that turn steps are missed.

## See Also

You can also measure the rotated position of a knob by using a variable resistor with the step response method ([Recipe 14.1](#)) or by using an analog-to-digital converter ([Recipe 14.7](#)).

## 13.8 Using a Keypad

## Problem

You want to interface a keypad with your Raspberry Pi.

## Solution

Keypads are arranged in rows and columns, with a push switch on the intersection of each row or column. To find out which key is pressed, you first connect all the row and column connections to Raspberry Pi GPIO pins. So, for a 4×3 keypad, you will need four pins for the rows and three pins for the columns (a total of seven). By scanning each column in turn (setting it to output high) and reading the value of each of the row inputs, you can determine which (if any) key is pressed. You can also tell if more than one key is pressed at the same time; as for a particular column, if more than one key is pressed, the corresponding rows will all be high.

Note that keypads show considerable variation in their pinouts.

To make this recipe, you will need the following:

- Breadboard and jumper wires (see “[Prototyping Equipment and Kits](#)”)
- 4×3 keypad (see “[Miscellaneous](#)”)
- Seven male header pins (see “[Miscellaneous](#)”)

[Figure 13-12](#) shows the wiring diagram for the project using the SparkFun keypad listed in “[Miscellaneous](#)”. The keypad is supplied without header pins, which you must solder onto the keypad.

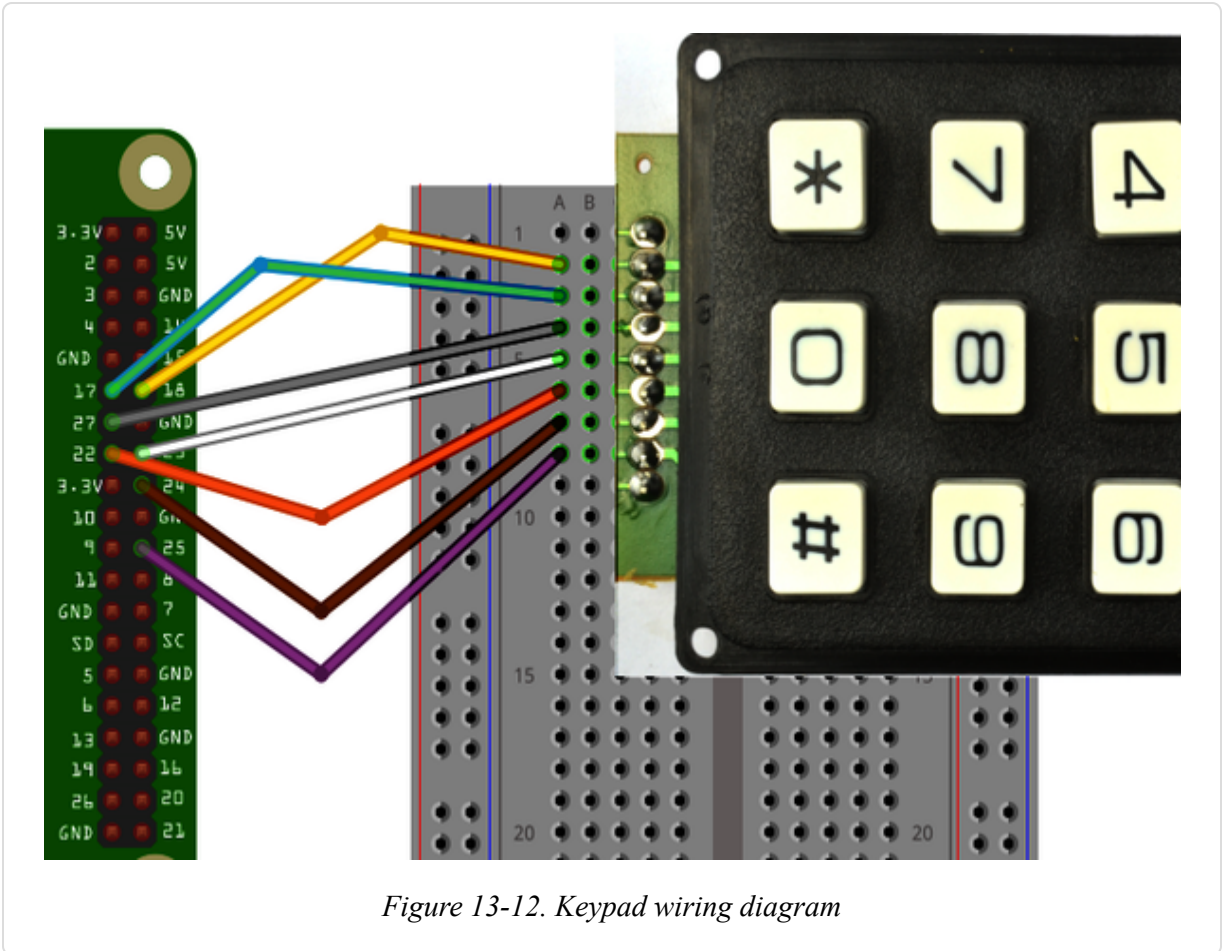


Figure 13-12. Keypad wiring diagram

Open an editor and paste in the following code (*ch\_13\_keypad.py*).

### WARNING

Before you run the program, make sure that the row and column pins are correct for the keypad that you are using. If necessary, change the values in the variables `rows` and `cols`. If you do not do this, it is possible that pressing a key could short one GPIO output to another, where one is high and the other is low. This would likely damage your Raspberry Pi.

```

from gpiozero import Button, DigitalOutputDevice
import time

rows = [Button(17), Button(25), Button(24), Button(23)]
cols = [DigitalOutputDevice(27), DigitalOutputDevice(18),
        DigitalOutputDevice(22)]
keys = [

```

```

    ['1', '2', '3'],
    ['4', '5', '6'],
    ['7', '8', '9'],
    ['*', '0', '#']]

def get_key():
    key = 0
    for col_num, col_pin in enumerate(cols):
        col_pin.off()
        for row_num, row_pin in enumerate(rows):
            if row_pin.is_pressed:
                key = keys[row_num][col_num]
        col_pin.on()
    return key

while True:
    key = get_key()
    if key :
        print(key)
        time.sleep(0.3)

```

When you run the program, you can see each keypress being printed:

```

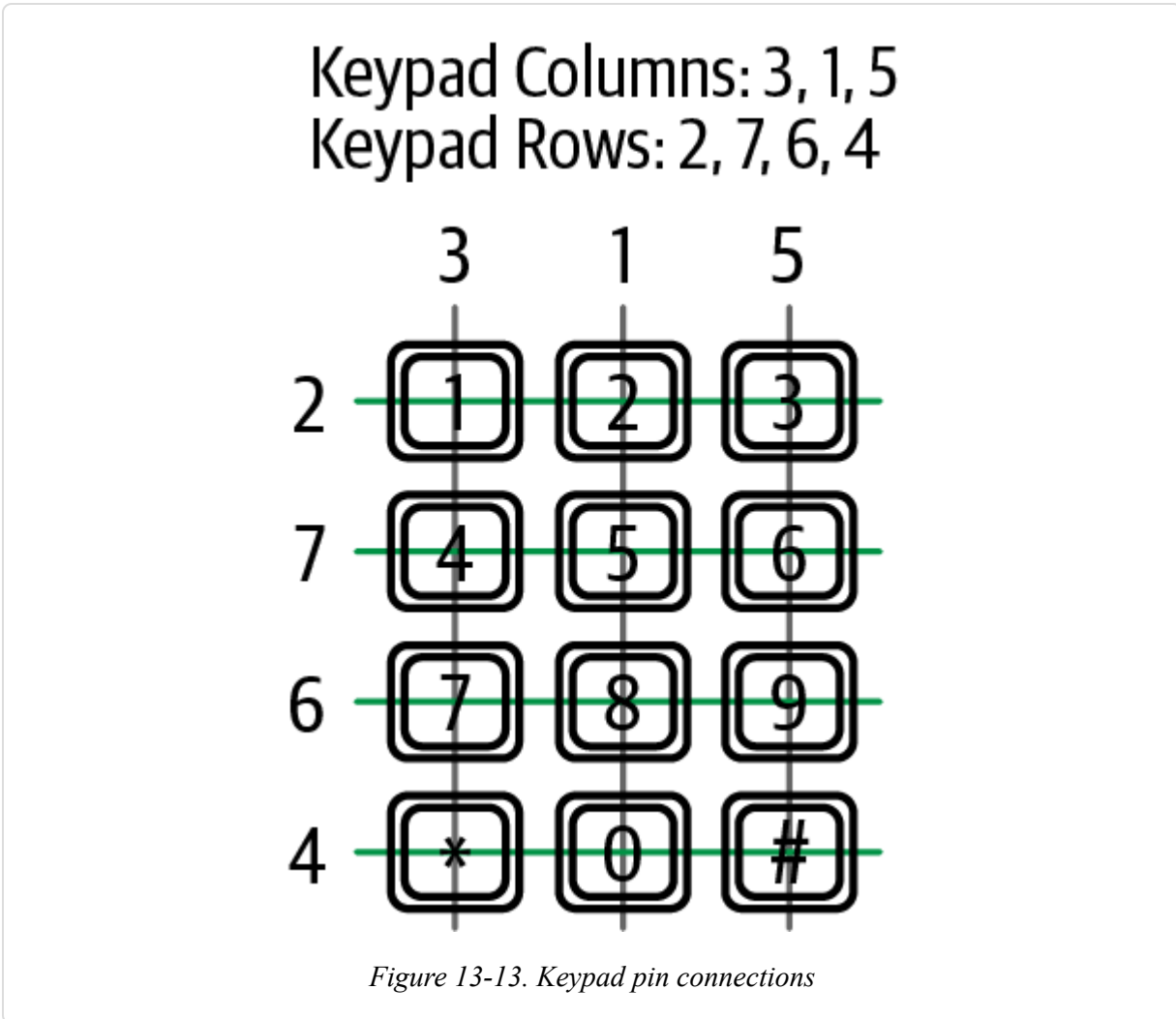
$ sudo python3 ch_13_keypad.py
1
2
3
4
5
6
7
8
9
*
0
#

```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

There is a push switch at the intersection of every row and column, so that when the switch is pressed, the particular row and column will become connected.

The rows and columns defined here are correct for the SparkFun keypad listed in “Miscellaneous” in Appendix A. The first row is connected to GPIO pin 17, the second to pin 25, and so on. The wiring of the row and column to the keypad connector is illustrated in Figure 13-13.



## Discussion

The `keys` variable contains a map of the key name for each row and column position. You can customize this for your keypad.

All the real action takes place in the `get_key` function. This enables each column in turn by setting it to low. An inner loop then tests each of the rows in turn. If one of the rows is low, the key name corresponding to that row



and column is looked up in the `keys` array. If no keypress is detected, the default value of `key` (0) is returned.

The main `while` loop just gets the key value and prints it. The `sleep` command slows down the output.

## See Also

An alternative to adding a keypad is simply to use a USB keyboard; that way you can just catch keystrokes, as described in [Recipe 13.11](#).

## 13.9 Detecting Movement

### Problem

You want to trigger some action in Python when movement is detected.

### Solution

Use a passive infrared (PIR) motion detector module.

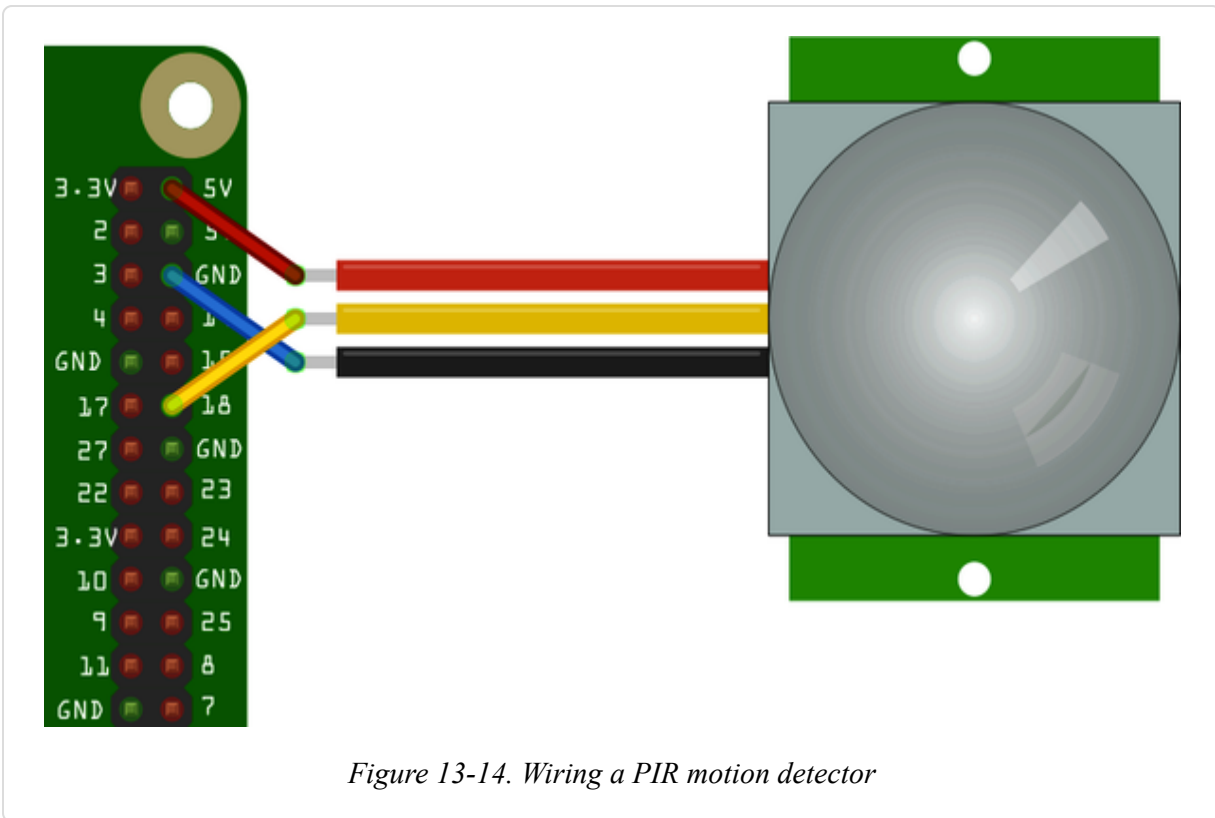
To make this recipe, you will need the following:

- Female-to-female jumper wires (see [“Prototyping Equipment and Kits”](#))
- PIR motion detector module (see [“Modules”](#))

[Figure 13-14](#) shows how the sensor module is wired. This module expects a power supply of 5V and has an output of 3.3V, making it ideal for use with a Raspberry Pi.

### WARNING

Make sure that the PIR module you use has a 3.3V output. If it has a 5V output, you will need to use a pair of resistors to reduce it to 3.3V (see [Recipe 14.8](#)).



Open an editor and paste in the following code (*ch\_13\_pir.py*):

```
from gpiozero import MotionSensor

pir = MotionSensor(18)

while True:
    pir.wait_for_motion()
    print("Motion detected!")
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

The program simply prints out the state of the GPIO input 18:

```
$ python3 ch_13_pir.py
Motion Detected
Motion Detected
```

## Discussion

`gpiozero` provides a class for the PIR sensor, `MotionSensor`, so we might as well use it. However, this class doesn't really do anything except monitor the pin in question as a digital input.

When triggered, the output of the PIR sensor will stay high for a little while. You can adjust this using one of the *trimpots* (variable resistors) on its circuit board. The second trimpot (if present) will set the threshold of light level that will disable the sensor. This is useful when the sensor is being used to control a light—turning the light on when detecting movement, but only when it's dark.

## See Also

More information is available in the [full documentation for `MotionSensor`](#).

You could combine this recipe with [Recipe 7.16](#) to send an email when an intruder is detected, or you could integrate it with If This Then That (IFTTT) to provide a bevy of possible ways of being notified (see [Recipe 17.4](#)).

To detect movement using computer vision and a webcam, see [Recipe 8.6](#).

## 13.10 Adding GPS to the Raspberry Pi

### Problem

You want to connect a serial GPS module to a mobile Raspberry Pi and access the data using Python.

### Solution

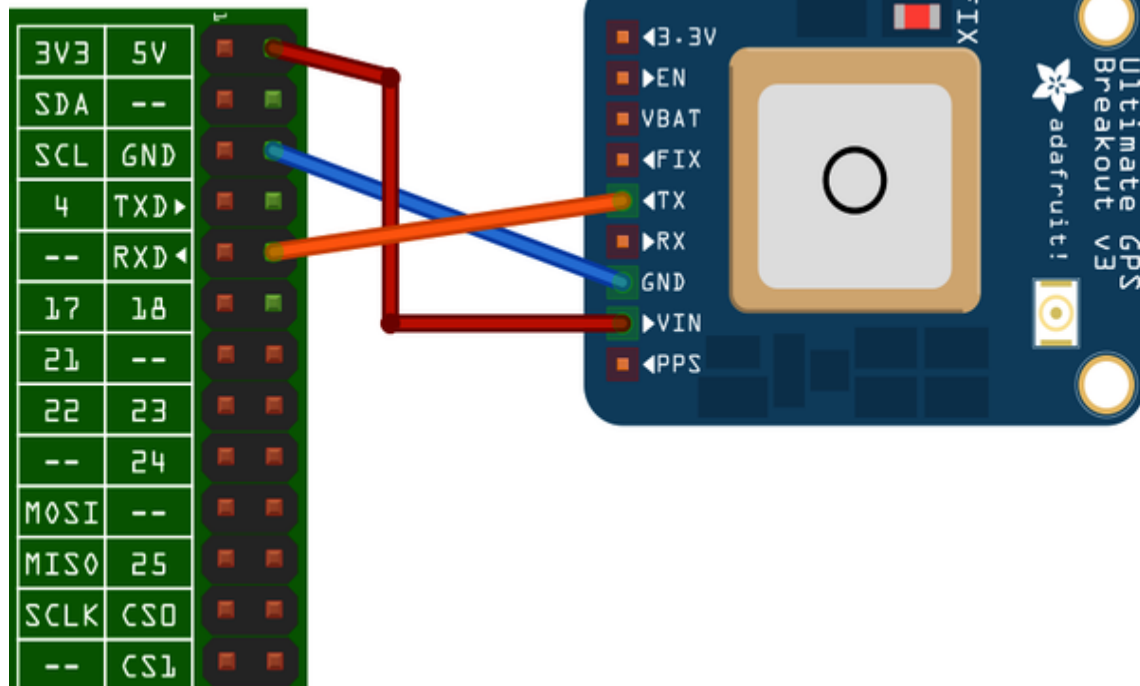
A serial GPS module, with 3.3V output, can be connected directly to Raspberry Pi's RXD connection. This means that for it to work, you must

follow [Recipe 2.6](#). Note that, although you need to enable the serial port hardware, you should *not* enable the serial console option.

[Figure 13-15](#) shows how the module is wired. The RXD of the Raspberry Pi connects to TX of the GPS module. The only other connections are for GND and power, so we can use three female-to-female headers.

## Check Your Voltage

Some GPS modules need a supply voltage of 3.3V rather than the 5V shown as follows. So check before connecting.



*Figure 13-15. Wiring a GPS to a Raspberry Pi*

You can see the raw GPS data using Minicom (see [Recipe 10.8](#)). Use the following `minicom` command to see the messages appear once per second on the `/dev/serial0` device:

```

$ minicom -b 9600 -o -D /dev/serial0
Welcome to minicom 2.8
OPTIONS: I18n
Port /dev/serial0, 12:55:56
Press CTRL-A Z for help on special keys
$GPGGA,120346.694,5342.6175,N,00239.7791,W,1,04,5.6,84.4,M,51.3,M
,,0000*7A
$GPGSA,A,3,01,03,31,04,,,,,,,,,6.9,5.6,4.0*3E
$GPGSV,3,1,12,04,63,150,23,03,56,078,41,19,46,263,23,17,39,226,18
*79
$GPGSV,3,2,12,09,37,197,20,06,36,302,20,01,26,134,13,31,20,048,31
*74
$GPGSV,3,3,12,12,08,327,14,21,06,136,,11,04,304,,25,04,001,*72
$GPRMC,120346.694,A,5432.6175,N,00139.7791,W,002.8,084.6,060922,,
,A*7F

```

If you don't see any data, check your connections and make sure that the serial port hardware is enabled. Note that it's fine to test that the GPS module is basically working indoors without a GPS signal. But if you want to get an actual GPS fix, you might need to take your Raspberry Pi outdoors, or at least have the GPS module right next to a window.

As you can see from the preceding sample, GPS messages require some decoding. The good news is that it's fairly easy to spot the actual latitude and longitude figures in the various messages that come from the GPS module. In the preceding example, you can see that the message starting with \$GPRMC has comma-separated fields, the third of which is a number (5432.6175) followed by a letter (N or S). This is the latitude in hundredths of a degree. The next two fields show the longitude in a similar way.

You can find an example of accessing GPS data from a Python program in *ch\_13\_gps\_serial.py*:

```

import serial

ser = serial.Serial('/dev/serial0')

while True:
    line = ser.readline().decode("utf-8")
    message = line.split(',')
    if message[0] == '$GPRMC':
        if message[2] == 'A':

```

```
        lat = message[3] + message[4]
        lon = message[5] + message[6]
        print(F"lat={lat} \t lon={lon}")
    else:
        print("No fix")
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

Run the program and you should see some trace like this. Remember it can take a while for the GPS module to get a fix:

```
$ python3 -i ch_13_gps_serial.py
No fix
No fix
lat=5432.6028N      lon=00139.0515W
lat=5432.6028N      lon=00139.0514W
```

## Discussion

The program reads every incoming message, looking for \$GPRMC messages and then splits the message into its component parts. The second element of the resulting list will be A if the position is known.

If the position is known, the latitude and longitude are.

## See Also

Find out more about the meaning of the various fields in the [GPS messages](#).

For an interesting tutorial on GPS tracking using Python, see <https://oreil.ly/tSVi6>.

# 13.11 Intercepting Keypresses

## Problem

You want to intercept individual keypresses on a USB keyboard or numeric keypad.

## Solution

There are at least two ways to solve this problem. The more straightforward approach is to use the `sys.stdin.read` function. This has the advantage over the other method of not requiring a GUI to be running, so a program using it can be run from an SSH session.

Open an editor, paste in the following code (*ch\_13\_keys\_sys.py*), run the program, and start pressing some keys:

```
import sys, tty, termios

def read_ch():
    fd = sys.stdin.fileno()
    old_settings = termios.tcgetattr(fd)
    try:
        tty.setraw(sys.stdin.fileno())
        ch = sys.stdin.read(1)
    finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
    return ch

while True:
    ch = read_ch()
    if ch == 'x':
        break
    print("key is: " + ch)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)). To stop the program, press the *x* key.

The alternative to this is to use `pygame`, a Python library intended for writing games, which can also be used to detect keypresses. You could then use this to perform some action.

The following example program (*ch\_13\_keys\_pygame.py*) illustrates the use of `pygame` to print out a message each time a key is pressed. However, it works only if the program has access to the windowing system, so you will need to run it using VNC ([Recipe 2.8](#)) or run it directly on the Raspberry Pi:

```

import pygame
import sys
from pygame.locals import *

pygame.init()
screen = pygame.display.set_mode((640, 480))
pygame.mouse.set_visible(0)

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            sys.exit()
        if event.type == KEYDOWN:
            print("Code: " + str(event.key) + " Char: " +
chr(event.key))

```

This opens a blank Pygame window, and keys will be intercepted only if the Pygame window is selected. The program produces output in the Terminal window from which the program is run.

If you press an arrow key or Shift key with the first `stdin` read approach, the program will throw an error because those keys don't have an ASCII value:

```

$ python3 ch_13_keys_pygame.py
Code: 97 Char: a
Code: 98 Char: b
Code: 99 Char: c
Code: 120 Char: x
Code: 13 Char:

```

In this case, Ctrl-C won't stop this program from running. To stop the program, click the *X* on the Pygame window.

## Discussion

When you are using the `pygame` approach, other keys have constant values defined for them, which allows you to use the cursor and other non-ASCII keys (like the up arrow key and Home) on the keyboard. This isn't possible with the other approach.



## See Also

Intercepting keyboard events can also be an alternative to using a matrix keypad ([Recipe 13.8](#)).

# 13.12 Intercepting Mouse Movements

## Problem

You want to detect mouse movements in Python.

## Solution

The solution to this is very similar to that of using `pygame` to intercept keyboard events ([Recipe 13.11](#)).

Open an editor and paste in the following code (*ch\_13\_mouse\_pygame.py*):

```
import pygame
import sys
from pygame.locals import *

pygame.init()
screen = pygame.display.set_mode((640, 480))
pygame.mouse.set_visible(0)

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            sys.exit()
        if event.type == MOUSEMOTION:
            print("Mouse: (%d, %d)" % event.pos)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

When you run the program, the `MOUSEMOTION` event is triggered whenever the mouse moves within the `pygame` window. You can find the coordinates from the `pos` value of the event. The coordinates are absolute coordinates relative to the upper-left corner of the window:

```
Mouse: (262, 285)
Mouse: (262, 283)
Mouse: (262, 281)
Mouse: (262, 280)
Mouse: (262, 278)
Mouse: (262, 274)
Mouse: (262, 270)
Mouse: (260, 261)
Mouse: (258, 252)
Mouse: (256, 241)
Mouse: (254, 232)
```

## Discussion

Other events that you can intercept are `MOUSEBUTTONDOWN` and `MOUSEBUTTONUP`. These can be used to detect when the left mouse button has been pressed or released.

## See Also

You can find the documentation for `mouse` at the [pygame website](#).

# 13.13 Giving the Raspberry Pi a Reset Button

## Problem

You want a reset button to start up your Raspberry Pi 4 or earlier like a typical desktop computer.

## Solution

If you have a Raspberry Pi 400, there is nothing to do; this version of the Raspberry Pi already has a power on switch.

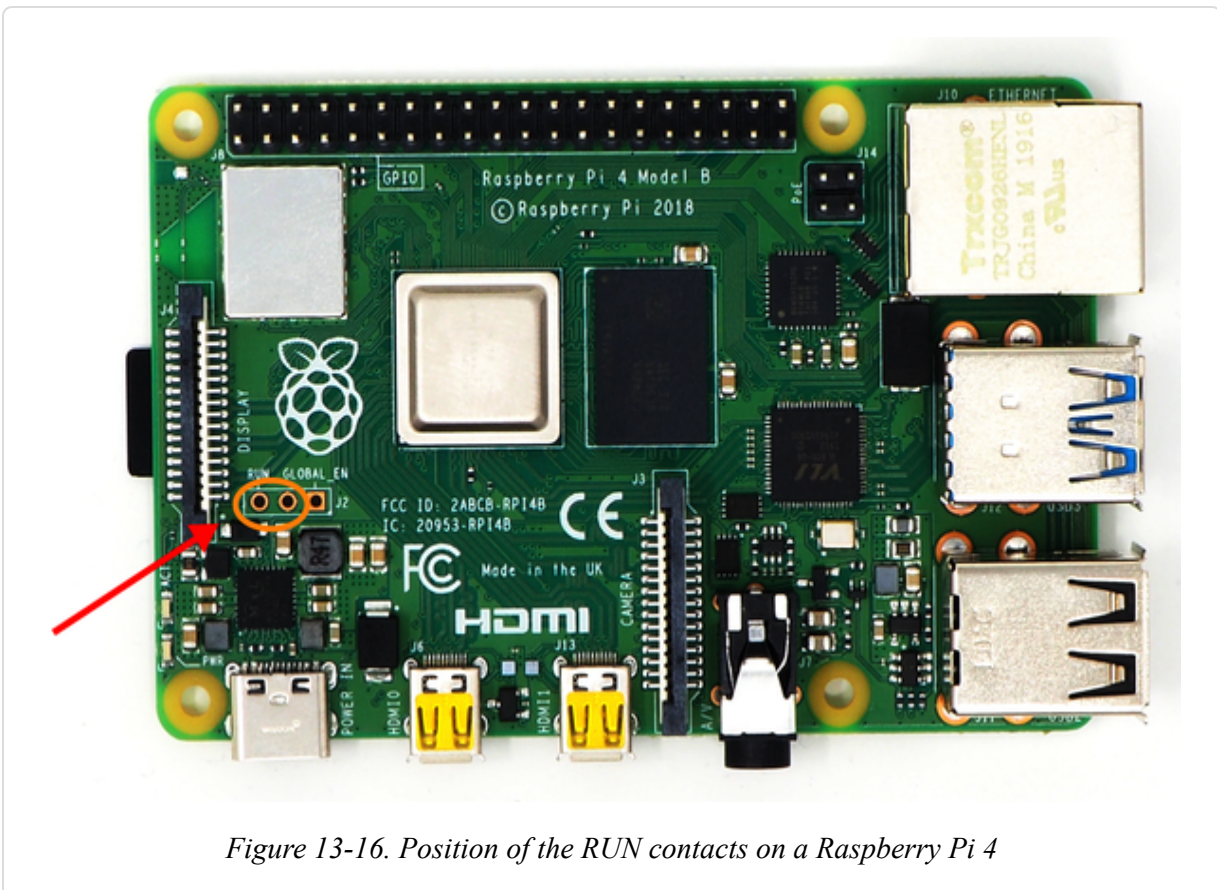
When you are finished using your Raspberry Pi, you should really shut it down; otherwise, it's possible to corrupt the SD card image, which would mean you'd have to reinstall Raspberry Pi OS. Having shut down your Raspberry Pi, you can get it to boot up again by unplugging the USB lead

and then plugging it back in. But a neater solution is to add a reset button to your Raspberry Pi.

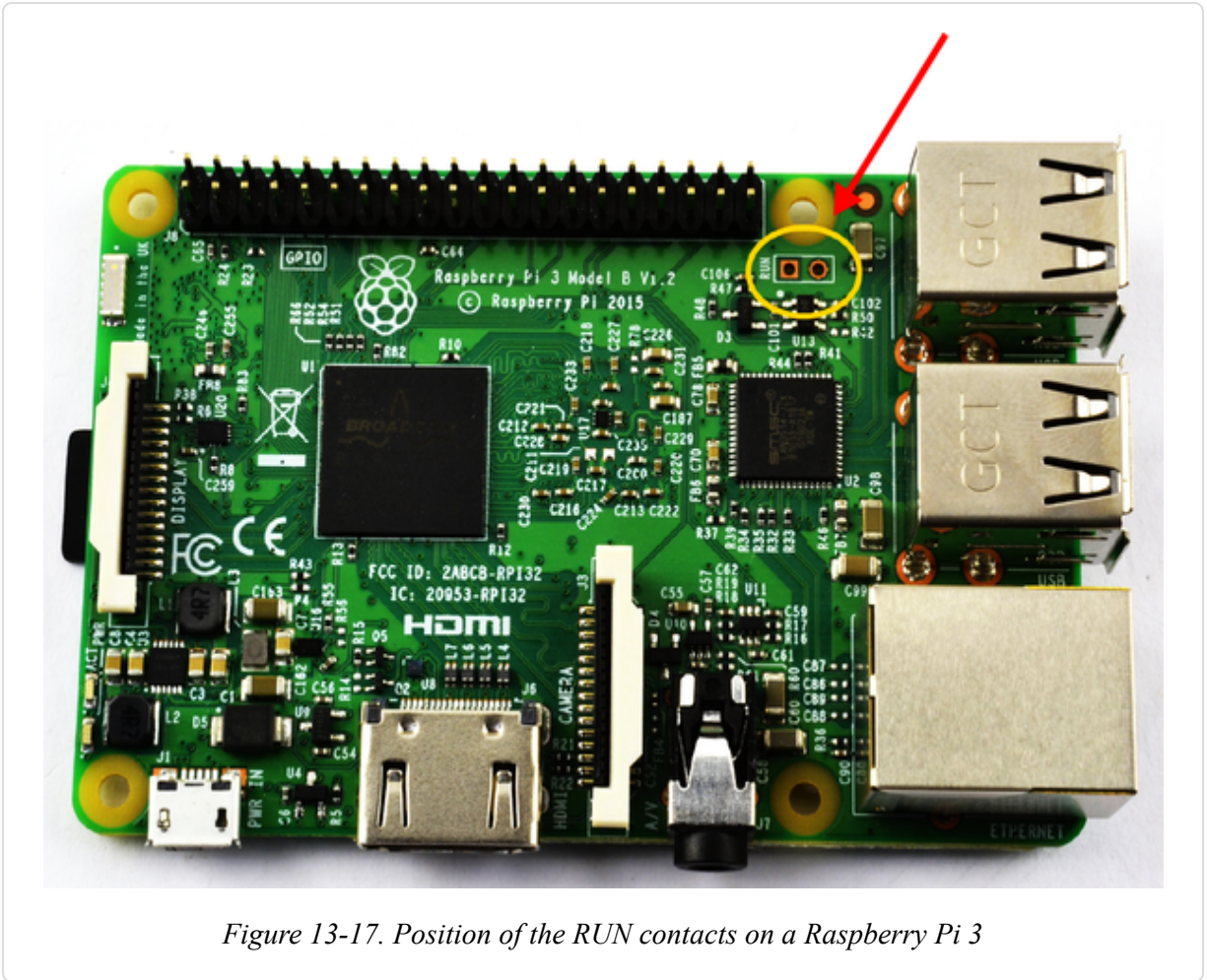
For this recipe, you will need the following:

- Two-way  $\frac{1}{10}$ -inch header pins (see “Miscellaneous”)
- A recycled PC start button or MonkMakes Squid Button (see “Modules”)
- Soldering equipment (see “Prototyping Equipment and Kits”)

Most models of Raspberry Pi have a connector just for this purpose. Its location on the board varies, but it’s always labeled RUN. [Figure 13-16](#) shows its position on a Raspberry Pi 4, and [Figure 13-17](#) on a Raspberry Pi 3.

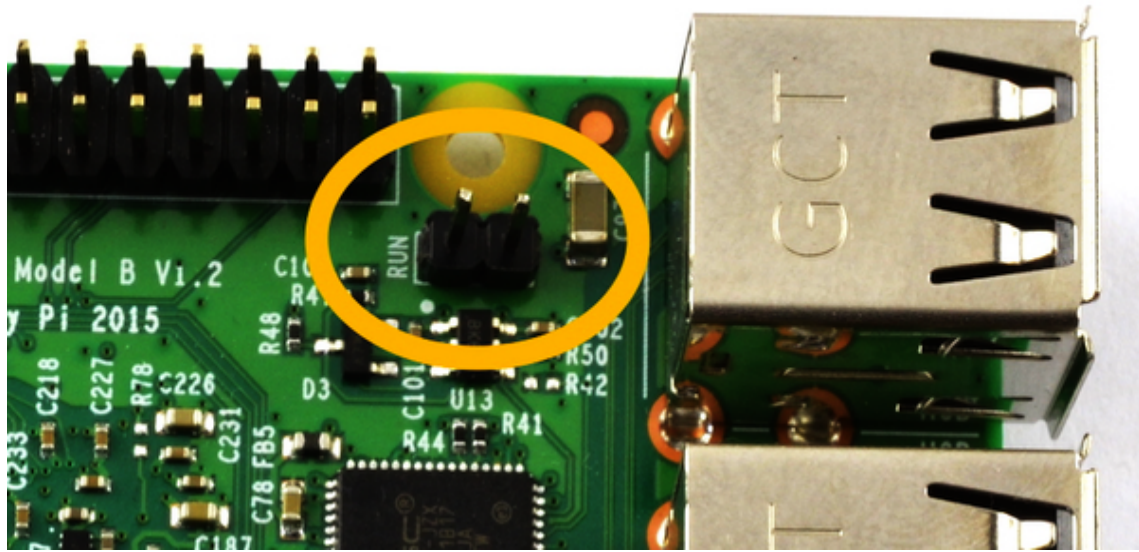


*Figure 13-16. Position of the RUN contacts on a Raspberry Pi 4*



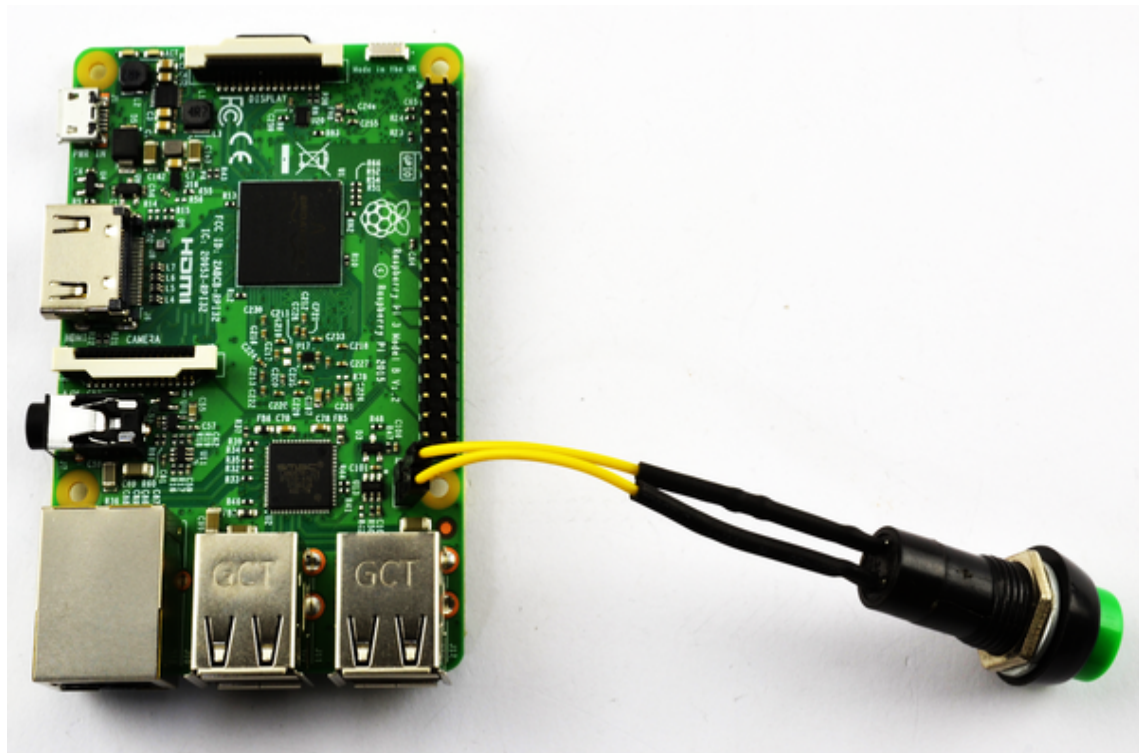
*Figure 13-17. Position of the RUN contacts on a Raspberry Pi 3*

The holes for the contacts are one-tenth of an inch apart and are designed to be fitted with standard header pins. Push through the short end of the pins from the top of the board and solder the underside. After you've soldered them in place, the Raspberry Pi with RUN header pins should look like [Figure 13-18](#).



*Figure 13-18. Pins attached to a Raspberry Pi*

Now that the pins are attached, the button connectors can just be pushed over the header pins, as shown in [Figure 13-19](#).



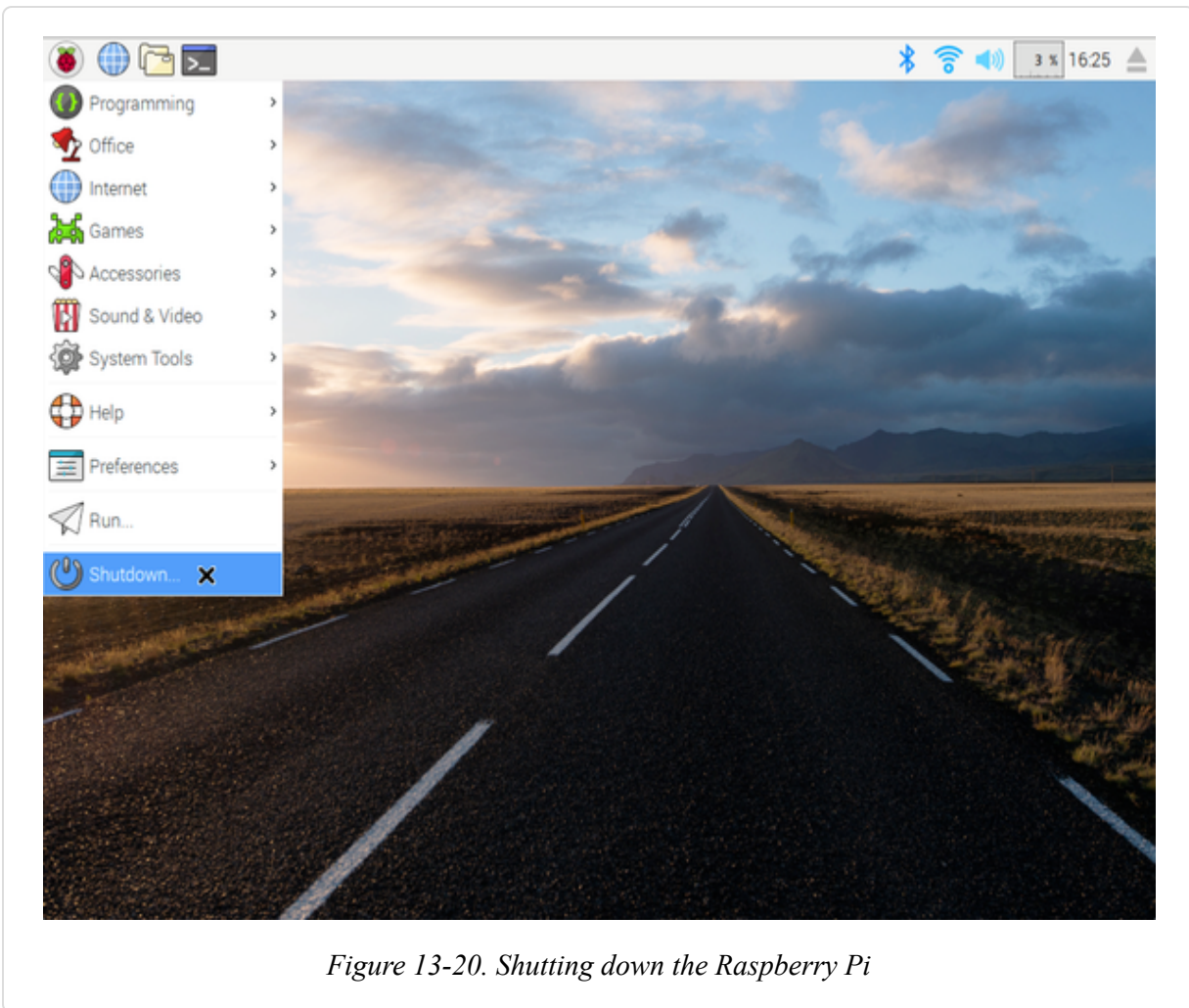
*Figure 13-19. A Raspberry Pi 3 complete with reset button*

## Discussion

To test out your modification, power up your Raspberry Pi and then shut it down by selecting Shutdown from the Raspberry Menu ([Figure 13-20](#)).

After a while the screen will close down and the Pi will go into a *halt* mode, where it uses minimal power and is basically on standby.

Now, to start your Pi, all you need to do is press the button, and it will boot up!



*Figure 13-20. Shutting down the Raspberry Pi*

## See Also

For more information on shutting down and starting your Raspberry Pi, see [Recipe 1.15](#).

# Chapter 14. Sensors

---

## 14.0 Introduction

In this chapter, we look at recipes for using sensors of various types that will enable the Raspberry Pi to measure temperature, light, and more.

Unlike boards such as the Arduino and the Raspberry Pi Pico, a regular Raspberry Pi lacks analog inputs. This means that for many sensors, it is necessary to use additional analog-to-digital converter (ADC) hardware. Fortunately, this is relatively easy to do. It is also possible to use resistive sensors with a capacitor and a couple of resistors.

Many of the recipes will require the use of a solderless breadboard and male-to-female jumper wires (see [Recipe 10.9](#)).

## 14.1 Using Resistive Sensors

### Problem

You want to connect a variable resistor to a Raspberry Pi and measure its resistance to determine the position of the variable resistor's knob in your Python program.

### Solution

You can measure resistance on a Raspberry Pi using nothing more than a capacitor, a couple of resistors, and two general-purpose input/output (GPIO) pins. In this case, you will be able to estimate the position of the knob on a small variable resistor (trimpot) by measuring its resistance from its slider contact to one end of the pot.

To make this recipe, you'll need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))

- 10kΩ trimpot (see “Resistors and Capacitors”)
- Two 1kΩ resistors (see “Resistors and Capacitors”)
- 330 nF capacitor (see “Resistors and Capacitors”)

Figure 14-1 shows the arrangement of components on the breadboard.

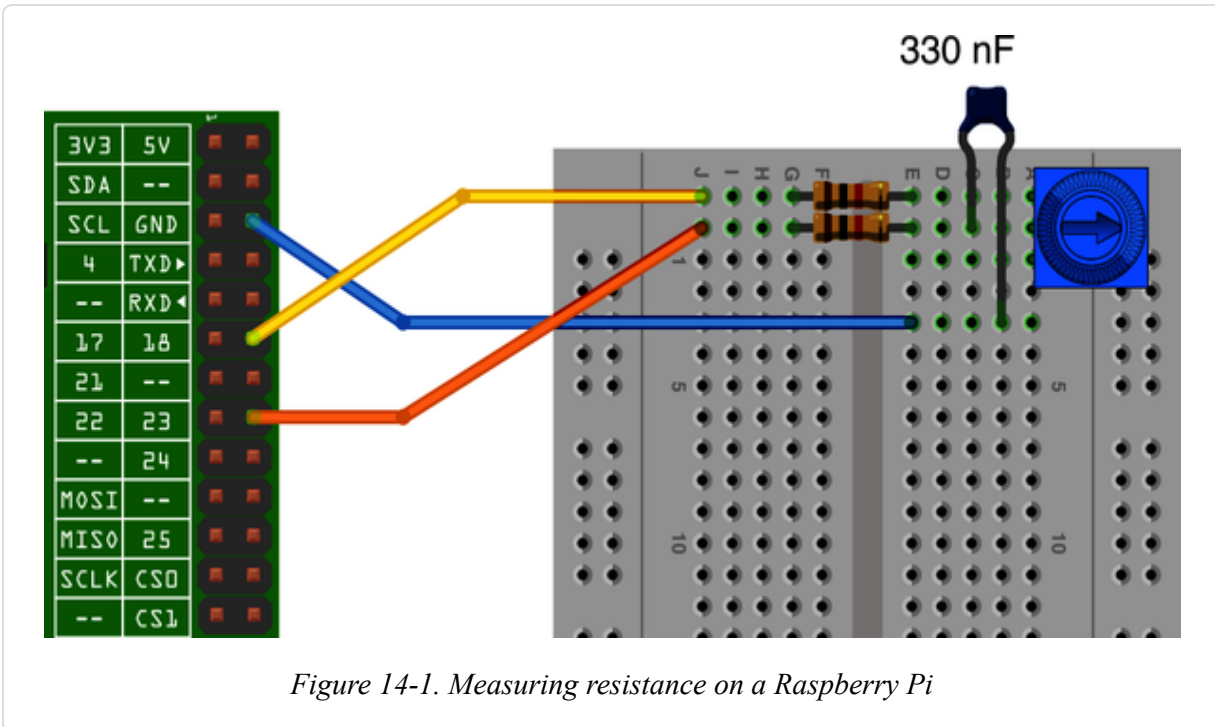


Figure 14-1. Measuring resistance on a Raspberry Pi

This recipe makes use of a Python library that the author developed to make this approach to using analog sensors easier. To install it, run the following commands:

```
$ git clone https://github.com/simonmonk/pi_analog.git
$ cd pi_analog
$ sudo python3 setup.py install
```

Open an editor and paste in the following code (*ch\_14\_resistance\_meter.py*):

```
from PiAnalog import *
import time
```



```
p = PiAnalog()

while True:
    print(p.read_resistance())
    time.sleep(1)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

When you run the program, you should see some output like this:

```
$ python3 ch_14_resistance_meter.py
5588.419502667787
5670.842306126099
8581.313103654076
10167.614271851775
8724.539614581638
4179.124682880563
267.41950235897957
```

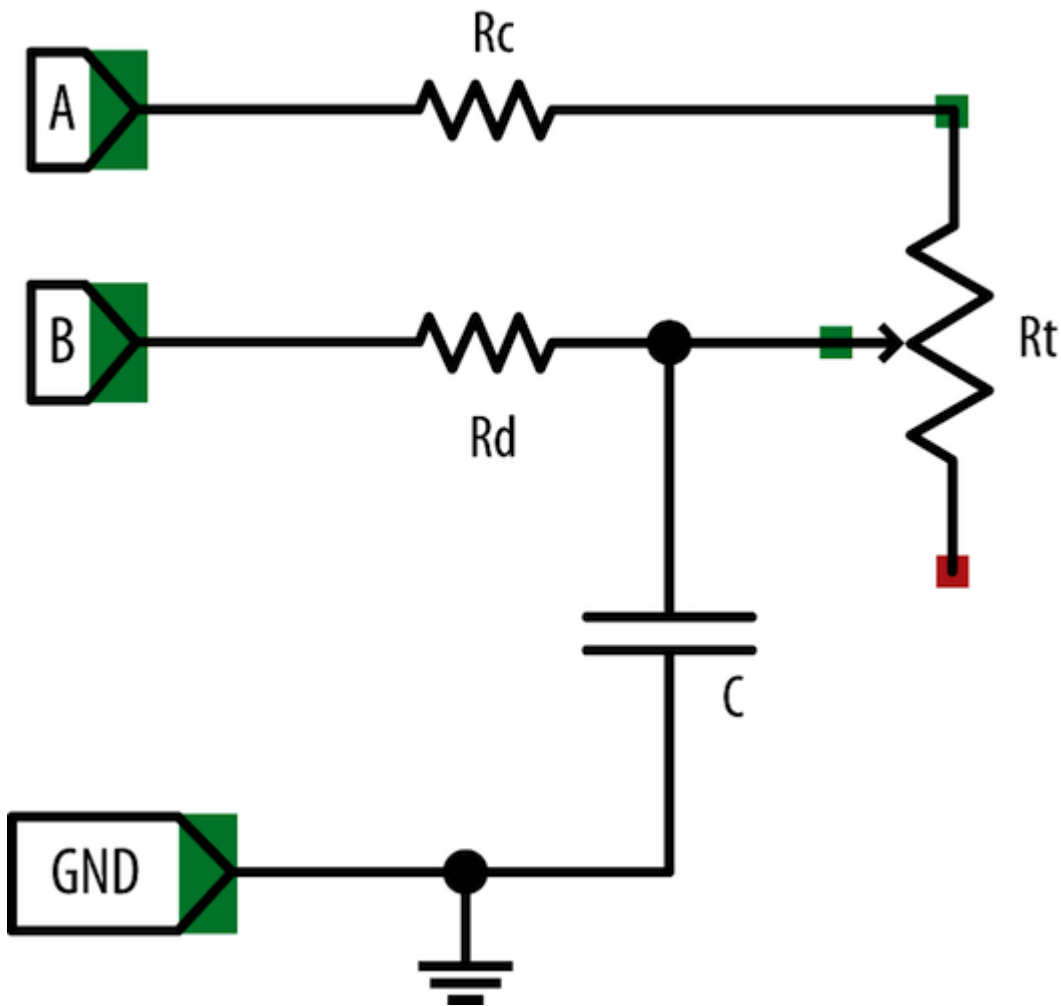
The reading will vary as you rotate the knob of the trimpot. Ideally, the resistance reading will vary between 0 and 10,000 $\Omega$ , but in practice, there will be some error.

## Discussion

To explain how the `PiAnalog` class works, I first need to explain how the *step response* technique for measuring the resistance of the variable resistor works.

[Figure 14-2](#) shows the schematic diagram for the recipe.

This way of doing things is called step response because it works by seeing how the circuit responds from the *step* change when an output is switched from low to high.



*Figure 14-2. Measuring resistance using step response*

You can think of a capacitor as a tank of electricity and, as it fills with charge, the voltage across it increases. You can't measure that voltage directly because the Raspberry Pi doesn't have an analog-to-digital converter (ADC). However, you can time how long it takes for the capacitor to fill with charge to the extent that it rises above the 1.65V or so that constitute a high digital input. The speed at which the capacitor fills with charge depends on the value of the variable resistor ( $R_t$ ). The lower the resistance, the faster the capacitor fills with charge and the voltage rises.

To be able to get a good reading, you must also be able to empty the capacitor each time before you take a reading. In [Figure 14-2](#), connection A is used to charge the capacitor through  $R_c$  and  $R_t$ , and connection B is used

to discharge (empty) the capacitor through  $R_d$ . The resistors  $R_c$  and  $R_d$  are used to prevent too much current from flowing through the Raspberry Pi's relatively fragile GPIO pins as the capacitor is charged and discharged.

The steps involved in taking a reading are first to discharge the capacitor through  $R_d$  and then to let it charge through  $R_c$  and  $R_t$ . To discharge it, connection A (GPIO 18) is set to be an input, effectively disconnecting  $R_c$  and  $R_t$  from the circuit. Connection B (GPIO 23) is then set to be a low output. This is held there for 100 milliseconds to empty the capacitor.

Now that the capacitor is empty, you can start to allow charge to flow into it by setting connection B to be an input (effectively disconnecting it) and then enabling connection A to be a high output at 3.3V. Capacitor C will now begin to charge through  $R_c$  and  $R_t$ .

**Figure 14-3** shows how a resistor and capacitor in this kind of arrangement charge and discharge as the voltage is toggled between high and low.

You can see that the voltage at the capacitor increases rapidly at first but then trails off as the capacitor becomes full. Fortunately, you are interested in the portion of the curve up until the capacitor reaches about 1.65V, which is a fairly straight line, meaning that the time taken for the voltage across the capacitor to rise to this point is roughly proportional to the resistance of  $R_t$  and hence the position of the knob.

This approach is not hugely accurate, but it is very low cost and easy to use. The inaccuracy is largely because capacitors of a suitable value are accurate to only 10%.

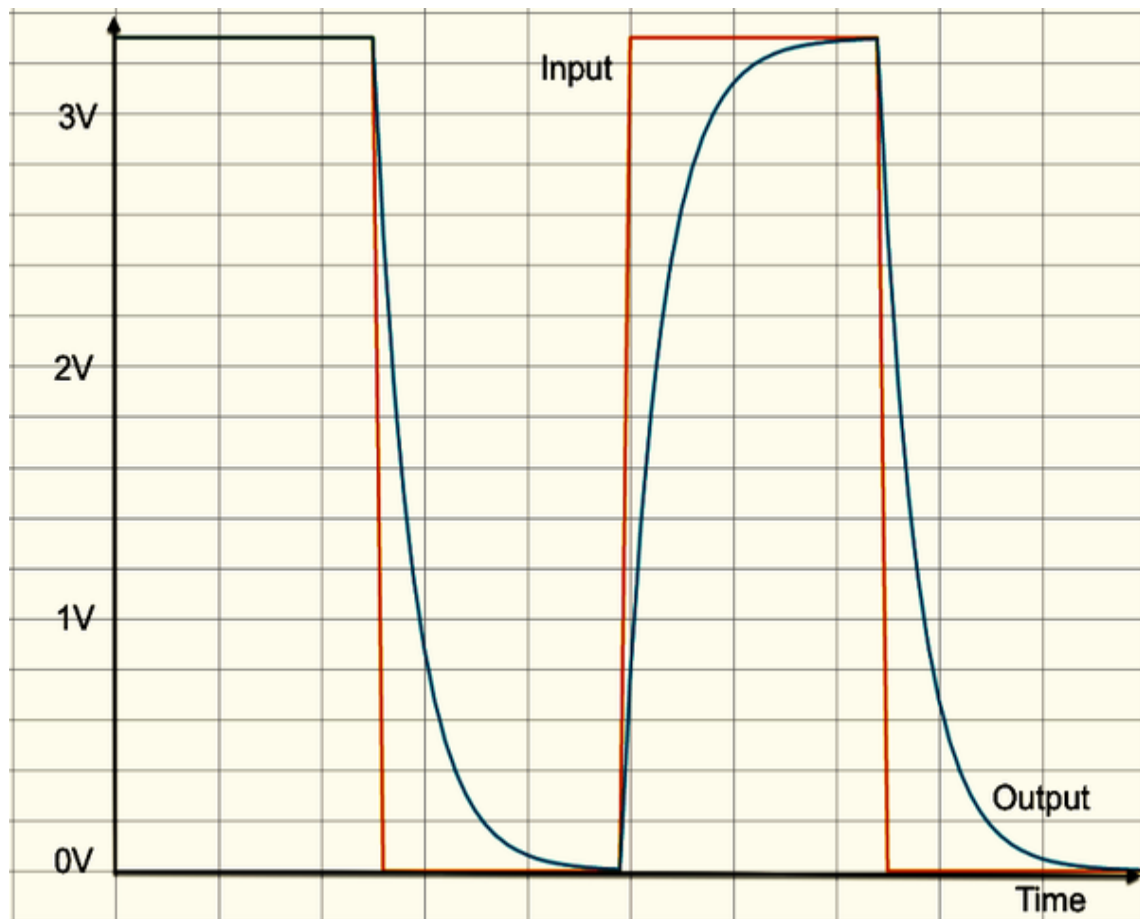


Figure 14-3. Charging and discharging a capacitor

## See Also

Using a step response works well with all kinds of resistive sensors for light ([Recipe 14.2](#)), temperature ([Recipe 14.3](#)), and even gas detection ([Recipe 14.4](#)).

For more accurate measurements of the trimpot position, see [Recipe 14.7](#), in which the pot is used with an ADC.

## 14.2 Measuring Light

### Problem

You want to measure light intensity with a Raspberry Pi and a photoresistor.

## Solution

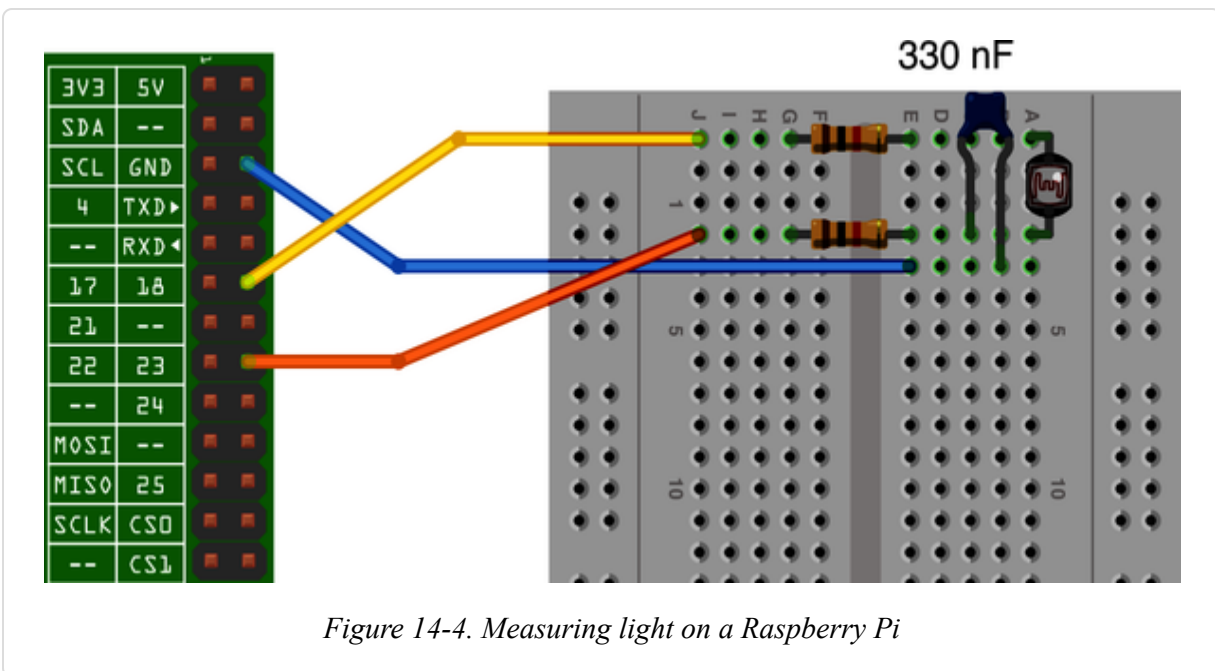
Use the same basic recipe and code as [Recipe 14.1](#), but replace the trimpot with a photoresistor.

To make this recipe, you'll need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- Photoresistor or Phototransistor (see [“Resistors and Capacitors”](#))
- Two 1k $\Omega$  resistors (see [“Resistors and Capacitors”](#))
- 330 nF capacitor (see [“Resistors and Capacitors”](#))

All of these parts are included in the Project Box 1 kit for Raspberry Pi from MonkMakes (see [“Prototyping Equipment and Kits”](#)).

[Figure 14-4](#) shows the arrangement of components on the breadboard. If you are using a phototransistor rather than a photoresistor, then the longer lead is the negative lead and should go to row 4 of the breadboard.



Using the same program as [Recipe 14.1](#) (`ch_14_resistance_meter.py`), you will see the output vary as you move your hand over the

photoresistor/phototransistor to cut out some of the light.

Note that this program needs the `PiAnalog` library to be installed (see [Recipe 14.1](#)).

This solution provides relatively reliable readings of light levels. As an adaptation of the general solution for using resistive sensors ([Recipe 14.1](#)), it also copes with measuring a resistance of  $0\Omega$  without any risk of damaging the GPIO pins of the Raspberry Pi.

## Discussion

A photoresistor is a resistor whose resistance varies depending on the amount of light coming through its transparent window. The brighter the light, the lower the resistance. Typically, the resistance varies from about  $1k\Omega$  in bright light up to perhaps  $100k\Omega$  in complete darkness.

A phototransistor works in a similar way, conducting more as more light falls on it. But unlike the photoresistor, the phototransistor has a positive and negative and won't work if connected the wrong way around.

## See Also

You could also use an ADC with the photoresistor or phototransistor ([Recipe 14.7](#)).

# 14.3 Measuring Temperature with a Thermistor

## Problem

You want to measure temperature using a thermistor.

## Solution

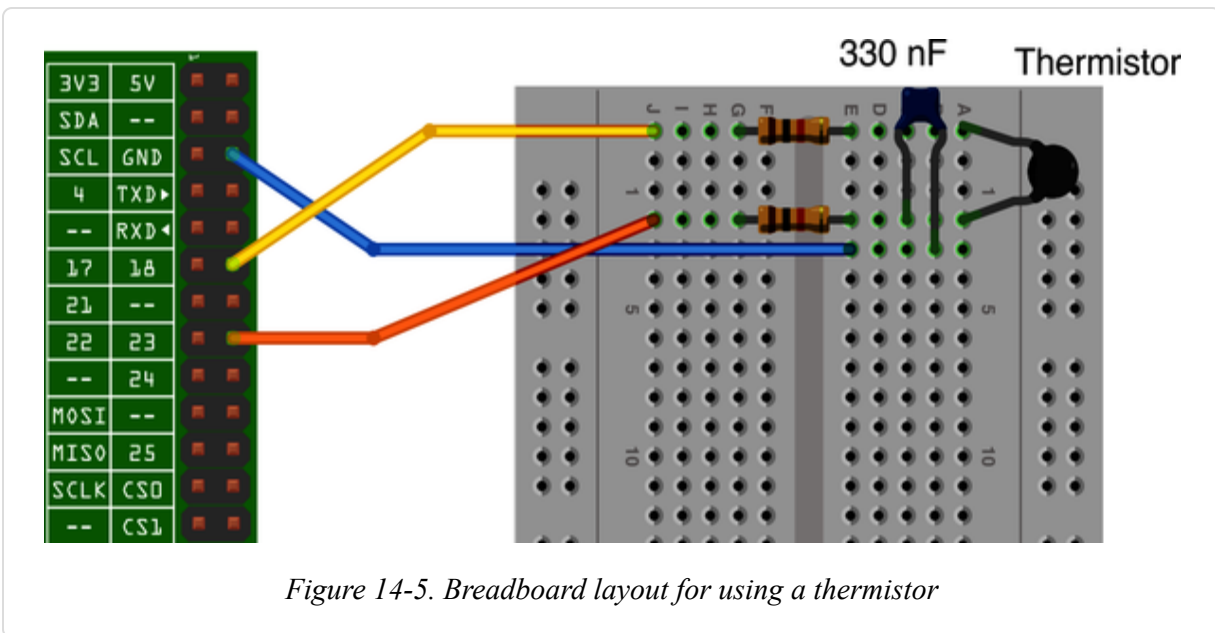
A thermistor is a resistor whose resistance varies with temperature. Use the step response method ([Recipe 14.1](#)) to measure the resistance of the thermistor and then calculate the temperature.

To make this recipe, you will need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- 1k thermistor (see [“Resistors and Capacitors”](#))
- Two 1k $\Omega$  resistors (see [“Resistors and Capacitors”](#))
- 330 nF capacitor (see [“Resistors and Capacitors”](#))

All of these parts are included in the Project Box 1 kit for Raspberry Pi from MonkMakes (see [“Prototyping Equipment and Kits”](#)). When you get your thermistor, make sure that you know its values of Beta and R0 (resistance at 25°C) and that it is a negative temperature coefficient (NTC) device.

[Figure 14-5](#) shows the breadboard layout for this recipe.



*Figure 14-5. Breadboard layout for using a thermistor*

Note that this program needs the `PiAnalog` library to be installed (see [Recipe 14.1](#)).

The following code (`ch_14_thermistor.py`) illustrates the use of the `PiAnalog` module:

```

from PiAnalog import *
import time

p = PiAnalog()

while True:
    print(p.read_temp_c())
    time.sleep(1)

```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

When you run the program, you will see a series of temperature measurements in degrees Celsius. To convert to degrees Fahrenheit, change the code `p.read_temp_c()` to `p.read_temp_f()`:

```

$ python3 ch_14_thermistor.py
18.735789861164392
19.32060395712483
20.2694035007122
21.03181169007422
21.26640936199749

```

## Discussion

Calculating the temperature from the resistance of the thermistor requires some fairly hairy math using logarithms called the Steinhart-Hart equation. This equation needs to know two things about the thermistor: its resistance at 25°C (called  $R_0$ ) and a constant for the thermistor called Beta, or sometimes just B. If you use a different thermistor, you will need to plug these values into the code when you call `read_temp_c`. For example:

```

read_temp_c(self, B=3800.0, R0=1000.0)

```

Note that a capacitor typically has an accuracy of only 10%, and thermistors are similarly inaccurate in their value of  $R_0$ , so do not expect massively accurate results.



## See Also

To measure temperature using a TMP36, see [Recipe 14.10](#).

To measure temperature using a digital temperature sensor (DS18B20), see [Recipe 14.13](#).

To measure temperature using a Sense HAT, see [Recipe 14.12](#).

## 14.4 Detecting Methane

### Problem

You want to measure gas levels using a methane sensor.

### Solution

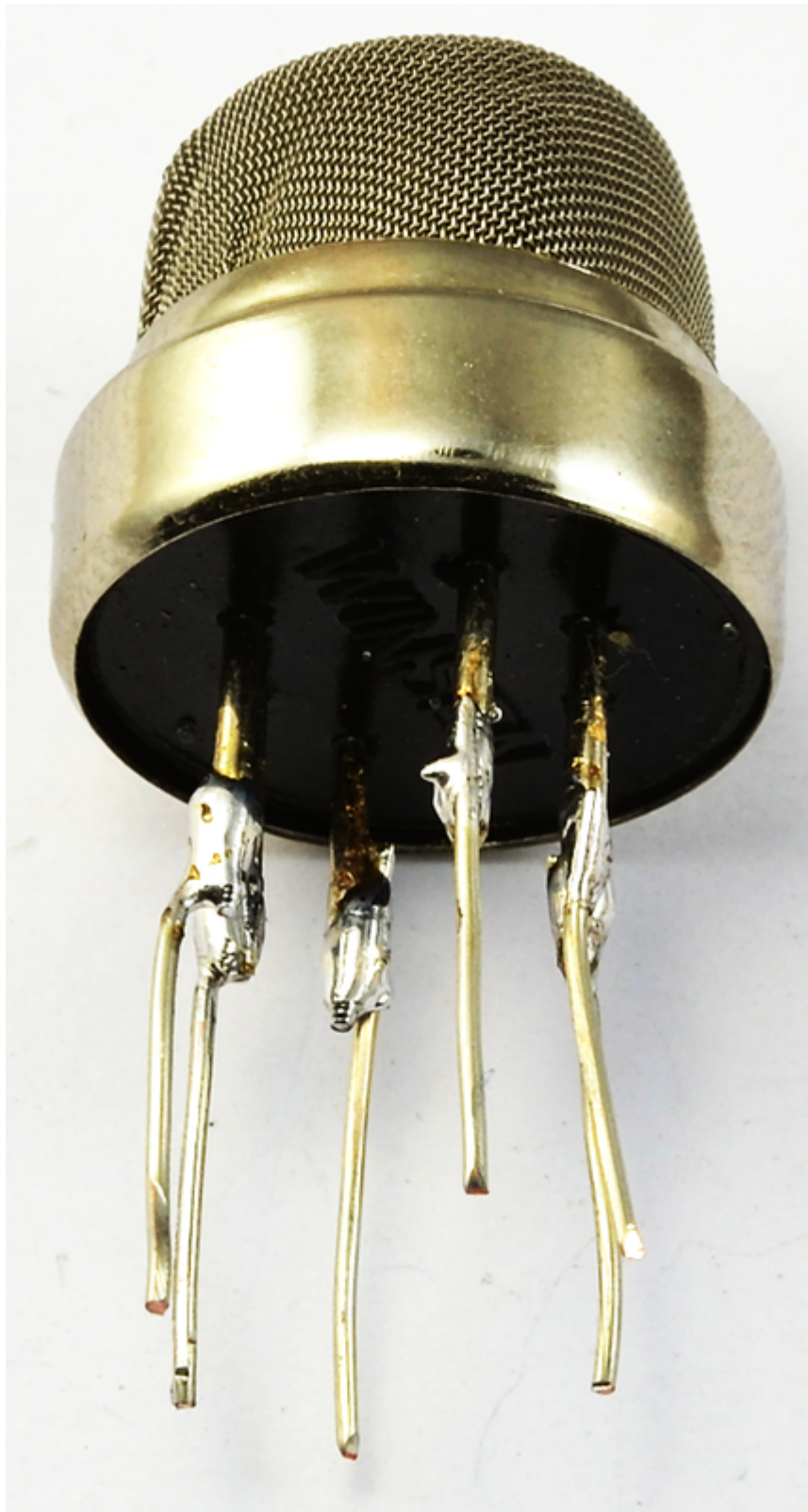
Low-cost resistive gas sensors are available that can easily be wired to a Raspberry Pi to detect gases such as methane. You can use the step response method that you first used in [Recipe 14.1](#).

To make this recipe, you'll need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- Methane sensor (see [“Modules”](#))
- Two 1k $\Omega$  resistors (see [“Resistors and Capacitors”](#))
- 330 nF capacitor (see [“Resistors and Capacitors”](#))

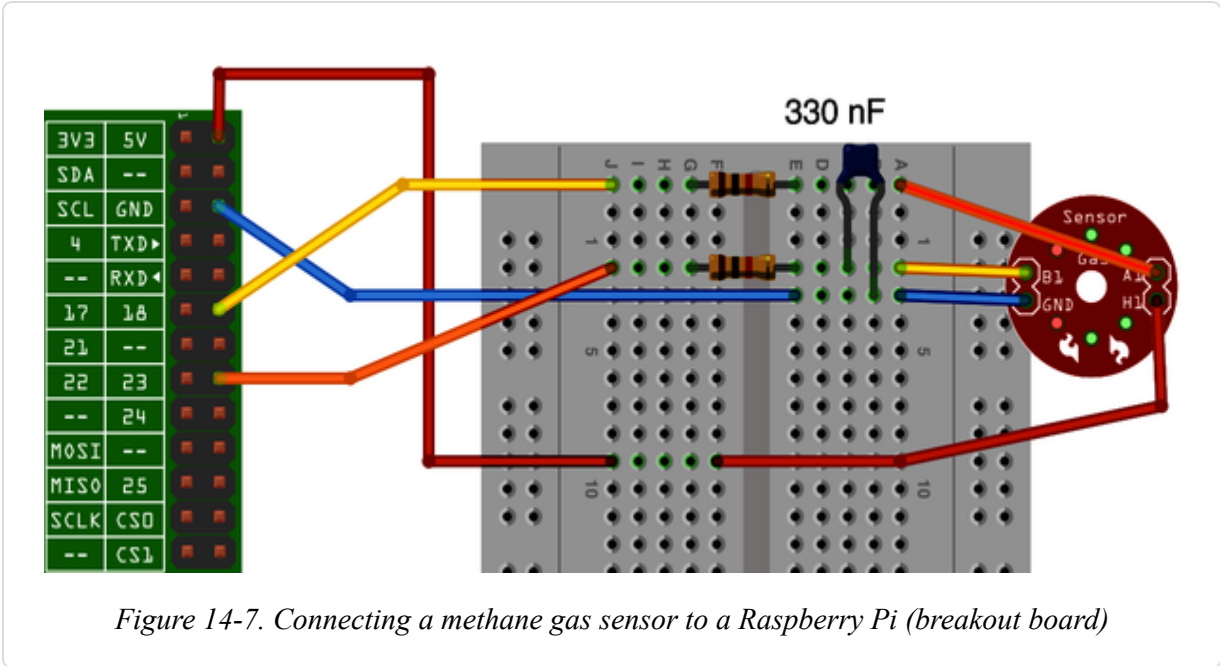
The sensor contains a heating element that requires 5V at up to 150mA. The Raspberry Pi is capable of providing this as long as its power supply can provide the extra 150mA.

The sensor module has rather thick legs—too thick to fit into breadboard holes. One way around this is to solder short lengths of solid core wire to each lead ([Figure 14-6](#)). Another is to buy [SparkFun's gas sensor breakout board](#), which will allow you to plug the sensor in directly.



*Figure 14-6. Soldering leads onto the gas sensor*

Wire the breadboard as shown in [Figure 14-7](#) if you're using the SparkFun breakout board, or as shown in [Figure 14-8](#) if you soldered longer leads to the gas sensor.



Note that the direct connection shown in [Figure 14-8](#) uses the same symbol for the breakout board rather than the sensor on its own, but if you look carefully, the connections are to the six sensor pins, not the four pins of the breakout.

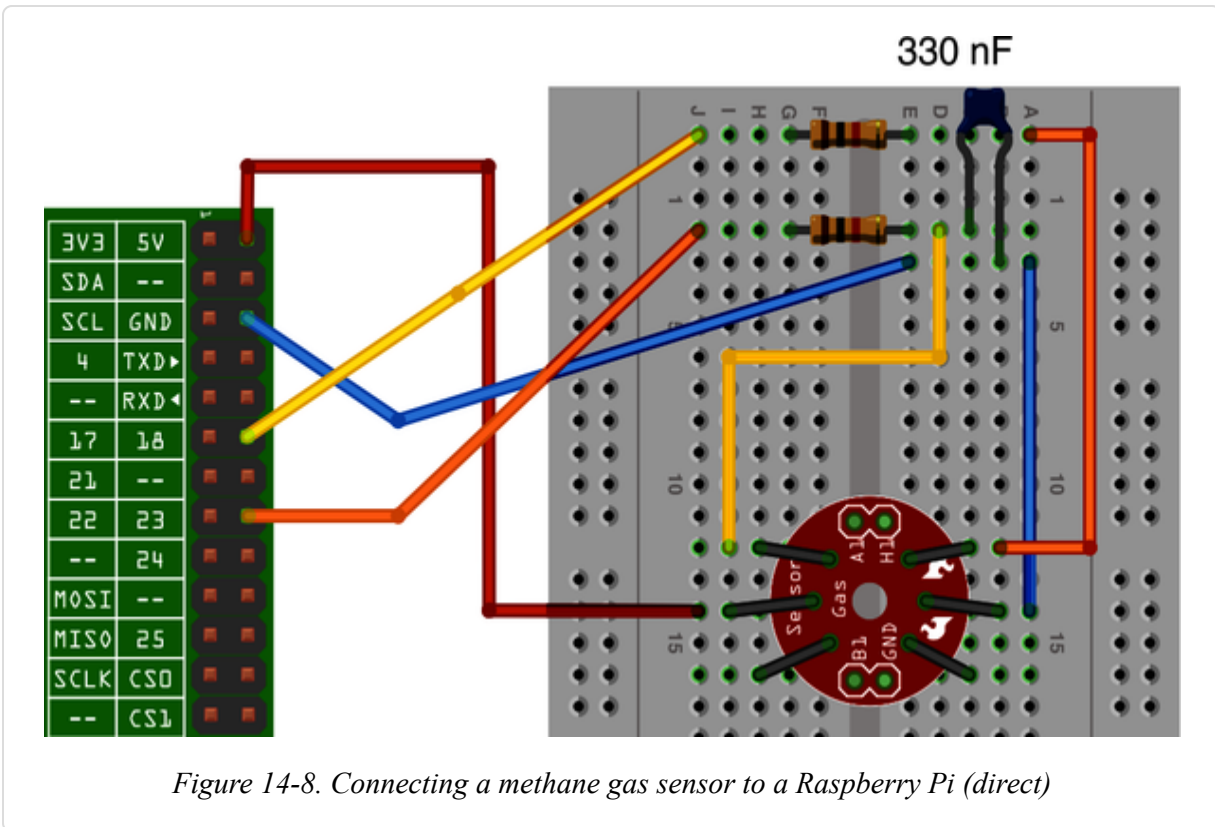


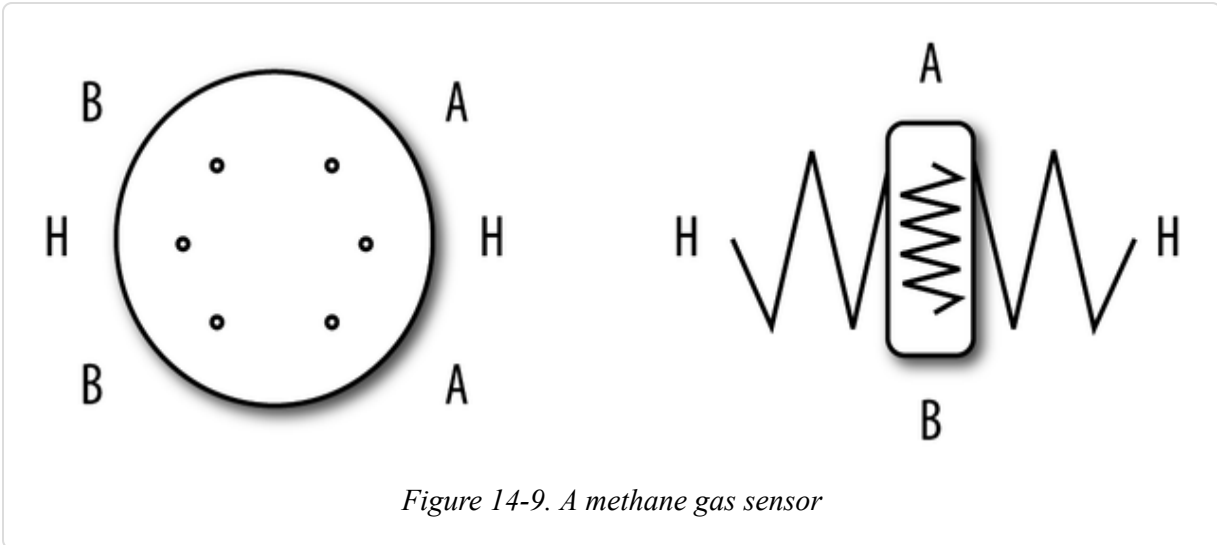
Figure 14-8. Connecting a methane gas sensor to a Raspberry Pi (direct)

You can use the exact same program as [Recipe 14.1](#), and you can test the methane sensor by breathing on it. You should see the readings from the sensor drop when you breathe on it.

## Discussion

The obvious use of a methane gas sensor is for novelty *fart-detecting* projects. A more serious use would be for detecting leaks of natural gas. You could, for instance, imagine a Raspberry Pi home-watch project that monitored the home with various sensors. It could then send you an email while you were on vacation informing you that your house was about to explode. Or maybe not.

These types of sensors ([Figure 14-9](#)) use a heating element that warms a resistive surface impregnated with a catalyst sensitive to a particular gas. When the gas is present, the resistance of the catalyst layer changes.



Both the heater and the sensing surface are electrically just resistors. So both can be connected either way around.

This particular gas sensor is most sensitive to methane, but it will also detect other gases to a lesser extent. That is why breathing on the sensor alters the reading, as healthy individuals will not normally breathe out methane. The cooling effect of blowing on the element can also have an effect.

## See Also

You can find the [datasheet for this sensor](#). This will give you all sorts of useful information about the sensor's sensitivity to various gases.

A range of these low-cost sensors are available for sensing different gases. See a list of sensors offered by [SparkFun](#).

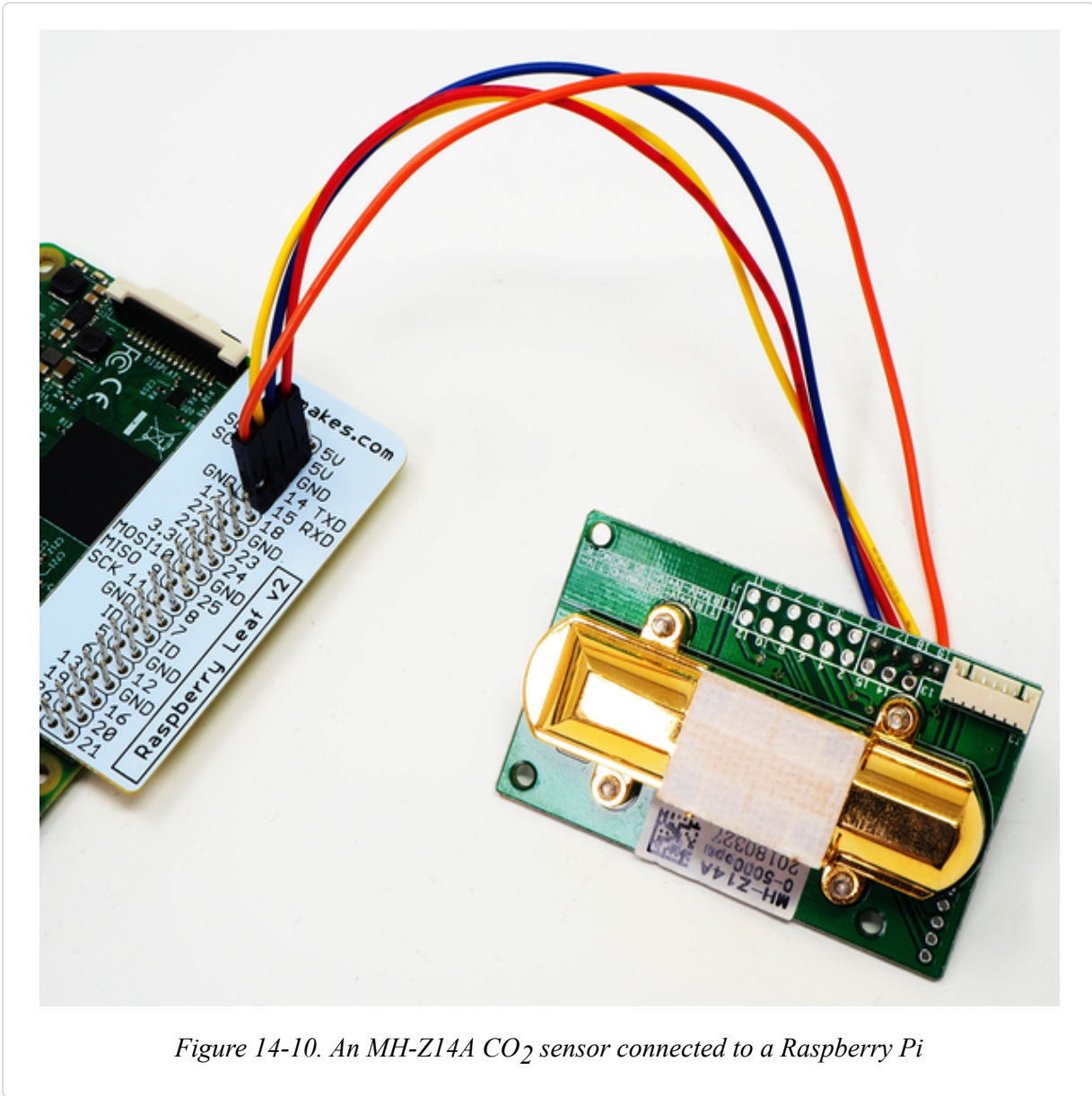
## 14.5 Measuring Air Quality (CO<sub>2</sub>)

### Problem

Carbon dioxide (CO<sub>2</sub>) concentration is an indicator of air quality. You want to use your Raspberry Pi to measure this quality using a special sensor.

## Solution

Use a low-cost (well, relatively low-cost) MH-Z14A CO<sub>2</sub> sensor module, shown connected to a Raspberry Pi in [Figure 14-10](#).



*Figure 14-10. An MH-Z14A CO<sub>2</sub> sensor connected to a Raspberry Pi*

To make this recipe, you'll need the following:

- An MH-Z14A CO<sub>2</sub> sensor module (see [“Modules”](#))
- Female-to-female jumper wires (see [“Prototyping Equipment and Kits”](#))

Make the following connections between the Z14A sensor and your Raspberry Pi:

- Pin 16 of the MH-Z14A to GND on the Raspberry Pi
- Pin 17 of the MH-Z14A to 5V on the Raspberry Pi
- Pin 18 of the MH-Z14A to GPIO 14 (TXD) on the Raspberry Pi
- Pin 19 of the MH-Z14A to GPIO 15 (RXD) on the Raspberry Pi

This sensor uses the serial port. This means that for it to work, you must follow [Recipe 2.6](#). Note that even though you need to enable the serial port hardware, you should *not* enable the serial console option.

The following test program reads the level of CO<sub>2</sub> from the sensor and reports it once per second (`ch_14_co2.py`):

```
import serial, time

request_reading = bytes([0xFF, 0x01, 0x86, 0x00, 0x00, 0x00,
                        0x00, 0x00, 0x79])

def read_co2():
    sensor.write(request_reading)
    time.sleep(0.1)
    raw_data = sensor.read(9)
    high = raw_data[2]
    low = raw_data[3]
    return high * 256 + low;

sensor = serial.Serial('/dev/serial0')
print(sensor.name)
if sensor.is_open:
    print("Open")

while True:
    print("CO2 (ppm):" + str(read_co2()))
    time.sleep(1)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

When the program is run, the CO<sub>2</sub> level should (unless you are in a small, unventilated room) be about 400 ppm. If you breathe on the sensor for a

few seconds, the readings will slowly begin to rise and then fall back to a normal reading over the next few minutes:

```
$ python3 ch_14_co2.py
/dev/ttyS0
Open
CO2 (ppm):489
CO2 (ppm):483
CO2 (ppm):483
CO2 (ppm):481
CO2 (ppm):491
CO2 (ppm):517
CO2 (ppm):619
CO2 (ppm):734
CO2 (ppm):896
CO2 (ppm):1367
```

The sensor uses a request/response communication protocol. So when you want to receive a reading from the sensor, you first need to send the 9-byte message contained in `request_reading`. The sensor will immediately respond with a 9-byte message. We are interested only in bytes 2 and 3, which contain the high and low bytes of the CO<sub>2</sub> reading in ppm.

## Discussion

Normal levels of CO<sub>2</sub> are around 400 to 1,000 ppm. Above that, the air can begin to feel stale, and you might feel drowsy. Studies have shown that high levels of CO<sub>2</sub> because of poor ventilation can result in decreased mental performance. As a result of running the program overnight, I now leave the window and/or door ajar in my bedroom.

## See Also

Learn more about the [Z14A protocol](#).

Find more information about [safe levels of CO<sub>2</sub>](#).

## 14.6 Measuring Soil Moisture

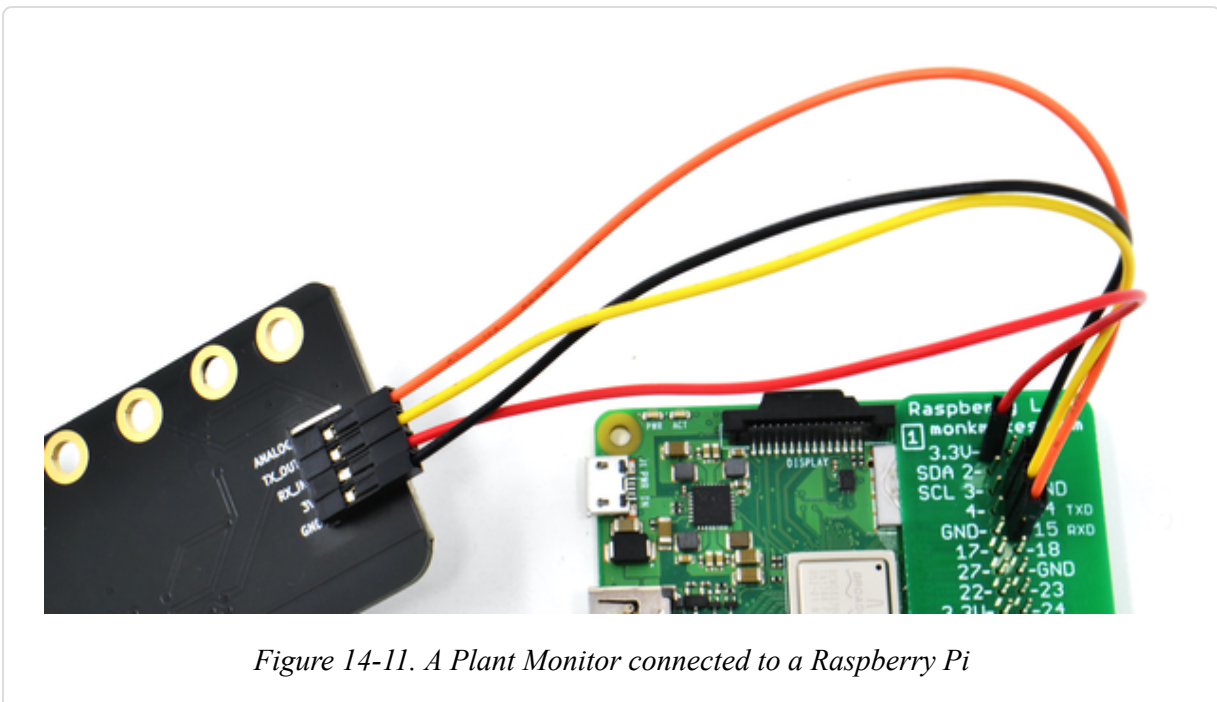


## Problem

You want to measure the moisture of the soil around your plants.

## Solution

Use a MonkMakes Plant Monitor board. Unlike most other soil moisture boards, this board has a serial interface, and so you must first follow [Recipe 2.6](#) to enable the serial interface. Connect up the board as shown in [Figure 14-11](#).



*Figure 14-11. A Plant Monitor connected to a Raspberry Pi*

The connections are:

- GND to GND
- 3.3V on the Raspberry Pi to 3V on the Plant Monitor
- 14 TXD on the Raspberry Pi to RX\_IN on the Plant Monitor
- 15 RXD on the Raspberry Pi to TX\_OUT on the Plant Monitor

Then download the Python software for the board using the command:

```
$ git clone https://github.com/monkmakes/pmon.git
```

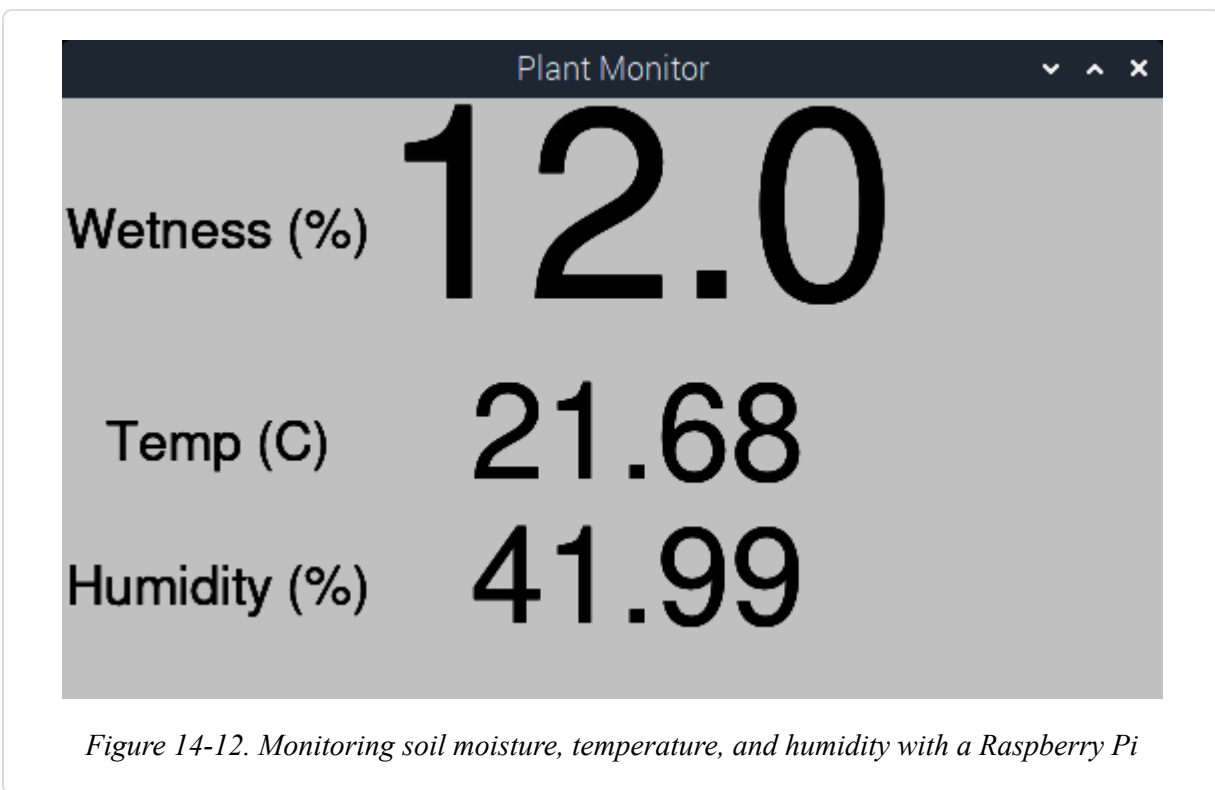
This is another program that uses `guizero`, so if you have not already installed `guizero`, do so, using the command:

```
$ pip3 install guizero
```

Then change to the examples directory and run the example program, using the commands:

```
$ cd pmon/raspberry_pi  
$ python3 01_meter.py
```

This will open the window shown in [Figure 14-12](#).



## Discussion

Most soil moisture sensors need analog inputs and so are not readily connected to a Raspberry Pi. The MonkMakes Plant Monitor uses a serial interface, making it very suitable for connecting to a Raspberry Pi. It

reports the temperature and humidity at the plant and has an RGB LED that indicates the soil moisture.

The code for the preceding example program is:

```
import threading
import time
from guizero import App, Text
from plant_monitor import PlantMonitor

pm = PlantMonitor()

app = App(title="Plant Monitor", width=550, height=300,
layout="grid")

def update_readings(): # update fields with new temp and eCO2
    readings
    while True:
        wetness_field.value = str(pm.get_wetness())
        temp_c_field.value = str(pm.get_temp())
        humidity_field.value = str(pm.get_humidity())
        time.sleep(2)

t1 = threading.Thread(target=update_readings)

# define the user interface
Text(app, text="Wetness (%)", grid=[0,0], size=20)
wetness_field = Text(app, text="-", grid=[1,0], size=100)
Text(app, text="Temp (C)", grid=[0,1], size=20)
temp_c_field = Text(app, text="-", grid=[1,1], size=50)
Text(app, text="Humidity (%)", grid=[0,2], size=20)
humidity_field = Text(app, text="-", grid=[1,2], size=50)
t1.start() # start the thread that updates the readings
app.display()
```

Most of the code is to provide the user interface using `guizero` ([Recipe 7.22](#)). The key steps for the plant monitor are to first create an instance of the `PlantMonitor` class using `pm = PlantMonitor()`, and then access wetness, temperature, and humidity readings using `pm.get_wetness()`, `pm.get_temp()`, and `pm.get_humidity()`, respectively.

## See Also

For full information on the MonkMakes Plant Monitor, see <https://oreil.ly/zOQWq>.

For measuring temperature and humidity using a Sense HAT, see [Recipe 14.12](#).

For an example of a Raspberry Pi Pico using a Plant Monitor, see [Recipe 19.11](#).

## 14.7 Measuring a Voltage

### Problem

You want to measure an analog voltage.

### Solution

The Raspberry Pi GPIO connector has only digital inputs. If you want to measure a voltage, you need to use a separate analog-to-digital converter (ADC).

Use the MCP3008 eight-channel ADC chip. This chip has eight analog inputs, so you can connect up to eight sensors to one of these and interface to the chip using the Raspberry Pi SPI.

To make this recipe, you will need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- MCP3008 eight-channel ADC integrated circuit (IC) (see [“Integrated Circuits”](#))
- 10k $\Omega$  trimpot (see [“Resistors and Capacitors”](#))

[Figure 14-13](#) shows the breadboard layout for using this chip. Make sure that you get the chip facing the proper direction. The little notch in the package should be toward the top of the breadboard.

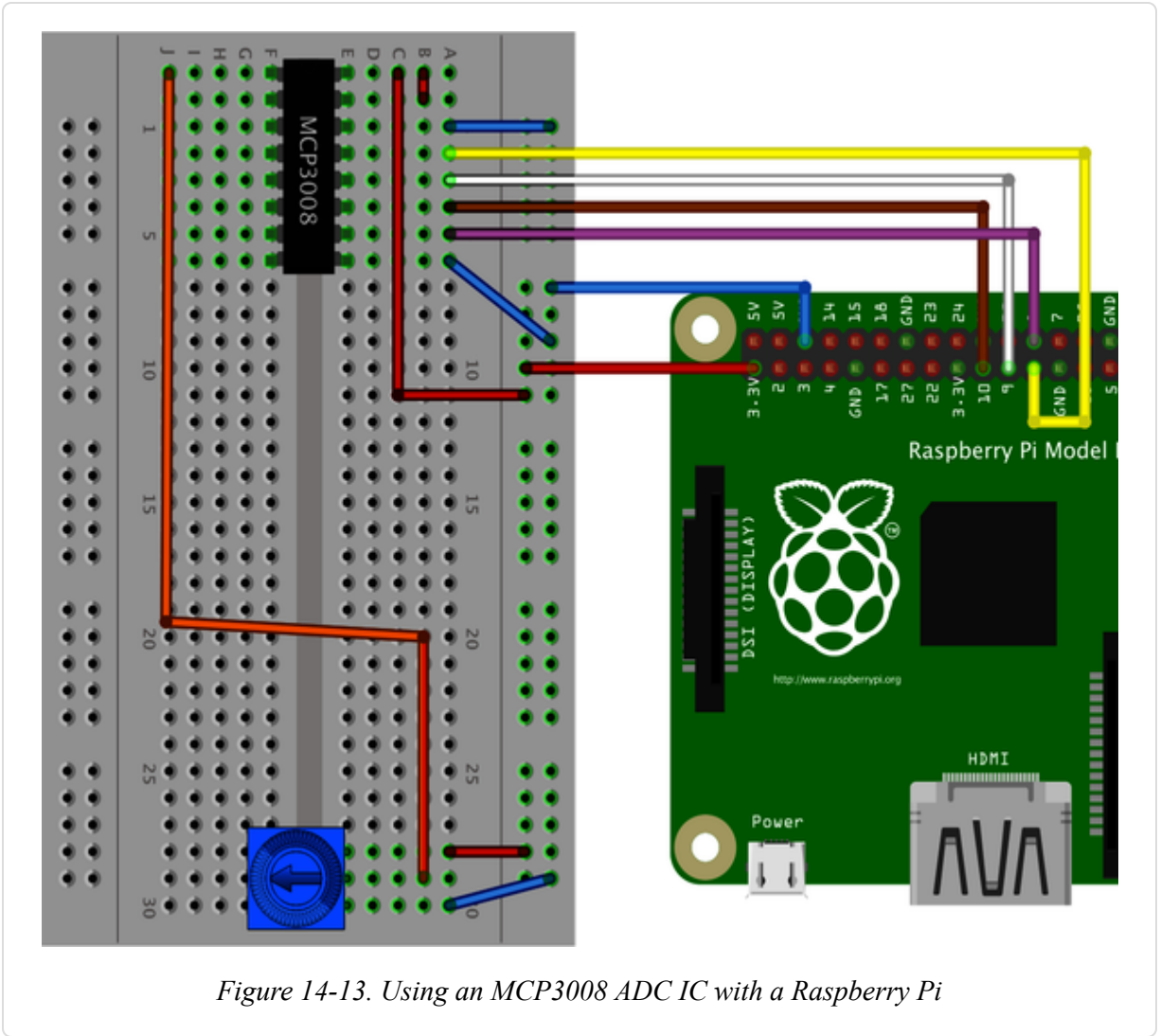


Figure 14-13. Using an MCP3008 ADC IC with a Raspberry Pi

The variable resistor has one end connected to 3.3V and the other to GND, which allows the middle connection to be set to any voltage between 0 and 3.3V.

Before trying the program, make sure you have SPI enabled ([Recipe 10.6](#)). This recipe also explains a bit more about SPI.

Open an editor and paste in the following code (*ch\_14\_adc\_test.py*):

```
from gpiozero import MCP3008
import time

analog_input = MCP3008(channel=0)
```

```
while True:
    reading = analog_input.value
    voltage = reading * 3.3
    print("Reading={:.2f}\tVoltage={:.2f}".format(reading,
voltage))
    time.sleep(1)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

When you run the program, you should see output like this as you turn the variable resistor to alter the voltage:

```
$ python3 ch_14_adc_test.py
Reading=0.60 Voltage=2.00
Reading=0.54 Voltage=1.80
Reading=0.00 Voltage=0.00
Reading=0.00 Voltage=0.00
Reading=0.46 Voltage=1.53
Reading=0.99 Voltage=3.28
```

The readings from the MCP3008 channel are between 0 and 1. You can convert these into a voltage by multiplying by 3.3 (the supply voltage).

## Discussion

The MCP3008 has 10-bit ADCs, so when you take a reading, it gives you a number between 0 and 1023. The `MCP3008` class converts this into a voltage value by multiplying the reading by the voltage range (3.3V) and then dividing it by 1,024.

If you don't need all the features of a *proper* Raspberry Pi, you can use a Raspberry Pi Pico, which has analog inputs available so that you don't need an ADC chip. You can find out about this in [Recipe 19.7](#).

You can combine any of the following recipes that use the MCP3008 to allow readings to be taken from up to eight sensors.

You can also use resistive sensors with the MCP3008 by combining them with a fixed-value resistor and arranging them as a voltage divider (see [Recipes 14.8](#) and [14.9](#)).

## See Also

If you're just interested in detecting the turning of a knob, you can use a rotary encoder instead of a pot ([Recipe 13.7](#)).

You can also detect the position of a pot without the use of an ADC chip by using the step response method ([Recipe 14.1](#)).

Check out the [datasheet for the MCP3008](#).

The Explorer HAT Pro from Pimoroni also has an ADC ([Recipe 10.16](#)).

## 14.8 Reducing Voltages for Measurement

### Problem

You want to measure a voltage, but it is higher than the 3.3V possible using an MCP3008 ([Recipe 14.7](#)).

### Solution

Use a pair of resistors to act as a voltage divider to reduce the voltage to a suitable range.

To try this recipe, you'll need the following:

- Breadboard and jumper wires (see "[Prototyping Equipment and Kits](#)")
- MCP3008 eight-channel ADC IC (see "[Integrated Circuits](#)")
- 10k $\Omega$  resistor (see "[Resistors and Capacitors](#)")
- 3.3k $\Omega$  resistor (see "[Resistors and Capacitors](#)")
- 9V battery and clip lead

[Figure 14-14](#) shows the arrangement for this, using a breadboard. The setup will measure the voltage of the battery.

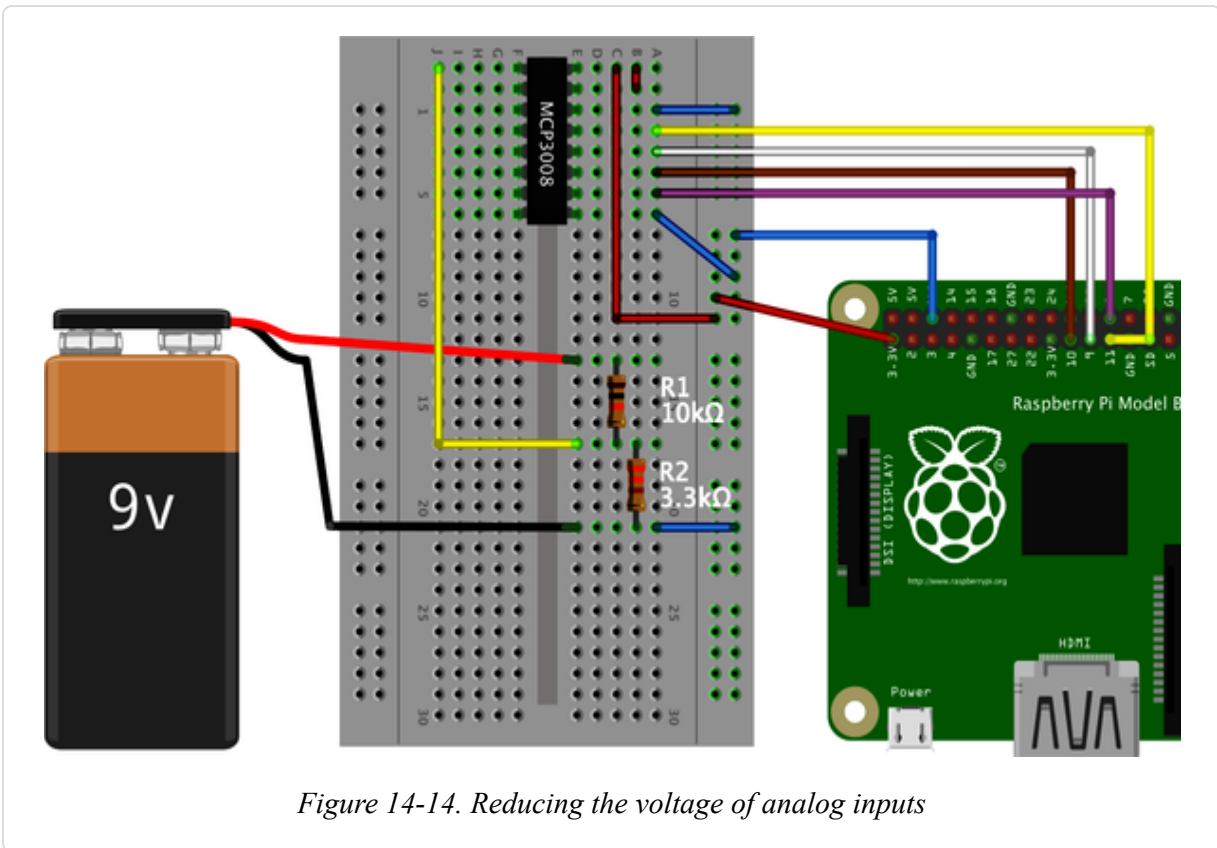


Figure 14-14. Reducing the voltage of analog inputs

## WARNING

Never use this recipe to measure high-voltage AC, or any type of AC for that matter. It is for low-voltage DC only.

Open an editor and paste in the following code (*ch\_14\_adc\_scaled.py*):

```
from gpiozero import MCP3008
import time

R1 = 10000.0
R2 = 3300.0
analog_input = MCP3008(channel=0)

while True:
    reading = analog_input.value
    voltage_adc = reading * 3.3
    voltage_actual = voltage_adc / (R2 / (R1 + R2))
```



```
print("Battery Voltage=" + str(voltage_actual))
time.sleep(1)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

The program is very similar to that of [Recipe 14.7](#). The main difference is the scaling, using the values of the two resistors, which are held in the variables R1 and R2.

When you run the program, the battery voltage is displayed:

```
$ python3 ch_14_adc_scaled.py
Battery Voltage=8.62421875
```

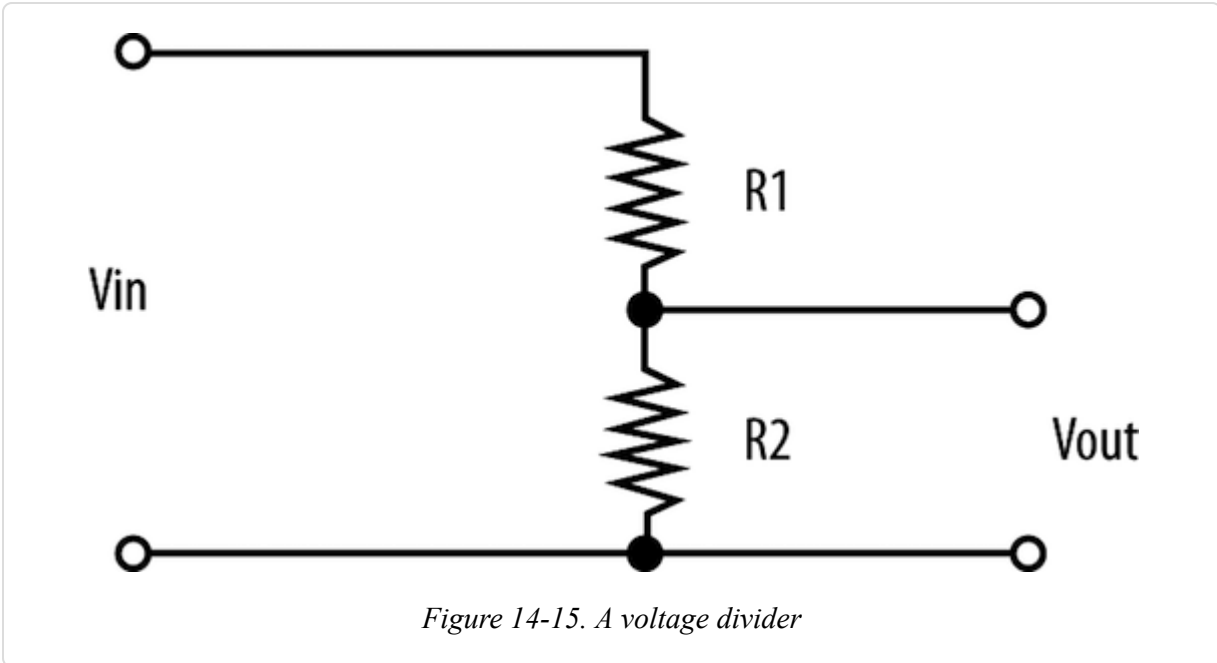
## WARNING

Read the discussion carefully before attaching anything higher than 9V, or you might destroy the MCP3008.

## Discussion

This arrangement of resistors is called a *voltage divider* or sometimes a *potential divider* ([Figure 14-15](#)). The formula for calculating the output voltage, given the input voltage and the values of the two resistors, is as follows:

- $V_{out} = V_{in} * R2 / (R1 + R2)$



This means that if  $R1$  and  $R2$  were both the same value (say,  $1\text{k}\Omega$ ),  $V_{out}$  would be half of  $V_{in}$ .

When choosing  $R1$  and  $R2$ , you also need to consider the current flowing through them. This will be  $V_{in}/(R1 + R2)$ . In the preceding example,  $R1$  is  $10\text{k}\Omega$  and  $R2$  is  $3.3\text{k}\Omega$ , so the current flowing will be  $9\text{V}/13.3\text{k}\Omega = 0.68\text{mA}$ . This is low, but still enough to eventually drain the battery, so do not leave it connected all the time.

## See Also

To avoid the math, you can use an [online resistor calculator](#).

The voltage divider is also used to convert resistance to voltage when using a resistive sensor with an ADC ([Recipe 14.9](#)).

## 14.9 Using Resistive Sensors with an ADC

### Problem

You have a resistive sensor that you wish to use with an MCP3008 ADC chip.

## Solution

Use a potential divider with one fixed resistor and the resistive sensor to convert the resistance of the sensor into a voltage that can be measured with the ADC. As an example, you can remake the light sensor project of [Recipe 14.2](#) to use the MCP3008 instead of the step response technique.

To try this recipe, you'll need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- MCP3008 eight-channel ADC IC (see [“Integrated Circuits”](#))
- 10kΩ resistor (see [“Resistors and Capacitors”](#))
- Photoresistor (see [“Resistors and Capacitors”](#))

[Figure 14-16](#) shows the arrangement for this, using a breadboard.

You can use the exact same program as [Recipe 14.7](#) (*ch\_14\_adc\_test.py*). Covering up the light sensor with your hand changes the readings. You also need to set up SPI on your Raspberry Pi so, if you haven't already done so, follow [Recipe 10.6](#) first.

```
$ python3 ch_14_adc_test.py
Reading=0.60 Voltage=2.00
Reading=0.54 Voltage=1.80
```

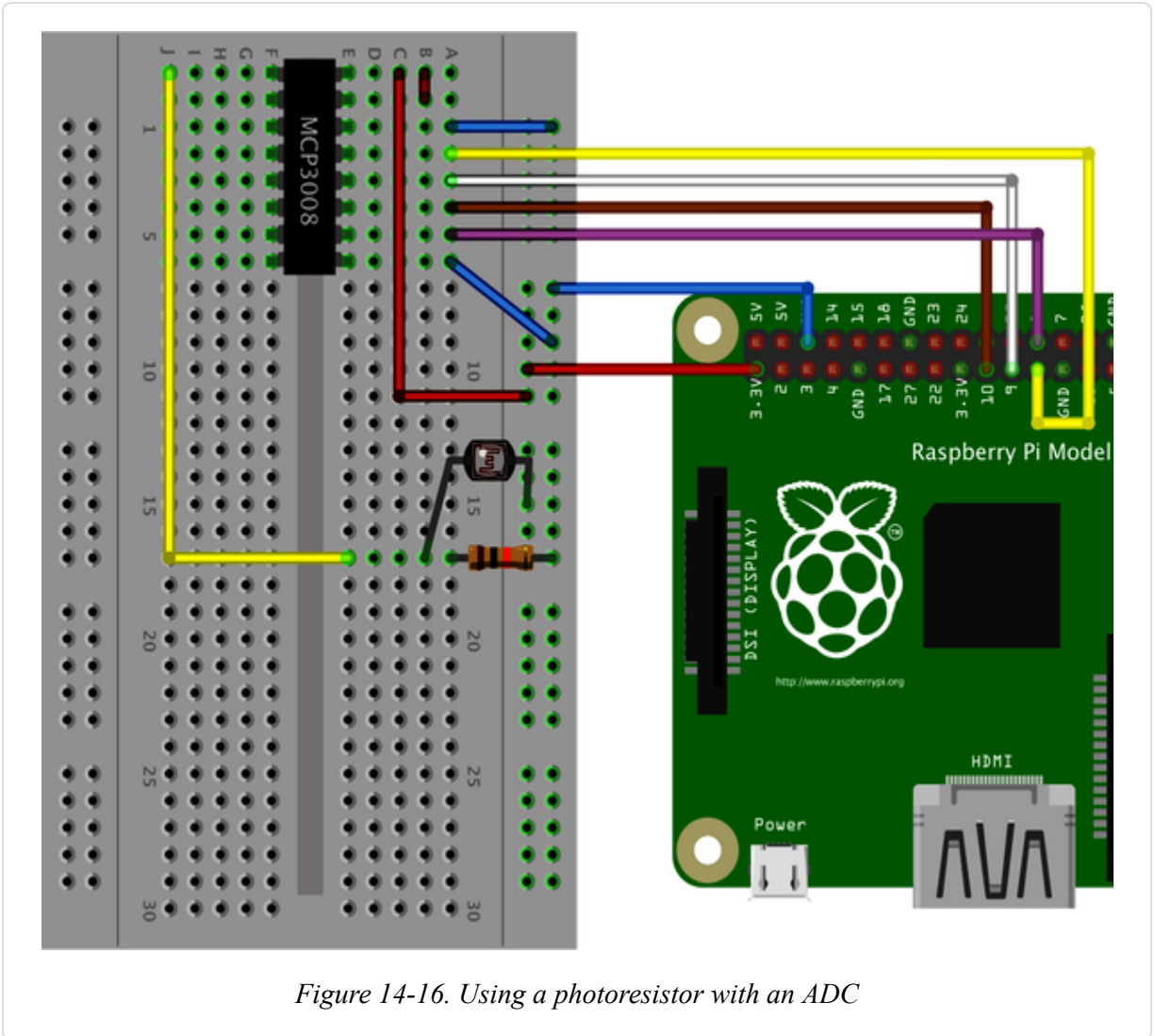


Figure 14-16. Using a photoresistor with an ADC

These readings can be quite a bit different, depending on your photoresistor, but the important thing is that the figure changes as the light level changes.

## Discussion

The choice of fixed-value resistor is not very critical. If the value is too high or too low, you will find that the range of readings is rather narrow. Select a resistor value somewhere between the minimum and maximum resistance of the sensor. You might need to experiment with a few resistors before deciding on one that suits your sensor over the range of readings you're interested in. If in doubt, start with 10kΩ and see how that works.

You can swap out the photoresistor for pretty much any resistive sensor. So, for instance, you could use the gas sensor from [Recipe 14.4](#).

## See Also

To measure light intensity without the complication of an ADC, see [Recipe 14.2](#).

For an example of using more than one channel of the ADC at a time, see [Recipe 14.14](#).

## 14.10 Measuring Temperature with an ADC

### Problem

You want to measure temperature using a TMP36 and an analog-to-digital converter.

### Solution

Use an MCP3008 ADC chip with the TMP36.

To try this recipe, you will need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- MCP3008 eight-channel ADC IC (see [“Integrated Circuits”](#))
- TMP36 temperature sensor (see [“Integrated Circuits”](#))

[Figure 14-17](#) shows the arrangement for this, using a breadboard.

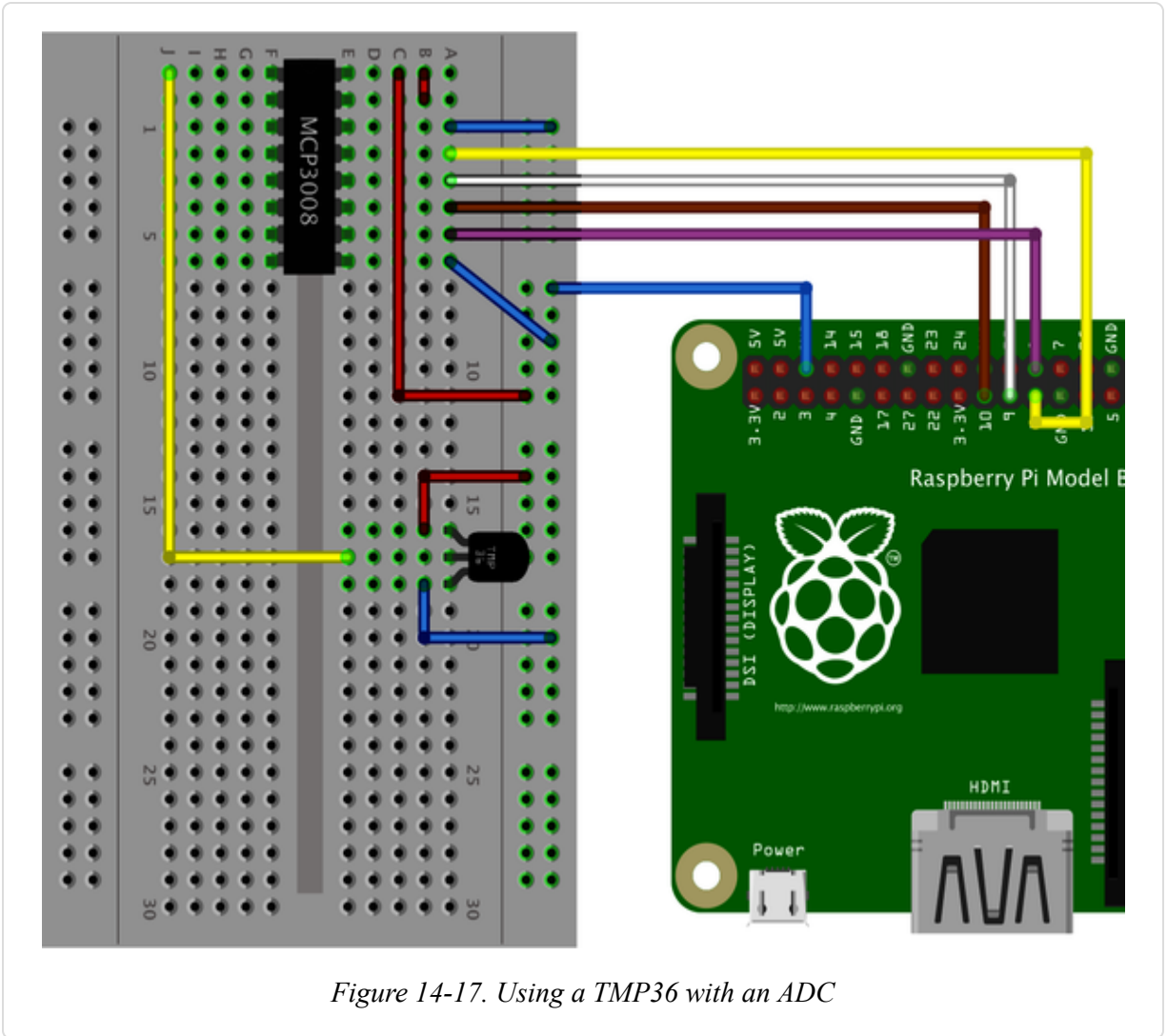


Figure 14-17. Using a TMP36 with an ADC

Make sure that you have the TMP36 facing the proper direction. One side of the package is flat, whereas the other is curved.

You will need to set up SPI on your Raspberry Pi, so if you haven't already done so, follow [Recipe 10.6](#).

Open an editor and paste in the following code (*ch\_14\_adc\_tmp36.py*):

```

from gpiozero import MCP3008
import time

analog_input = MCP3008(channel=0)

while True:
    reading = analog_input.value

```

```
voltage = reading * 3.3
temp_c = voltage * 100 - 50
temp_f = temp_c * 9.0 / 5.0 + 32
print("Temp C={:.2f}\tTemp F={:.2f}".format(temp_c, temp_f))
time.sleep(1)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

The program is based on that of [Recipe 14.7](#). A little bit of additional math calculates the temperature in degrees Celsius and Fahrenheit:

```
$ python3 ch_14_adc_tmp36.py
Temp C=18.64      Temp F=65.55
Temp C=20.25      Temp F=68.45
Temp C=23.47      Temp F=74.25
Temp C=25.08      Temp F=77.15
```

## Discussion

The TMP36 outputs a voltage that is proportional to the temperature. According to the datasheet for the TMP36, the temperature in degrees Celsius is calculated as the voltage (in volts)  $\times 100 - 50$ .

The TMP36 is fine for measuring the approximate temperature but is specified as having an accuracy of only  $2^{\circ}\text{C}$ . This will only get worse if you attach long leads to it. To some extent, you can calibrate an individual device, but, for better accuracy, use a DS18B20 ([Recipe 14.13](#)), which has a stated accuracy of 0.5% over a temperature range of  $-10$  to  $+85$  degrees Celsius. Being a digital device, it should not suffer any loss of accuracy when attached to long leads.

## See Also

Take a look at the [TMP36 datasheet](#).

To measure temperature using a thermistor, see [Recipe 14.3](#).

To measure temperature using a Sense HAT, see [Recipe 14.12](#).

To measure temperature using a digital temperature sensor (DS18B20), see [Recipe 14.13](#).

## 14.11 Measuring the Raspberry Pi CPU Temperature

### Problem

You want to know just how hot your Raspberry Pi's CPU is getting.

### Solution

Use the `gpiozero` library to access the temperature sensor built into the Broadcom chip. Your Raspberry Pi should already have the `gpiozero` library installed. But if it doesn't, you can install it using:

```
$ sudo pip3 install gpiozero
```

The example program `ch_14_cpu_temp.py` will repeatedly print the CPU temperature:

```
import time
from gpiozero import CPUtemperature

while True:
    cpu_temp = CPUtemperature()
    print(cpu_temp.temperature)
    time.sleep(1)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

When you run this program, it reports the temperature in degrees C once a second:



```
$ python3 ch_14_cpu_temp.py
37.485
38.459
36.511
36.998
```

## Discussion

Because the Raspberry Pi's CPU is busy running code, the temperature reported will not have much to do with the ambient temperature, but will be more an indication as to how hard the Pi is working and how well ventilated it is.

## See Also

To measure temperature using a thermistor, see [Recipe 14.3](#).

To measure temperature using a TMP36, see [Recipe 14.10](#).

To measure temperature using a Sense HAT, see [Recipe 14.12](#).

To measure temperature using a digital temperature sensor (DS18B20), see [Recipe 14.13](#).

## 14.12 Measuring Temperature, Humidity, and Pressure with a Sense HAT

### Problem

You want to measure temperature, humidity, and pressure, but you don't really want to have to attach three separate sensors.

### Solution

Use a Raspberry Pi Sense HAT ([Figure 14-18](#)). That way, you get all of those sensors plus some extras, like a display.

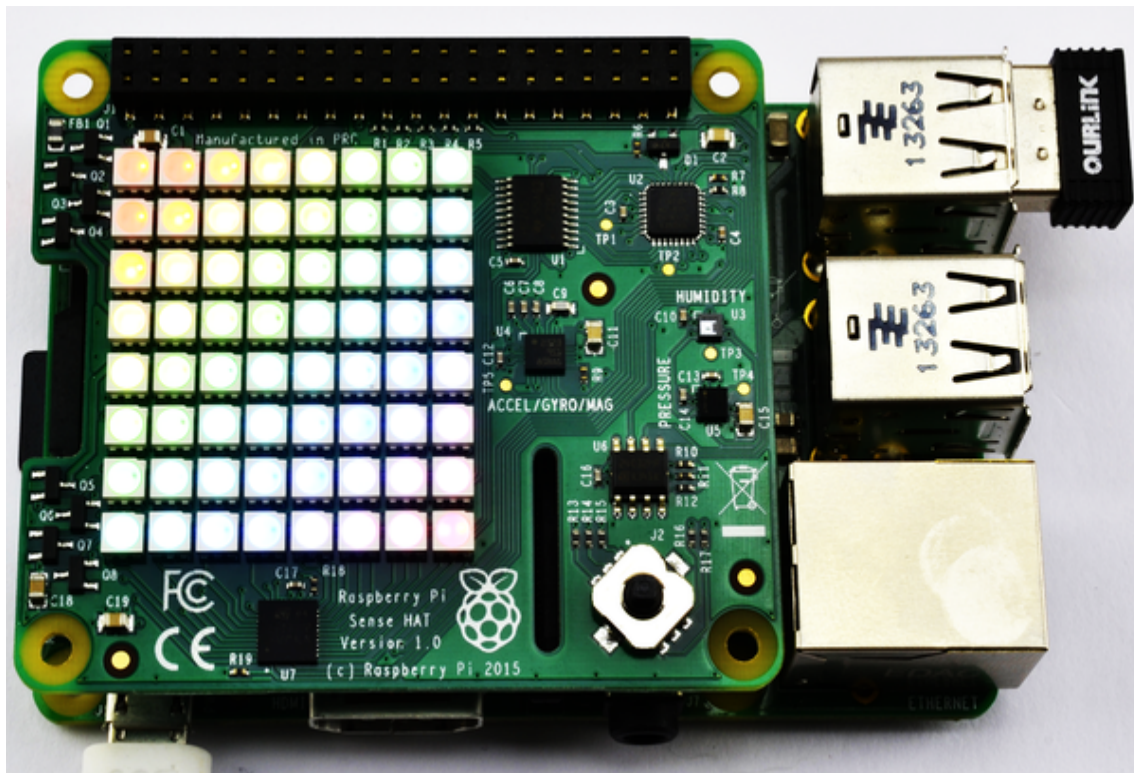


Figure 14-18. A Sense HAT

The Sense Hat software comes pre-installed on Raspberry Pi OS.

Open an editor and paste in the following code (*ch\_14\_sense\_hat\_thp.py*):

```
from sense_hat import SenseHat
import time

hat = SenseHat()

while True:
    t = hat.get_temperature()
    h = hat.get_humidity()
    p = hat.get_pressure()
    print('Temp C: {:.2f} Hum: {:.0f} Pres: {:.0f}'.format(t, h, p))
    time.sleep(1)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

When you run the program, the Terminal displays something like this:

```
$ python3 ch_14_sense_hat_thp.py
Temp C:27.71 Hum:56 Pres:1005
Temp C:27.60 Hum:55 Pres:1005
```

The temperature is in degrees C, the humidity is the relative humidity as a percentage, and the atmospheric pressure is in millibars.

## Discussion

You will find that the temperature readings from the Sense HAT are on the high side. This is because the temperature sensor is built into the humidity sensor and is on the Sense HAT PCB. The Sense HAT generates very little heat (unless you use the display), but the Raspberry Pi under the Sense HAT does get warm and will increase the temperature of the HAT. The best way to avoid this problem is to use a 40-way ribbon cable to move the Sense HAT away from the Raspberry Pi. There are also [attempts to adjust the readings](#) by using the temperature reading of the Raspberry Pi. Personally, I feel that these compensation attempts are probably very specific to the user's posting and are unlikely to produce reliable results.

As well as reading the temperature from the humidity sensor, the pressure sensor also has a temperature sensor built in that you can read like this:

```
t = hat.get_temperature_from_pressure()
```

It is unclear from the documentation as to whether this reading is any more accurate than using the humidity sensor, but for my setup, it reported temperatures about 1 degree Celsius lower than the humidity sensor.

## See Also

To get started with the Sense HAT, see [Recipe 10.15](#).

Check out the [programming reference for the Sense HAT](#).

The Sense HAT also has an accelerometer, a magnetometer ([Recipe 14.15](#)), and a gyroscope ([Recipe 14.16](#)) for navigation-type projects. It also has a

full-color 8×8 LED matrix display ([Recipe 15.3](#)).

## 14.13 Measuring Temperature Using a Digital Sensor

### Problem

You want to measure temperature using an accurate digital sensor.

### Solution

Use the DS18B20 digital temperature sensor. This device is more accurate than the TMP36 used in [Recipe 14.10](#), and it uses a digital interface, so it doesn't require an ADC chip.

Although the interface to this chip is called *one-wire*, this just refers to the data pin. You do need at least one other wire to connect to a one-wire device.

To make this recipe, you will need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- DS18B20 temperature sensor (see [“Integrated Circuits”](#))
- 4.7kΩ resistor (see [“Resistors and Capacitors”](#))

Fit the components onto the breadboard as shown in [Figure 14-19](#). Make sure that you get the DS18B20 facing the proper direction.

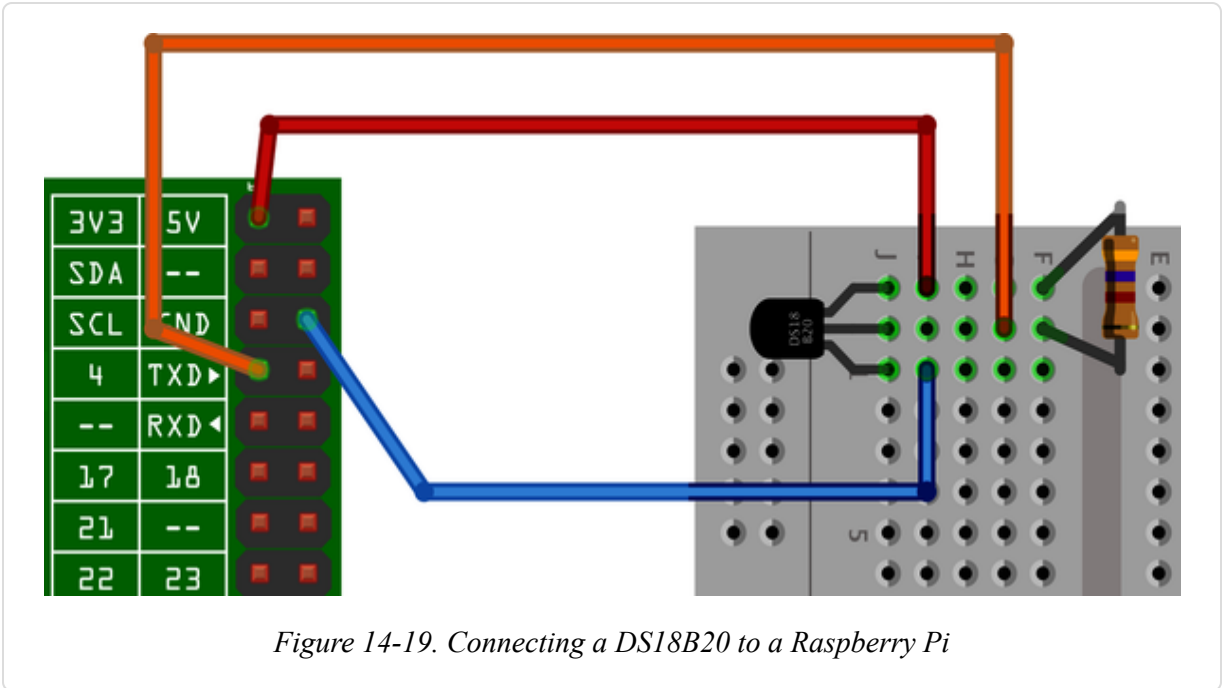


Figure 14-19. Connecting a DS18B20 to a Raspberry Pi

The latest version of Raspberry Pi OS has support for the one-wire interface used by the DS18B20, but you do need to enable it using the Raspberry Pi Configuration tool (Figure 14-20).

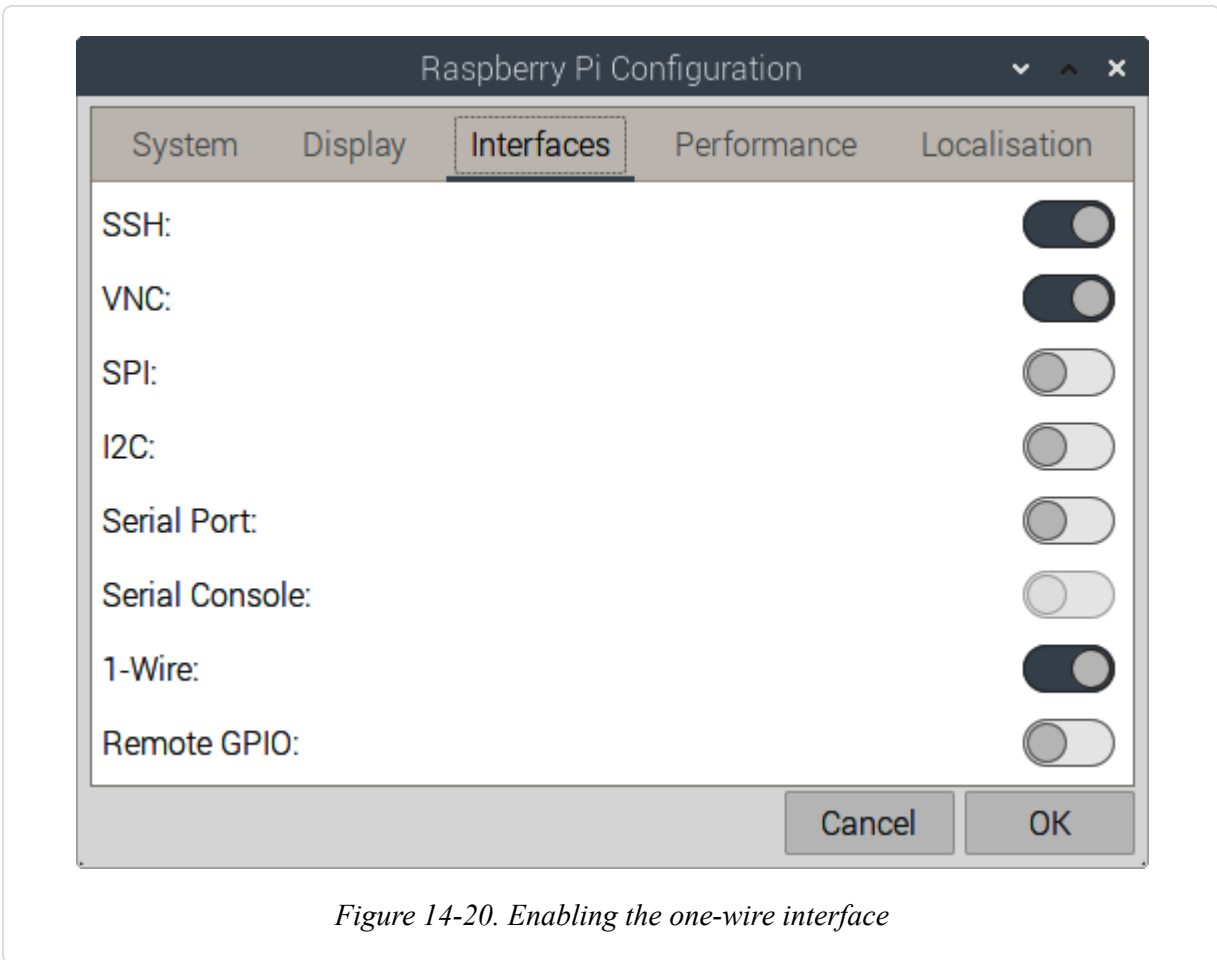


Figure 14-20. Enabling the one-wire interface

You will find the example code for this recipe in `ch_14_temp_DS18B20.py`:

```
import glob, time

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
```

```

equals_pos = lines[1].find('t=')
if equals_pos != -1:
    temp_string = lines[1][equals_pos+2:]
    temp_c = float(temp_string) / 1000.0
    temp_f = temp_c * 9.0 / 5.0 + 32.0
    return temp_c, temp_f

while True:
    temp_c, temp_f = read_temp()
    print('Temp C={:.2f}\ttemp F={:.2f}'.format(temp_c, temp_f))
    time.sleep(1)

```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

When the program is run, it reports the temperature once per second in both degrees Celsius and degrees Fahrenheit:

```

$ python3 ch_14_temp_DS18B20.py
temp C=25.18    temp F=77.33
temp C=25.06    temp F=77.11
temp C=26.31    temp F=79.36
temp C=28.87    temp F=83.97

```

## Discussion

At first sight, the program looks a little odd. The interface to the DS18B20 uses a file-like interface. The file interface for the device will always be in the folder `/sys/bus/w1/devices/`, and the name of the filepath will start with `28`, but the rest of the filepath will be different for each sensor.

The code assumes there will be only one sensor and finds the first folder starting with `28`. To use multiple sensors, use different index values inside the square brackets.

Within that folder will be a file called `w1_slave`, which is opened and read to find the temperature.

The sensor actually returns strings of text like this:

```

81 01 4b 46 7f ff 0f 10 71 : crc=71 YES
81 01 4b 46 7f ff 0f 10 71 t=24062

```

The remainder of the code extracts the temperature part of this message. This appears after `t=` and is the temperature in one-thousandths of a degree Celsius.

The `read_temp` function calculates the temperature in both degrees Celsius and degrees Fahrenheit and returns both values.

In addition to the basic chip version of the DS18B20, you can also buy a version encapsulated in a rugged and waterproof probe.

## See Also

To find out about logging readings, see [Recipe 14.24](#).

This recipe is heavily based on this [Adafruit tutorial](#).

Take a look at the [datasheet for the DS18B20](#).

To measure temperature using a thermistor, see [Recipe 14.3](#).

To measure temperature using a TMP36, see [Recipe 14.10](#).

To measure temperature using a Sense HAT, see [Recipe 14.12](#).

## 14.14 Measuring Acceleration with an MMA8452Q Module

### Problem

You want to connect a triple-axis accelerometer to a Raspberry Pi.

### Solution

Use an I2C accelerometer chip to measure the X, Y, and Z analog outputs.

To try this recipe, you will need the following:

- Breadboard (see [“Prototyping Equipment and Kits”](#))
- Four female-to-female jumper wires (see [“Prototyping Equipment and Kits”](#))



- MMA8452Q triple-axis accelerometer (see “Modules”)

Figure 14-21 shows the arrangement for this, using a breadboard. It uses three channels of the ADC to measure the X, Y, and Z acceleration forces.

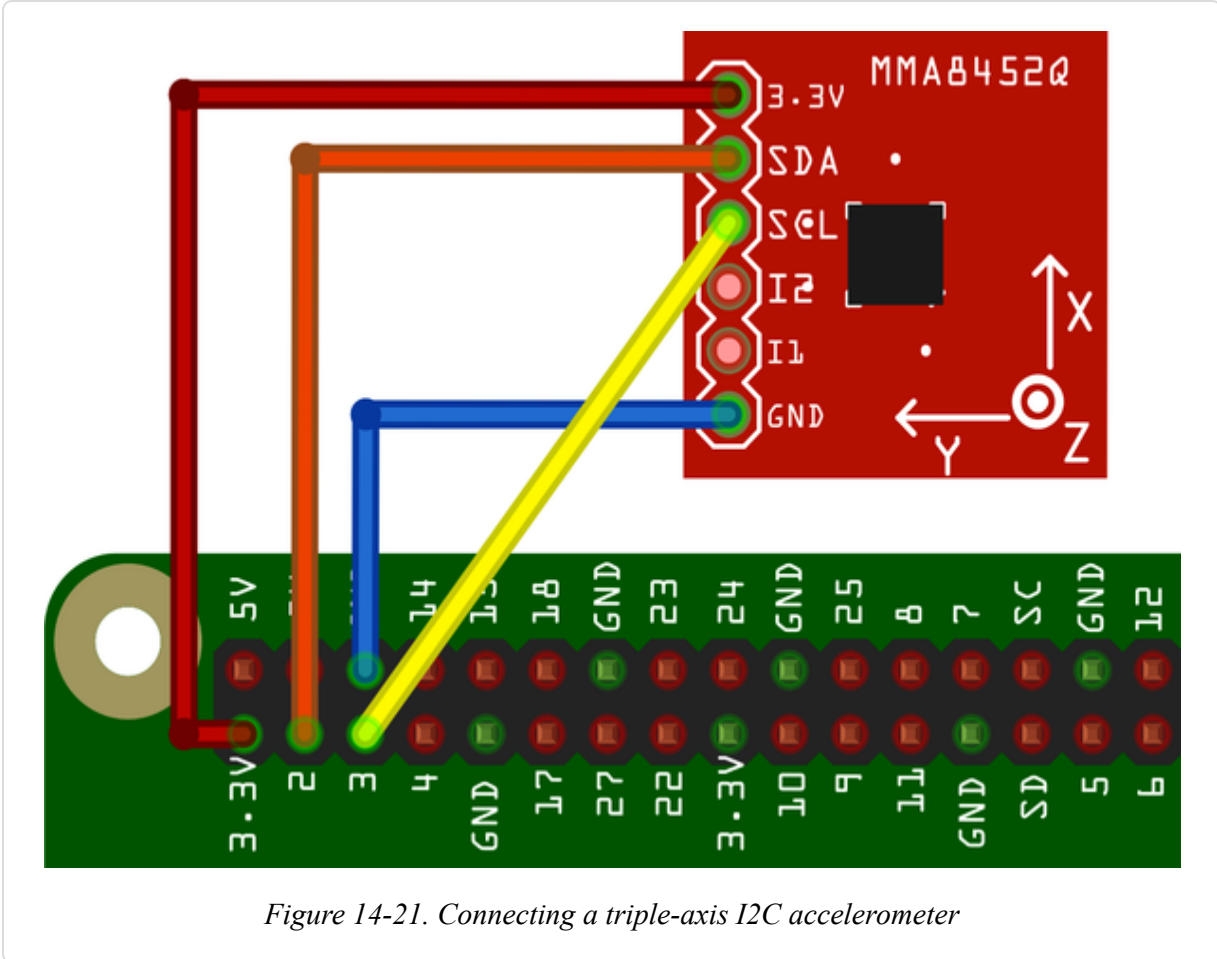


Figure 14-21. Connecting a triple-axis I2C accelerometer

You will need to enable I2C on your Raspberry Pi so, if you have not already done so, follow [Recipe 10.4](#).

The example code for this can be found in the file `ch_14_i2c_acc.py`:

```
import smbus
import time

bus = smbus.SMBus(1)

i2c_address = 0x1D
control_reg = 0x2A
```

```

bus.write_byte_data(i2c_address, control_reg, 0x01) # Start
bus.write_byte_data(i2c_address, 0x0E, 0x00) # 2g range

time.sleep(0.5)

def read_acc():
    data = bus.read_i2c_block_data(i2c_address, 0x00, 7)
    x = (data[1] * 256 + data[2]) / 16
    if x > 2047 :
        x -= 4096
    y = (data[3] * 256 + data[4]) / 16
    if y > 2047 :
        y -= 4096
    z = (data[5] * 256 + data[6]) / 16
    if z > 2047 :
        z -= 4096
    return (x, y, z)

while True:
    print("x={:.6f}\ty={:.6f}\tz={:.6f}".format(x, y, z))
    time.sleep(0.5)

```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

The program reads the three forces corresponding to the accelerations and prints them out:

```

$ python3 ch_14_i2c_acc.py
x=-122.000000 y=56.000000 z=-1023.000000
x=-933.000000 y=251.000000 z=-350.000000
x=-937.000000 y=257.000000 z=-347.000000
x=-933.000000 y=262.000000 z=-350.000000
x=-931.000000 y=259.000000 z=-355.000000
x=-1027.000000 y=-809.000000 z=94.000000

```

Tilt the accelerometer in various directions to see how the readings change. A reading of 0 indicates no net force. Positive values (up to 2047 for 2g) indicate force in one direction and negative in the opposite. You can see that the Z force is close to  $-1023$  (1g) in the first reading where the sensor is horizontal.

## Discussion

You might need to change the I2C address of your device. You can check this by wiring it up and then running the following command:

```
$ sudo i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- 1d -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- --
```

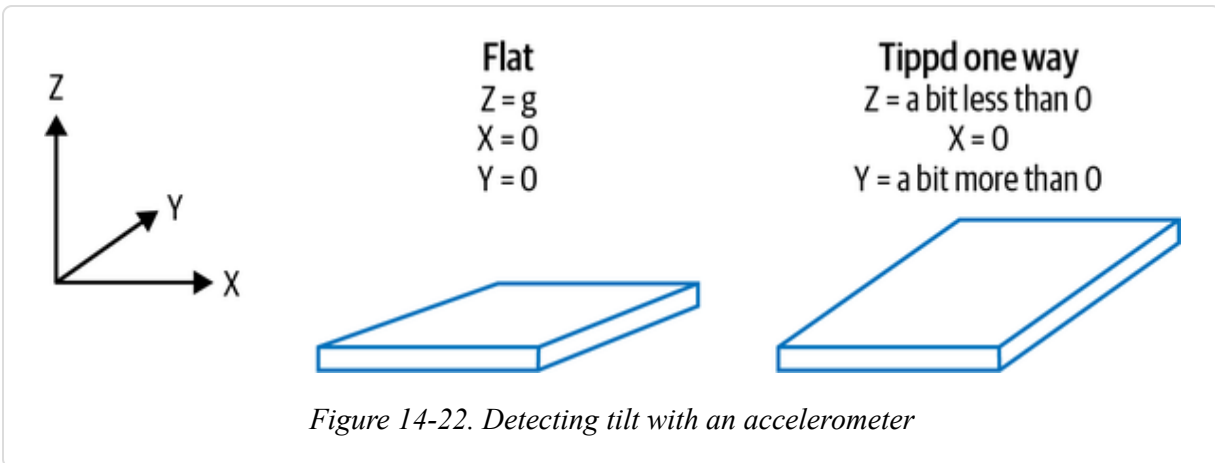
As you can see, in this case, the module that I used has an I2C address of 1d. Thus, that is what the variable `i2c_address` is set to in the program.

Referring to the preceding Python code, the device has a control register to which a command 1 must be written to start the device running. A second configuration command of 0 is then written to register 0x0E, to set the acceleration range of the device to a maximum of 2g. These parameters are specified in the [datasheet for the MMA8452Q](#).

When a reading is required, the I2C bus is read and the data bytes split into the three X, Y, and Z acceleration readings.

The most common use for an accelerometer is to detect tilt. This works because the Z-axis force is dominated by the pull of gravity ([Figure 14-22](#)).

When the accelerometer is tilted in one direction, some of that vertical force of gravity becomes active on another axis of the accelerometer.



We can use this principle to detect when the tilt is past a certain threshold. The following program (*ch\_14\_i2c\_acc\_tilt.py*) illustrates this point:

```
import smbus
import time

bus = smbus.SMBus(1)

i2c_address = 0x1D
control_reg = 0x2A

bus.write_byte_data(i2c_address, control_reg, 0x01) # Start
bus.write_byte_data(i2c_address, 0x0E, 0x00) # 2g range

time.sleep(0.5)

def read_acc():
    data = bus.read_i2c_block_data(i2c_address, 0x00, 7)
    x = (data[1] * 256 + data[2]) / 16
    if x > 2047 :
        x -= 4096
    y = (data[3] * 256 + data[4]) / 16
    if y > 2047 :
        y -= 4096
    z = (data[5] * 256 + data[6]) / 16
    if z > 2047 :
        z -= 4096
    return (x, y, z)

while True:
    x, y, z = read_acc()
    if x > 400:
        print("Left")
```

```
elif x < -400:  
    print("Right")  
elif y > 400:  
    print("Back")  
elif y < -400:  
    print("Forward")  
time.sleep(0.2)
```

When you run the program, you will begin to see direction messages:

```
$ python3 ch_14_i2c_acc_tilt.py  
Left  
Left  
Right  
Forward  
Forward  
Back  
Back
```

You could use this to control a roving robot or a motorized pan-tilt head with a webcam attached.

## See Also

Take a look at the [datasheet for the MMA8452Q](#).

The Sense HAT includes an accelerometer ([Recipe 14.16](#)).

## 14.15 Finding Magnetic North with the Sense HAT

### Problem

You want to use a Sense HAT to detect magnetic north.

### Solution

Use the Python library for the built-in three-axis magnetometer in the Sense HAT.

First follow [Recipe 10.15](#) to install the Sense HAT library.

Open an editor and paste in the following code

(*ch\_14\_sense\_hat\_compass.py*):

```
from sense_hat import SenseHat
import time

sense = SenseHat()

while True:
    bearing = sense.get_compass()
    print('Bearing: {:.0f} to North'.format(bearing))
    time.sleep(0.5)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

When you run this program, you'll see a series of bearing readings:

```
$ python3 ch_14_sense_hat_compass.py
Bearing: 138 to North
Bearing: 138 to North
```

## Discussion

The compass will be sensitive to other nearby sources of magnetic field, so you might find it difficult to get accurate bearings. It can, however, make quite a good magnet detector.

## See Also

You can find documentation for Sense HAT at <https://oreil.ly/dhevo> and <https://oreil.ly/xdFCf>.

To use the Sense HAT to detect a magnet, see [Recipe 14.18](#).

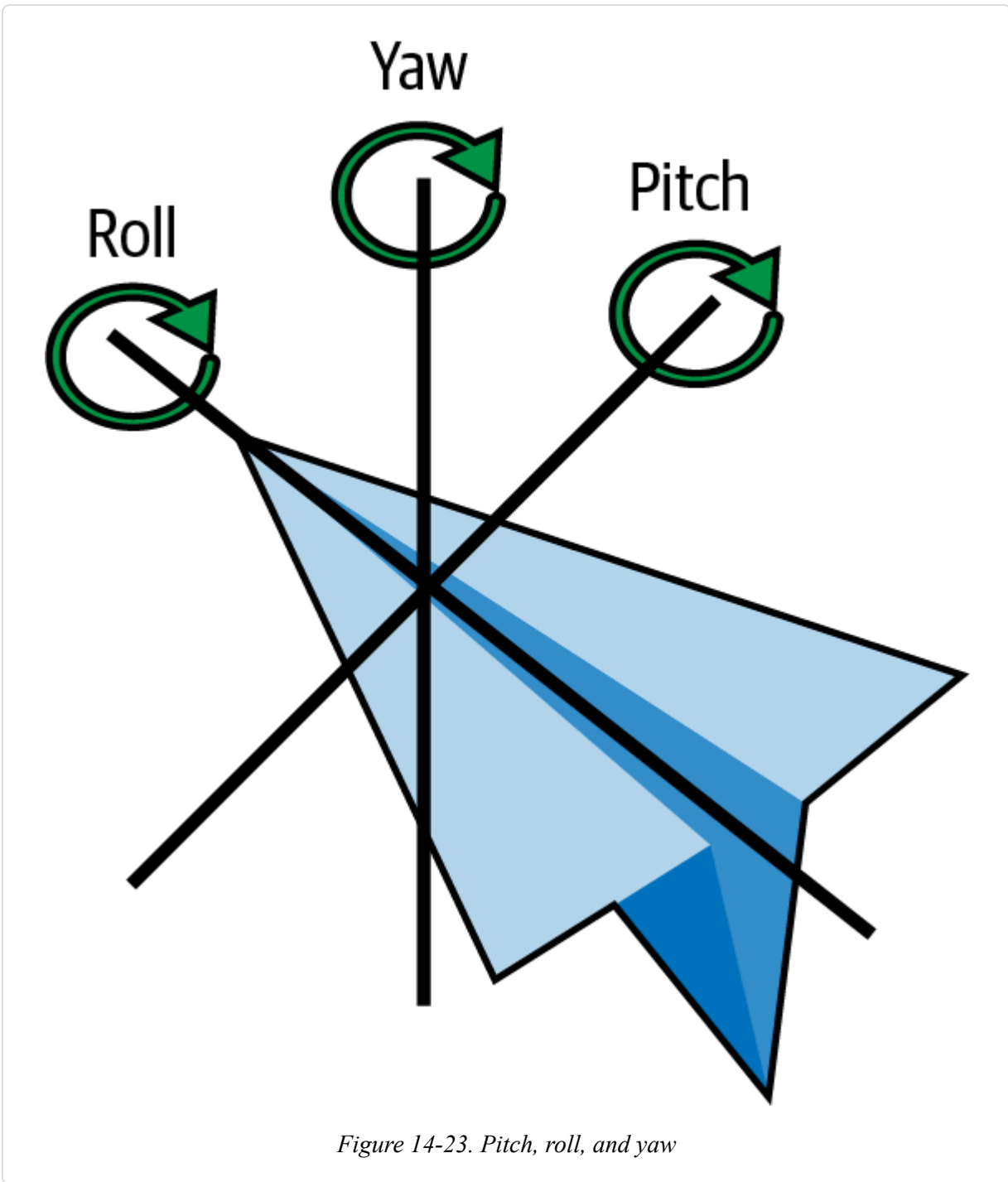
## 14.16 Using the Inertial Measurement Unit of the Sense HAT

### Problem

You want more accurate orientation information from your Raspberry Pi than provided by the accelerometer from [Recipe 14.14](#).

### Solution

Use the Inertial Measurement Unit (IMU) of the Sense HAT. This unit includes a three-axis accelerometer like the one in [Recipe 14.14](#), but it also has a three-axis gyroscope and a magnetometer. The readings from these different sensors are combined to let you get a more accurate orientation for the Sense HAT, expressed as pitch, roll, and yaw (as shown in [Figure 14-23](#)).



Pitch, roll, and yaw are three terms that come from aviation. They are relative to the axis of the plane's flight. The pitch is the angle to the horizontal. The roll is the degree of rotation around the plane's axis of flight (imagine one wing going up and the other down), and yaw is the rotation on the horizontal axis (think changing bearing).



Open an editor and paste in the following code (*ch\_14\_sense\_hat\_orientation.py*):

```
from sense_hat import SenseHat

sense = SenseHat()

sense.set_imu_config(True, True, True)

while True:
    o = sense.get_orientation()
    print("p: {:.0f}, r: {:.0f}, y: {:.0f}".format(o['pitch'],
o['roll'],
o['yaw']))
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

The function `set_imu_config` specifies which of the compass, gyroscope, and accelerometer (in that order) should be used to measure the orientation. Setting all three to `True` means that all are used to make the measurement.

When you run the program, you'll see output similar to this:

```
$ python3 ch_14_sense_hat_orientation.py
p: 1, r: 317, y: 168
p: 1, r: 318, y: 169
```

Try tilting the Sense HAT and Raspberry Pi forward toward the USB ports, and you should see the value of pitch increase.

## Discussion

An accelerometer measures forces on a stationary mass and can therefore measure the degree of tilt by calculating how much of the force supplied by gravity (Z-axis) influences the force measured on the X- and Y-axes.

A gyroscope is different. It measures the force on moving masses (vibrating back and forth) as those masses turn relative to the path of the movement,

using a force called the Coriolis effect.

## See Also

For more information on the Sense HAT's IMU, see <https://oreil.ly/Tr8YV>.

To measure temperature, humidity, and atmospheric pressure, see [Recipe 14.12](#).

The IMU of the Sense HAT can also be used to make a compass and detect the presence of a magnet ([Recipe 14.18](#)).

To find out more about gyroscopes and the Coriolis effect, see <https://oreil.ly/TXHz>.

## 14.17 Sensing a Magnet with a Reed Switch

### Problem

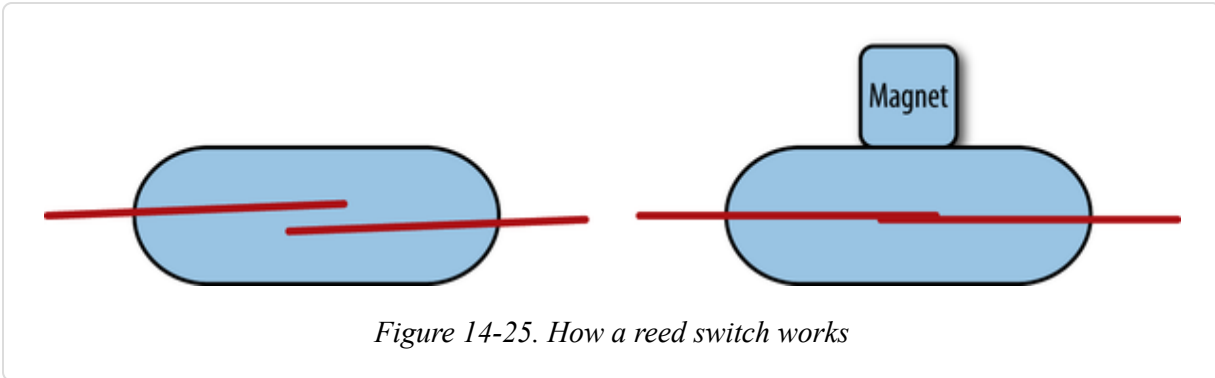
You want to detect the presence of a magnet.

### Solution

Use a reed switch ([Figure 14-24](#)). It works just like a regular switch except it activates only when a magnet is near. [Figure 14-25](#) shows how a reed switch works.



*Figure 14-24. A reed switch*



*Figure 14-25. How a reed switch works*

The two reed contacts are encased in a glass tube. When a magnet is placed near the reed switch, the reeds are pulled together and make contact.

You can use any of the regular switch recipes in [Chapter 13](#) with a reed switch, starting with [Recipe 13.1](#).

## Discussion

Reed switches are a low-tech way of detecting a magnet. They have been around since the 1930s and are extremely reliable. They are commonly used in security systems in which a plastic-encased reed switch is placed on a door frame, with a fixed magnet in another plastic enclosure on the door itself. When the door opens, the reed switch contacts open, triggering the alarm.

## See Also

To detect a magnet using the magnetometer of the Sense HAT, see [Recipe 14.18](#).

## 14.18 Sensing a Magnet with the Sense HAT

### Problem

You want to detect the presence of a magnet using a Sense Hat with built-in magnetometer and a Python program.

## Solution

Use the Sense HAT's Python library to interface with its magnetometer.

Open an editor and paste in the following code

(*ch\_14\_sense\_hat\_magnet.py*):

```
from sense_hat import SenseHat
import time

hat = SenseHat()
fill = (255, 0, 0)

while True:
    reading = int(hat.get_compass_raw()['z'])
    if reading > 200:
        hat.clear(fill)
        time.sleep(0.2)
    else:
        hat.clear()
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

When the magnet draws close to the Sense HAT, the LEDs will all turn red for one-fifth of a second.

## Discussion

It doesn't matter which axis of the compass data you use; all three will be greatly disrupted by the presence of a fixed magnet.

## See Also

To detect a magnet using a reed switch, see [Recipe 14.17](#).

For other ways to use the Sense HAT display, see [Recipe 15.3](#).

## 14.19 Measuring Distance Using Ultrasound

## Problem

You want to measure distance using an ultrasonic range finder.

## Solution

Use a low-cost HC-SR04 range finder. This device needs two GPIO pins: one to trigger the pulse of ultrasound and the other to monitor how long it takes for the echo to return.

To make this recipe, you will need the following:

- Breadboard and jumper wires (see “[Prototyping Equipment and Kits](#)”)
- HC-SR04 range finder (eBay)
- 470Ω resistor (see “[Resistors and Capacitors](#)”)
- 270Ω resistor (see “[Resistors and Capacitors](#)”)

Fit the components onto the breadboard as shown in [Figure 14-26](#). The resistors are necessary to reduce the echo output of the range finder from 5V to 3.3V (see [Recipe 10.12](#)).

Open an editor and paste in the following code (*ch\_14\_ranger.py*):

```
from gpiozero import DistanceSensor
from time import sleep

sensor = DistanceSensor(echo=18, trigger=17)
while True:
    cm = sensor.distance * 100
    inch = cm / 2.5
    print("cm={:.0f} \tinches={:.0f}".format(cm, inch))
    sleep(0.5)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

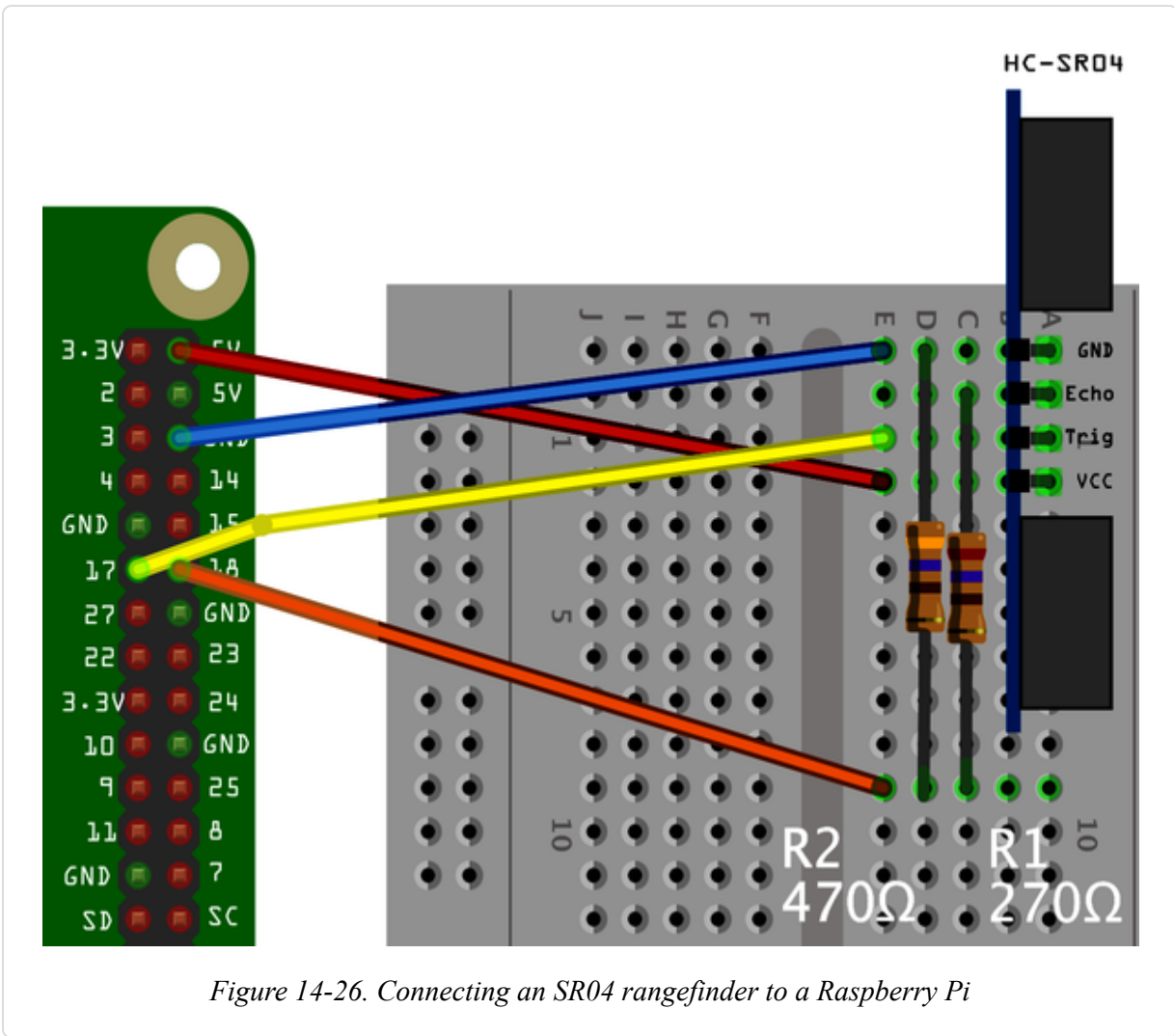


Figure 14-26. Connecting an SR04 rangefinder to a Raspberry Pi

The operation of the program is described in the Discussion. When the program is run, it reports the distance in both centimeters and inches once per second. Use your hand or some other obstacle to change the reading:

```
$ python3 ch_14_ranger.py
cm=154.7 inches=61.8
cm=12.9 inches=5.1
cm=14.0 inches=5.6
cm=20.2 inches=8.0
```

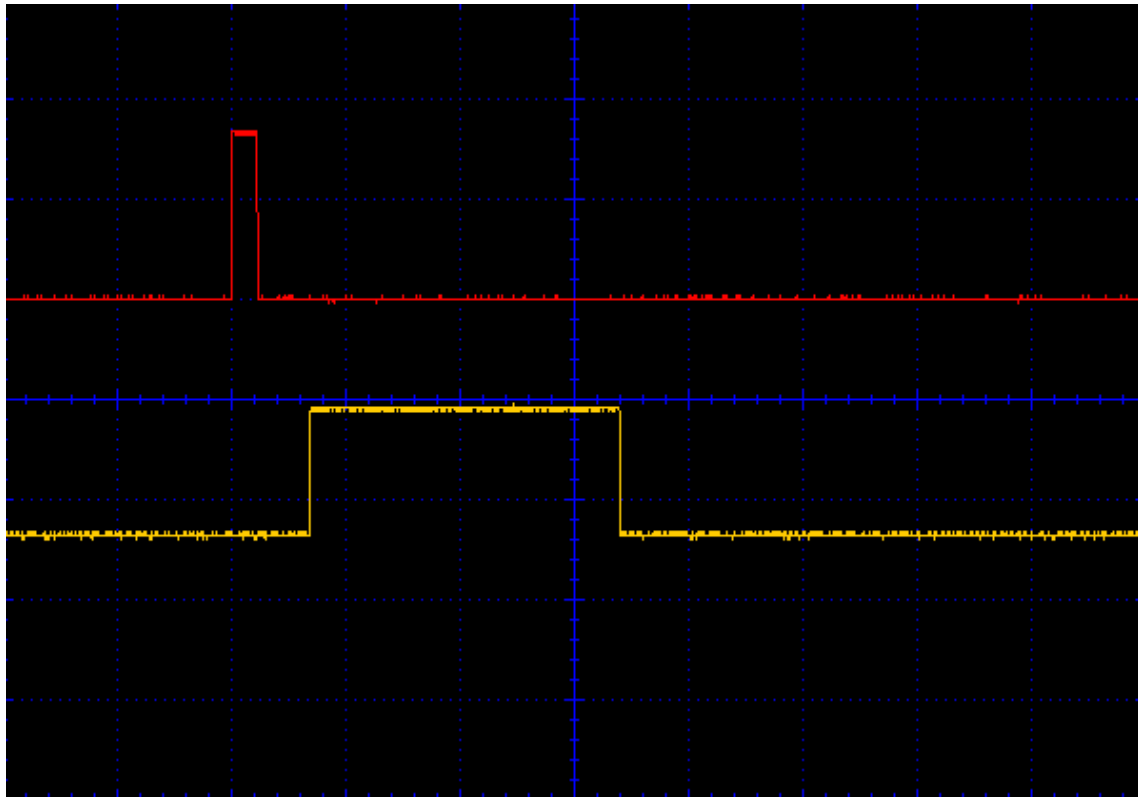
## Discussion

Although several ultrasonic range finders are available, the type used here is easy to use and low cost. It works by sending a pulse of ultrasound and then measuring the amount of time taken for the echo to be received. One of the round ultrasonic transducers on the front of the device is the transmitter, and the other is the receiver.

This process is controlled from the Raspberry Pi. The difference between this type of device and more expensive models is that the more expensive versions include their own microcontroller, which carries out all the necessary timing and provides an I2C or serial interface to return a final reading.

When you are using one of these sensors with a Raspberry Pi, the *trig* (trigger) input of the range finder is connected to a GPIO output, and the *echo* output of the range finder is connected to a GPIO input on the Raspberry Pi after having its voltage range lowered from 5V to a safe 3.3V.

**Figure 14-27** shows an oscilloscope trace of the sensor in action. The top trace is connected to *trig*, and the bottom trace is connected to *echo*. You can see that first the *trig* pin is taken high for a short pulse. There is then a short delay before the *echo* pin goes high. This stays high for a period that is proportional to the distance from the sensor.



*Figure 14-27. The oscilloscope trace for trigger and echo*

This program makes use of the `DistanceSensor` class of the `gpiozero` library, which takes care of the pulse generation and measurement for us.

This method of measuring distance is not terribly accurate because temperature, pressure, and relative humidity all alter the speed of sound (about 30 cm per millisecond) and hence the distance readings.

## See Also

Take a look at the [datasheet for the ultrasonic range finder](#).

The documentation for the `DistanceSensor` class is available at <https://oreil.ly/XYk3m>.



## 14.20 Measuring Distance Using a Time-of-Flight Sensor

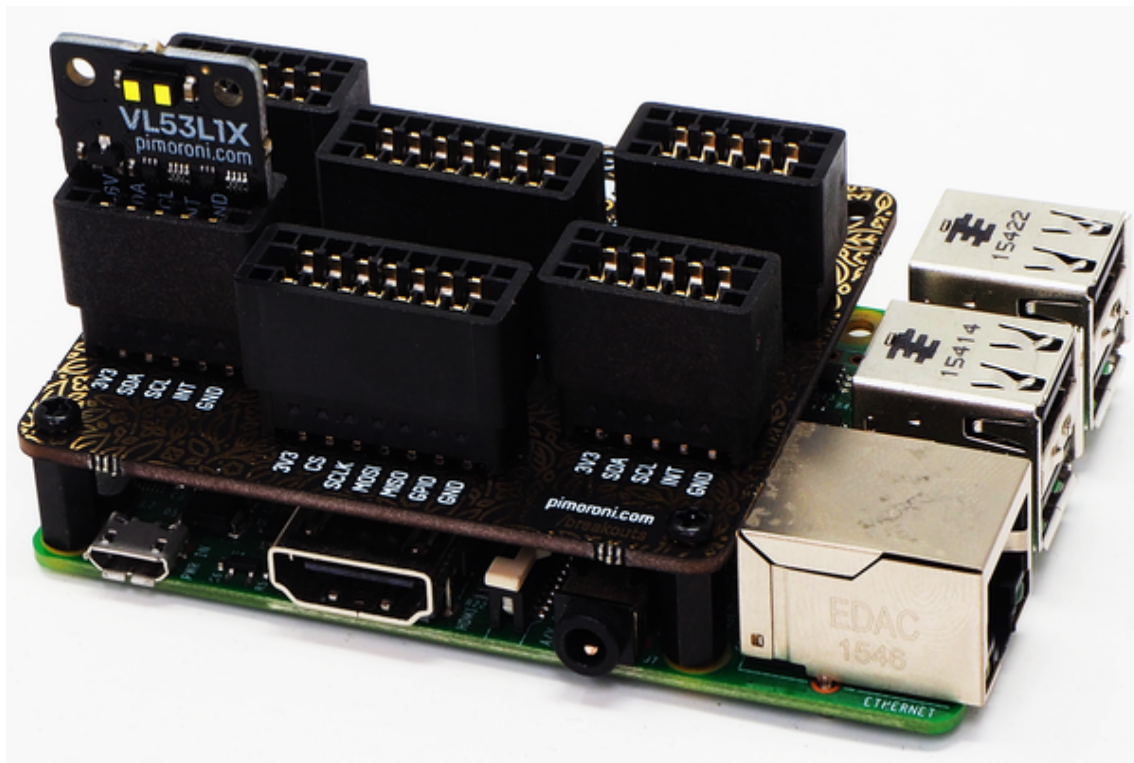
### Problem

You want to measure distance without using ultrasound (perhaps you have animals who would be frightened by ultrasound, or you want to measure distance more accurately).

### Solution

Use a VL53L1X I2C time-of-flight (ToF) sensor. These sensors are more expensive than their ultrasound equivalents. However, because they use light rather than sound, they are more accurate.

The most common of these devices is the VL53L1X, and it's available as a low-cost module from eBay as well as from other suppliers. The Pimoroni device we are using here has the advantage that it is compatible with the Pimoroni Breakout Garden system and can therefore be plugged in without any soldering. [Figure 14-28](#) shows the ToF sensor and Breakout Garden.



*Figure 14-28. A VL53L1X ToF sensor and Pimoroni Breakout Garden*

To make this project, you will need either:

- Pimoroni Breakout Garden (see “[Prototyping Equipment and Kits](#)”)
- Pimoroni VL53L1X distance sensor (see “[Modules](#)”)

or:

- Generic VL53L1X distance sensor module (see “[Modules](#)”)
- Four male-to-female jumper wires (see “[Prototyping Equipment and Kits](#)”)

Connecting this I2C device to a Raspberry Pi is just like connecting any other. The device will work at 3V so, in addition to connecting 3V and GND, you should connect the SDA pin of the sensor to the SDA pin of the Raspberry Pi (also called GPIO 2) and the SCL pin of the sensor to the SCL pin of the Raspberry Pi (GPIO 3).

If you chose to use the Breakout Garden, just make sure you place the sensor in the appropriate orientation (see [Figure 14-28](#)). If you are using jumper wires, connect the devices as follows:

- The VCC pin of the VL53L1X to 3V on the Raspberry Pi
- The GND pin of the VL53L1X to GND on the Raspberry Pi
- The SDA pin of the VL53L1X to GPIO 2 (SDA) on the Raspberry Pi
- The SCL pin of the VL53L1X to GPIO 3 (SCL) on the Raspberry Pi

The VL53L1X uses I2C, so you need to enable this by following [Recipe 10.4](#). After you've enabled it, run the following commands to install the software for the VL53L1X:

```
$ sudo pip3 install smbus2
$ sudo pip3 install vl53l1x
```

The test program for this recipe (*ch\_14\_tof.py*) prints the distance measurement in millimeters once per second:

```
import VL53L1X, time

tof = VL53L1X.VL53L1X(i2c_bus=1, i2c_address=0x29)
tof.open()
tof.start_ranging(1) # Start range1=Short 2=Medium 3=Long

while True:
    mm = tof.get_distance() # Grab the range in mm
    print("mm=" + str(mm))
    time.sleep(1)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

Note that if your device has a different I2C address, you might need to change `0x29` to the address of your device.

## Discussion

The VL53L1X ToF sensor is an amazing little device that contains a low-power infrared laser and receiver as well as all the electronics needed to communicate over I2C.

The module works on a similar principle to that of the ultrasonic range finder of [Recipe 14.19](#), except that instead of gauging distance by measuring the time it takes for sound to travel to a target and back as a reflection, the ToF sensor measures the time it takes for a pulse of laser light to bounce back from a target.

## See Also

To measure distance using ultrasonics, see [Recipe 14.19](#).

See the datasheet for the [VL53L1X](#).

## 14.21 Adding Touch Sensing to Your Raspberry Pi

### Problem

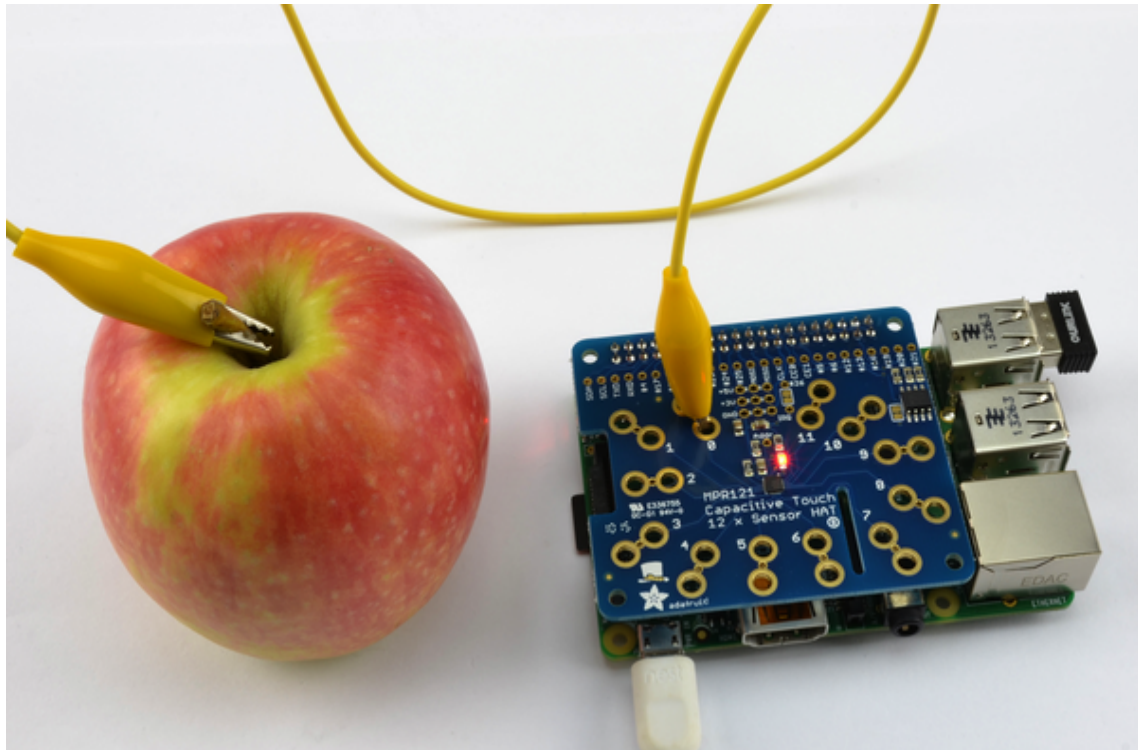
You want to provide a touch interface to your Raspberry Pi.

### Solution

Use an Adafruit Capacitive Touch HAT ([Figure 14-29](#)).

Touch sensors are a lot of fun and are great for educational use. You can attach anything that conducts even just a little bit of electricity—including fruit. A popular project is to construct a fruit keyboard using alligator clips to attach a variety of fruits and vegetables to the sense terminals on the board. Then, as you touch the different pieces of fruit, different sounds are made.

The Adafruit Capacitive Touch HAT uses the Raspberry Pi's I2C interface. You also need SPI tools installed so, if you have not already done so, follow [Recipes 10.4](#) and [10.6](#).



*Figure 14-29. An Adafruit Capacitive Touch HAT attached to an apple*

To install the Python library for the HAT, run the following commands:

```
$ pip3 install adafruit-blinka
$ pip3 install adafruit-circuitpython-mpr121
```

To test out the Capacitive Touch HAT, run the following program (*ch\_14\_touch.py*):

```
import time
import board
import busio
import adafruit_mpr121
i2c = busio.I2C(board.SCL, board.SDA)
mpr121 = adafruit_mpr121.MPR121(i2c)

while True:
    if mpr121[0].value:
        print("Pin 0 touched!")
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

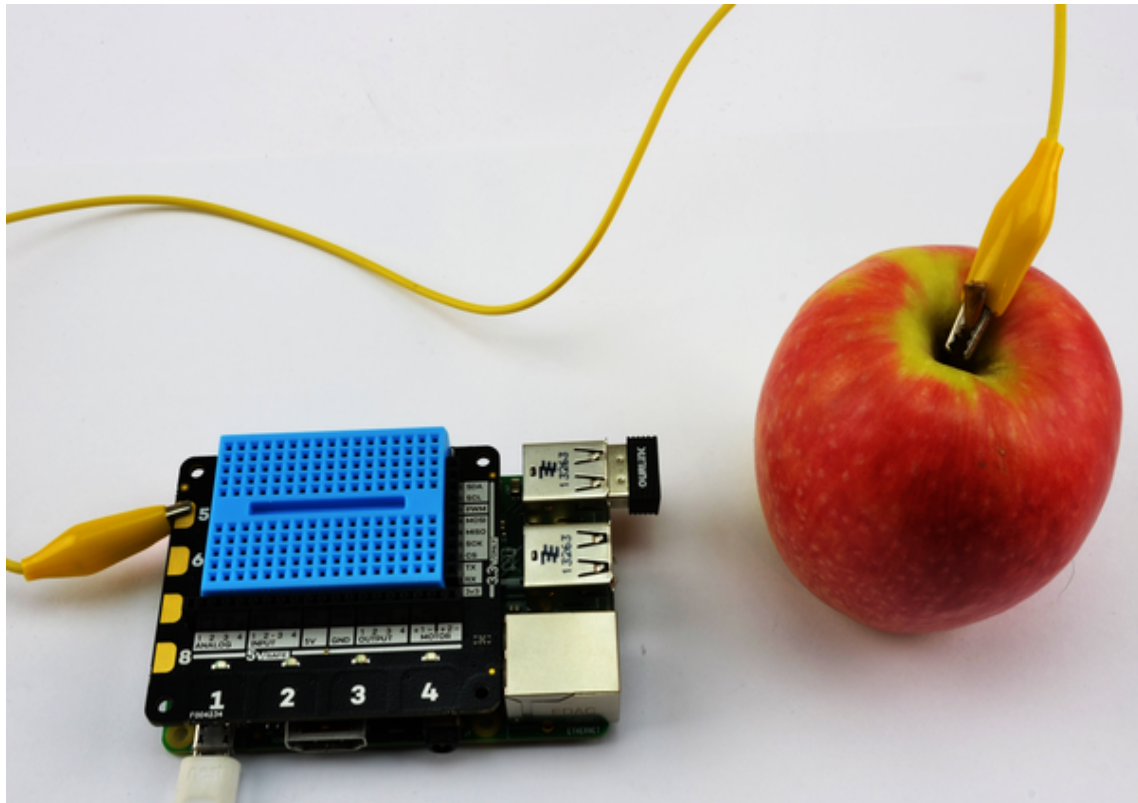
When you touch the pad labeled 0 with your finger, you should see output like this:

```
$ python3 ch_14_touch.py
Pin 0 touched!
Pin 0 touched!
```

You can either just touch the connection pads or connect the pads to a piece of fruit using an alligator clip, as shown in [Figure 14-29](#).

## Discussion

The Adafruit Capacitive Touch HAT has 12 touch contacts. If you need only a few touch contacts, you can use the Pimoroni Explorer HAT Pro, which has four alligator-clip-compatible contacts ([Figure 14-30](#)).



*Figure 14-30. The Explorer HAT Pro with fruit*

To use the Explorer HAT Pro's touch contacts, first follow [Recipe 10.16](#) to install the library for the HAT.

In addition to the four terminals on the side that are designed for alligator clips, four touch switches labeled 1 to 4 also use the touch interface.

## **See Also**

More information is available in the [Adafruit Touch HAT documentation](#) and the [Explorer HAT Pro documentation](#).

## **14.22 Reading Smart Cards with an RFID Reader/Writer**

## Problem

You want to read from and write to radio-frequency identification (RFID) smart cards.

## Solution

Get a low-cost RC-522 RFID card reader/writer and use the `SimpleMFRC522` Python library.

For this recipe, you will need the following:

- RC-522 card reader; this is often sold with a selection of RFID tags to use (see “[Modules](#)”)
- Seven female-to-female jumper wires (see “[Prototyping Equipment and Kits](#)”)

[Figure 14-31](#) shows the RC-522 wired up to a Raspberry Pi. The RC-522 uses the Raspberry Pi’s SPI interface, so you will need to follow [Recipe 10.6](#).



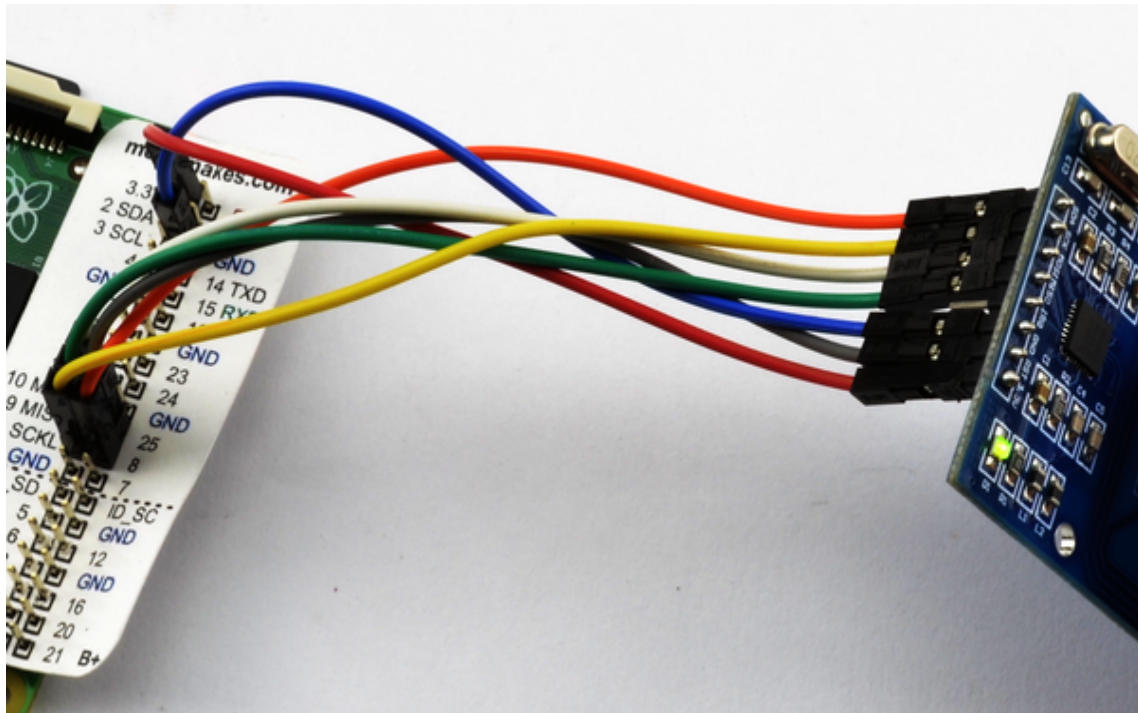


Figure 14-31. Connecting a Raspberry Pi and RC-522

Table 14-1 shows the connections you need to make using the jumper wires, with suggested colors for the leads to make it easy to identify which lead is which.

Table 14-1. Connecting a Raspberry Pi and an RFID reader/writer

Lead color	RC-522 pin	Raspberry Pi pin
Orange	SDA	GPIO8
Yellow	SCK	SCLK / GPIO11
White	MOSI	MOSI / GPIO10
Green	MISO	MISO/GPIO9
Blue	GND	GND
Gray	RST	GPIO25
Red	3.3V	3.3V

Note that, although the RC-522 pin has some pins marked SDA and SCL, as if the device were using I2C, in this recipe the device uses the Raspberry Pi's SPI interface.

To use the module, first fetch the Clever Card Kit software using the commands that follow. This will install all of the prerequisite software for the RC-522. You will need to reboot when it's finished installing:

```
$ wget http://monkmake.com/downloads/mmccck.sh
$ chmod +x mmccck.sh
$ ./mmccck.sh
```

To test the reader, run the program in the *clever\_card\_kit* directory called *01\_read.py*:

```
$ cd ~/clever_card_kit
$ python3 01_read.py
Hold a tag near the reader
894922433952

894922433952

894922433952
```

When you hold a card next to the RC-522, the RFID tag inside the card's unique number will be printed out. When you've had enough of reading cards, hit Ctrl-C. Here's the code:

```
import RPi.GPIO as GPIO
import SimpleMFRC522

reader = SimpleMFRC522.SimpleMFRC522()

print("Hold a tag near the reader")

try:
    while True:
        id, text = reader.read()
        print(id)
        print(text)
```

```
finally:
    print("cleaning up")
    GPIO.cleanup()
```

The `RPi.GPIO` is imported merely so that it can be used to cleanup the GPIO pins when the program exits. The function call `reader.read()` will wait for an RFID tag to come near the reader/writer and return the card's unique number (`id`) and any text message stored on the card (`text`).

You can't change the unique number assigned to each tag during manufacture, but you can store small amounts of data on a card. To do this, use the program *02\_write.py*:

```
$ python3 02_write.py
New Text: Raspberry Pi
Now scan a tag to write
written
894922433952
Raspberry Pi
New Text:
```

Having written some text onto the card, you can check whether it's there by using *01\_read.py*. The code for *02\_write.py* is:

```
import RPi.GPIO as GPIO
import SimpleMFRC522

reader = SimpleMFRC522.SimpleMFRC522()

try:
    while True:
        text = input('New Text: ')
        print("Now scan a tag to write")
        id, text = reader.write(text)
        print("written")

        print(id)
        print(text)
finally:
    print("cleaning up")
    GPIO.cleanup()
```

Using the full features of an RFID card reader is pretty complex. The `SimpleMFRC522` class greatly simplifies this process into basic reading and writing of text to the card. After instantiating the `SimpleMFRC522` class, you can just call `read` to read from a card that is being scanned. This returns both the ID and any text written on the card. The `write` method returns the same values, but also allows you to specify the text to be written to the card.

## Discussion

RFID tags are available in all sorts of shapes and sizes. But a number of different standards are used that operate at different frequencies and use different communication protocols, so when looking for cards to work with the RC-522, look for cards described as 13.56 MHz. The `SimpleMFRC522` code is also a bit fussy about the cards it will work with, so if you plan to use it, also look for tags described as being Mifare 1k compatible.

Because of the differences between what cards can store in their memory and how they store it, it's often better not to rely on storing data on the tag but rather to use the card's unique ID. You can then store data against that unique key. The programs `05_launcher_setup.py` and `05_launcher.py` in the `clever_card_kit` directory show how you can use this approach using table data stored in a pickle file (see [Recipe 7.9](#)).

## See Also

More information is available in the [full documentation of SimpleMFRC522 code](#).

# 14.23 Displaying Sensor Values

## Problem

You have a sensor wired to your Raspberry Pi, and you want a big digital display of the reading on the screen.

## Solution

Use the `guizero` library to open a window, and display the reading on it in a large font (Figure 14-32).

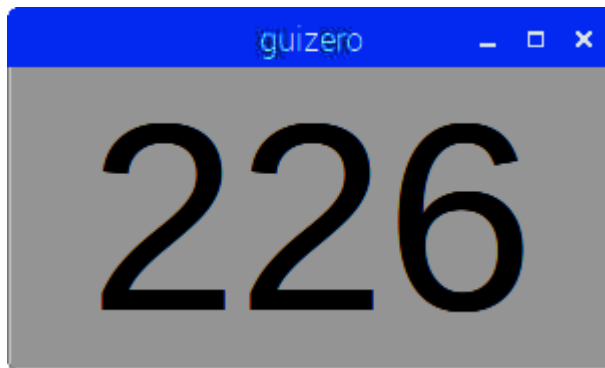


Figure 14-32. Displaying a sensor reading using `guizero`

This example uses data from the ToF range finder of [Recipe 14.20](#). So complete that recipe first if you want to try out this example. Alternatively, most of the other sensor recipes in this chapter could be adapted to work with this recipe. Test out the recipe by opening an editor and pasting in the following code (`ch_14_gui_sensor_reading.py`):

```
import VL53L1X, time
from guizero import App, Text

tof = VL53L1X.VL53L1X(i2c_bus=1, i2c_address=0x29)
tof.open()
tof.start_ranging(1)

def update_reading():
    mm = tof.get_distance()
    reading_text.value = str(mm)

app = App(width=300, height=150)
reading_text = Text(app, size=100)
```

```
reading_text.repeat(1000, update_reading)
app.display()
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

The function `update_reading` gets a new reading from the range finder (or whatever sensor you have chosen to use) and sets the value of `reading_text` to be that value (as a string).

To ensure that the readings are updated automatically, the `repeat` method is called on `reading_text` with a first parameter of the number of milliseconds between updates (in this case, 1000) and a second parameter of the function to call (`update_reading`).

## Discussion

Although this recipe uses a distance sensor, it works equally well with the other sensor recipes in this chapter. You just need to change the method of obtaining a reading from the sensor.

## See Also

For information on formatting numbers to a certain number of decimal places, see [Recipe 7.1](#).

For an example of displaying sensor data in a web browser rather than an application window, see [Recipe 17.2](#).

## 14.24 Logging to a USB Flash Drive

### Problem

You want to log data measured with a sensor onto a USB flash drive.

### Solution

Write a Python program that writes the data to a file on a USB flash drive. By writing the file in comma-separated value (CSV) format, you can import it directly into a spreadsheet, including LibreOffice on the Raspberry Pi (Recipe 4.3).

The example program (*ch\_14\_temp\_log.py*) will log temperature readings recorded from your Raspberry Pi's CPU (see Recipe 14.11):

```
import glob, time, datetime
from gpiozero import CPUtemperature

log_period = 600 # seconds

logging_folder = glob.glob('/media/pi/*')[0]
dt = datetime.datetime.now()
file_name = "temp_log_{:%Y_%m_%d}.csv".format(dt)
logging_file = logging_folder + '/' + file_name

def read_temp():
    cpu_temp = CPUtemperature().temperature
    return cpu_temp

def log_temp():
    temp_c = read_temp()
    dt = datetime.datetime.now()
    f = open(logging_file, 'a')
    line = '\n{:%H:%M:%S}', "{}".format(dt, temp_c)
    f.write(line)
    print(line)
    f.close()

print("Logging to: " + logging_file)
while True:
    log_temp()
    time.sleep(log_period)
```

As with all the program examples in this book, you can download this program (see Recipe 3.22).

The program is set to log the temperature every 10 minutes (600 seconds). You can alter this by changing the value of `log_period`.

You need to run this program using `sudo` to allow access to the flash drive:

```
$ $ sudo python3 ch_14_temp_log.py
Logging to: /media/pi/temp_log_2022_06_22.csv
"13:09:02", "38.459"
"13:09:12", "38.946"
"13:09:22", "37.972"
"13:09:32", "37.485"
```

When logging starts, the path to the logging file on the flash drive is displayed.

Note that to speed things up, the logging period was set to 10 seconds.

## Discussion

When you plug a USB flash drive into a Raspberry Pi, it automatically installs it under */media/pi*. If more than one removable drive is attached to your Raspberry Pi, the program uses the first folder it finds inside */media*. The name of the logging file is constructed from the current date.

If you open the file in a spreadsheet, you will be able to edit it directly. Your spreadsheet might ask you to specify the separator for the data, which will be a comma.

**Figure 14-33** shows a set of data captured using this recipe, and the resulting file has been opened with the LibreOffice spreadsheet (**Recipe 4.3**) running on the Raspberry Pi.



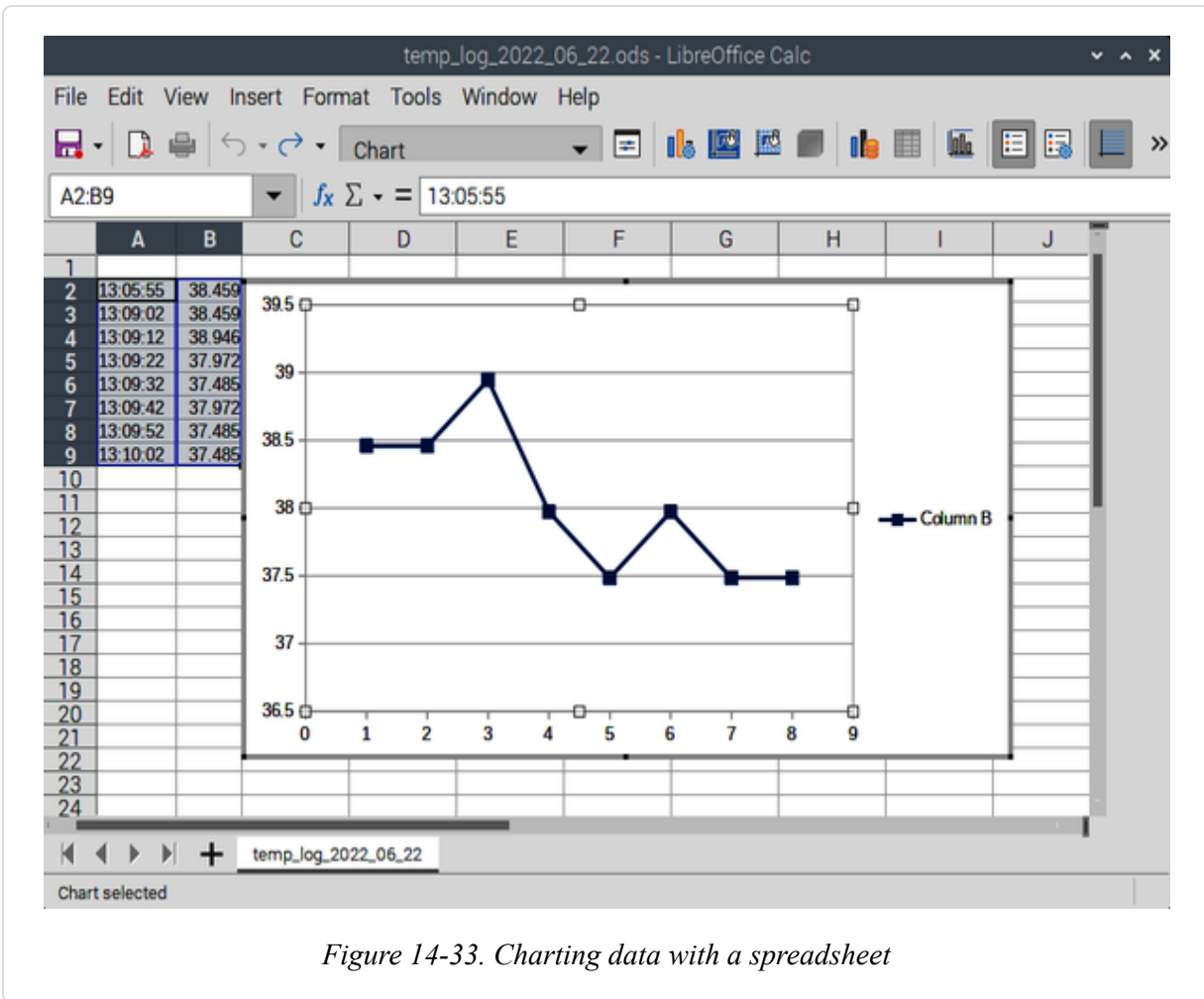


Figure 14-33. Charting data with a spreadsheet

## See Also

This program could easily be adapted for use with any of the other sensors used in this chapter.

For an example of logging sensor data to a web service, see [Recipe 17.7](#).

# Chapter 15. Displays

---

## 15.0 Introduction

Although the Raspberry Pi can use a monitor or TV as a display, it is often nice to use a smaller, more specialized display with it. In this chapter, we explore a range of different displays that can be attached to a Raspberry Pi.

An alternative to attaching special-purpose displays as described in this chapter is to attach a monitor (this can be a small one) to your Raspberry Pi or even use a phone or tablet connected over VNC.

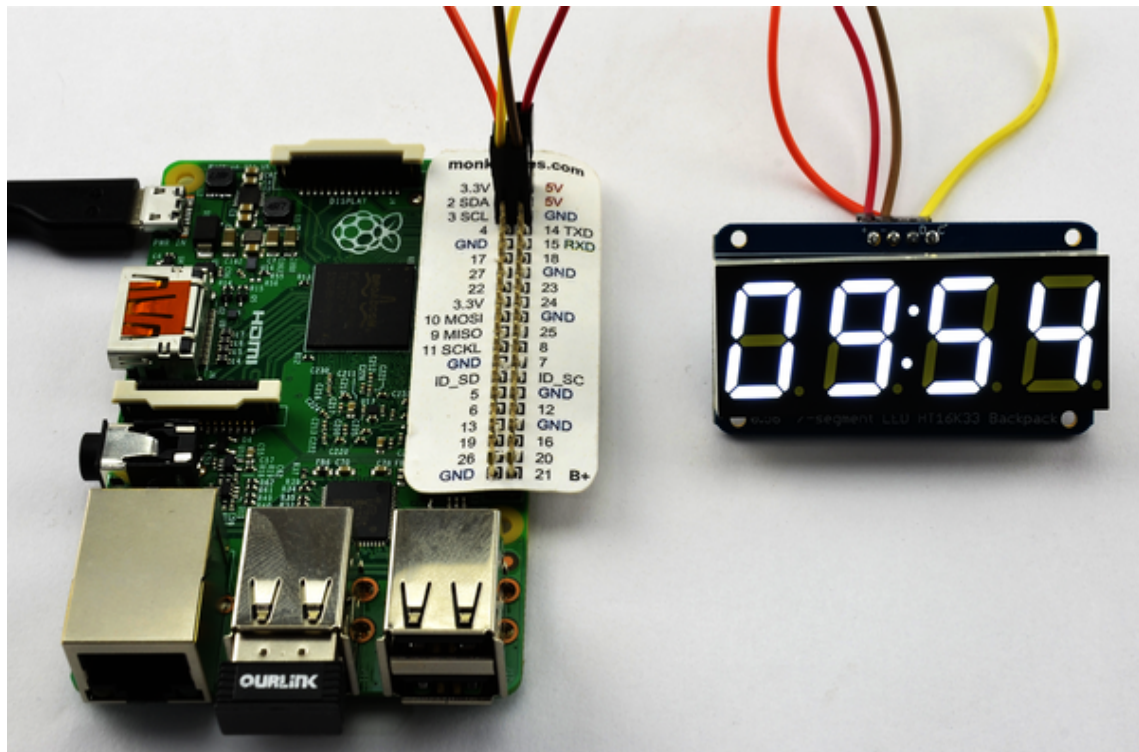
## 15.1 Using a Four-Digit LED Display

### Problem

You want to display a four-digit number in an old-fashioned, seven-segment LED display.

### Solution

Attach an Inter-Integrated Circuit (I2C) LED module, such as the model shown in [Figure 15-1](#), to a Raspberry Pi using female-to-female jumper wires.



*Figure 15-1. Seven-segment LED display with a Raspberry Pi*

To make this recipe, you'll need the following:

- Four female-to-female jumper wires (see “[Prototyping Equipment and Kits](#)”)
- Adafruit 4×7-segment LED with I2C backpack (see “[Modules](#)”)

The connections between the Raspberry Pi and the module are as follows:

- VCC (+) on the display to 5V on the Raspberry Pi general-purpose input/output (GPIO) connector
- GND (–) on the display to GND on the Raspberry Pi GPIO connector
- SDA (D) on the display to GPIO 2 (SDA) on the Raspberry Pi GPIO connector
- SCL (C) on the display to GPIO 3 (SCL) on the Raspberry Pi GPIO connector

Note that Adafruit also supplies a jumbo-size LED display. You can connect this to the Raspberry Pi using the connections in the preceding list, but the larger display has two positive power pins: one for the logic (V<sub>IO</sub>) and one for the display (5V). This is because, being a large display, it requires more current. Fortunately, the Raspberry Pi can supply enough power for it. You can use an extra female-to-female jumper wire to connect this extra pin to the second 5V pin on the GPIO connector.

For this recipe to work, you'll also need to set up your Raspberry Pi for I2C, so follow [Recipe 10.4](#) first.

Enter these commands to install the Adafruit code to support this display:

```
$ cd ~
$ pip3 install adafruit-blinka
$ pip3 install adafruit-circuitpython-ht16k33
$ sudo apt install python3-pil
```

As a simple example, the program *ch\_15\_7\_seg.py* counts up from 0 to 9999 once a second and then restarts at 0:

```
import board
from adafruit_ht16k33.segments import Seg7x4
from time import sleep

i2c = board.I2C()
display = Seg7x4(i2c)
display.brightness = 0.5

x = 0

while True:
    display.print("    ")
    display.print(x)
    x += 1
    if x > 9999:
        x = 0
    sleep(1)
```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

## Discussion

The Adafruit software for this display uses the Blinka module that allows you to run Adafruit's CircuitPython libraries in regular Python.

To set up the display, an I2C instance is created, and then a Seg7x4 instance is created with the I2C interface as a parameter.

The display can show numbers or rudimentary text. It displays text and numbers sequentially, without clearing the previous display, so if you asked it to display 1 and then 2, it would actually display 12, scrolling the old digits to the left before displaying the new digit. So, to clear the display before showing the next digit, you can print four spaces using `display.print(" ")`.

## See Also

Find out more about the [Adafruit CircuitPython library](#).

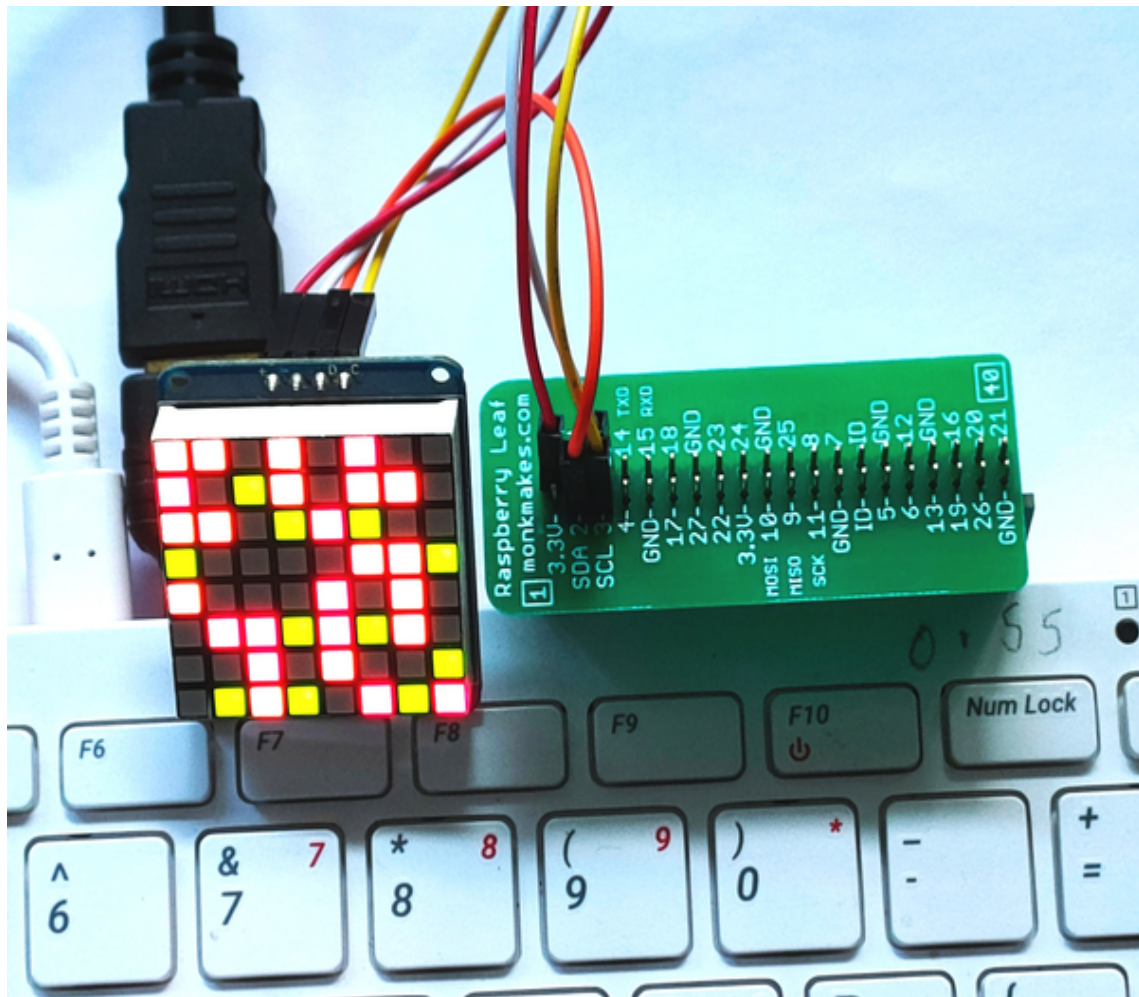
## 15.2 Displaying Graphics on an I2C LED Matrix

### Problem

You want to control the pixels of a multicolor LED matrix display.

### Solution

Use an I2C LED module, such as the model shown in [Figure 15-2](#), attached to a Raspberry Pi using female-to-female jumper wires.



*Figure 15-2. LED matrix display with a Raspberry Pi 400 and GPIO adapter*

To make this recipe, you'll need the following:

- Four female-to-female jumper wires (see [“Prototyping Equipment and Kits”](#))
- Adafruit bicolor LED square-pixel matrix with I2C backpack (see [“Modules”](#))
- If you are using a Pi 400 as shown in [Figure 15-2](#), you will also need a GPIO adapter to be able to access the pins easily.

The connections between the Raspberry Pi and the module are as follows:

- VCC (+) on the display to 5V on the Raspberry Pi GPIO connector

- GND (–) on the display to GND on the Raspberry Pi GPIO connector
- SDA (D) on the display to GPIO 2 (SDA) on the Raspberry Pi GPIO connector
- SCL (C) on the display to GPIO 3 (SCL) on the Raspberry Pi GPIO connector

For this recipe to work, you will also need to set up your Raspberry Pi for I2C, so follow [Recipe 10.4](#) first.

The display uses the same module code as [Recipe 15.1](#). If you have not already installed it, run the following commands:

```
$ cd ~
$ pip3 install adafruit-blinka
$ pip3 install adafruit-circuitpython-ht16k33
$ sudo apt install python3-pil
```

The example program *ch\_15\_matrix.py* sets a randomly selected pixel to a randomly selected color from off: red, green, or orange. It does this 10 times a second:

```
import board
from adafruit_ht16k33.matrix import Matrix8x8x2
from time import sleep
from random import randint

i2c = board.I2C()
display = Matrix8x8x2(i2c)
display.brightness = 0.5

while True:
    x = randint(0, 8)
    y = randint(0, 8)
    color = randint(0, 4)
    display[x, y] = color
    sleep(0.1)
```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

## Discussion

The program imports the `Matrix8x8x2` class, which is then initialized with the I2C port. `x` and `y` coordinates are selected at random, each from a value of 0 to 7. The color is a randomly selected number between 0 and 3: 0 is off, 1 is red, 2 is green, and 3 is orange (both red and green). These are the only colors available on this display, but it still looks pretty!

## See Also

You can find out more about the I2C LED module at <https://oreil.ly/yA49j>.

# 15.3 Using the Sense HAT LED Matrix Display

## Problem

You want to display messages and graphics using the display of a Sense HAT.

## Solution

Follow [Recipe 10.15](#) to install the software that the Sense HAT needs and then use the library commands to display text.

The program `ch_15_sense_hat_clock.py` illustrates this by repeatedly displaying the date and time in a scrolling message:

```
from sense_hat import SenseHat
from datetime import datetime
import time

hat = SenseHat()
time_color = (0, 255, 0) # green
date_color = (255, 0, 0) # red

while True:
    now = datetime.now()
```



```
date_message = '{:%d %B %Y}'.format(now)
time_message = '{:%H:%M:%S}'.format(now)

hat.show_message(date_message, text_colour=date_color)
hat.show_message(time_message, text_colour=time_color)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

## Discussion

Two colors are defined so that the date and time parts of the message can be displayed in different colors. These colors are then used as an optional parameter to `show_message`. Other optional parameters to `show_message` are:

`scroll_speed`

This is actually the delay between each scrolling step rather than the speed. A higher value makes the scrolling slower.

`back_colour`

This sets the background color. Note the British spelling of “colour,” with a “u.”

You can use the display for a lot more than just displaying scrolling text. Starting at its most basic, you can set individual pixels using `set_pixel`, set the orientation of the display using `set_rotation`, and display an image (albeit a small one) with `load_image`. The example that follows, which you can find in `ch_15_sense_hat_taster.py`, illustrates these function calls. As with all the program examples in this book, you can download them (see [Recipe 3.22](#)).

The image must be just 8×8 pixels, but you can use most common graphics formats, such as `.jpg` and `.png`, and the bit depth will be handled automatically:

```
from sense_hat import SenseHat
import time
```

```
hat = SenseHat()

red = (255, 0, 0)

hat.load_image('small_image.png')
time.sleep(1)
hat.set_rotation(90)
time.sleep(1)
hat.set_rotation(180)
time.sleep(1)
hat.set_rotation(270)
time.sleep(1)

hat.clear()
hat.set_rotation(0)
for xy in range(0, 8):
    hat.set_pixel(xy, xy, red)
    hat.set_pixel(xy, 7-xy, red)
```

Figure 15-3 shows the Sense HAT displaying a crude image.



Figure 15-3. A Sense HAT displaying an “image”

## See Also

More information is available in the [full documentation on the Sense HAT](#).

For information on formatting dates and times, see [Recipe 7.2](#).

Other recipes that use the Sense HAT are Recipes [10.15](#), [14.12](#), [14.15](#), [14.16](#), and [14.18](#).

## 15.4 Using an OLED Graphical Display

### Problem

You want to attach a graphical OLED (organic LED) display to your Raspberry Pi.

### Solution

Use an OLED display based on the SSD1306 driver chip, using an I2C interface ([Figure 15-4](#)).

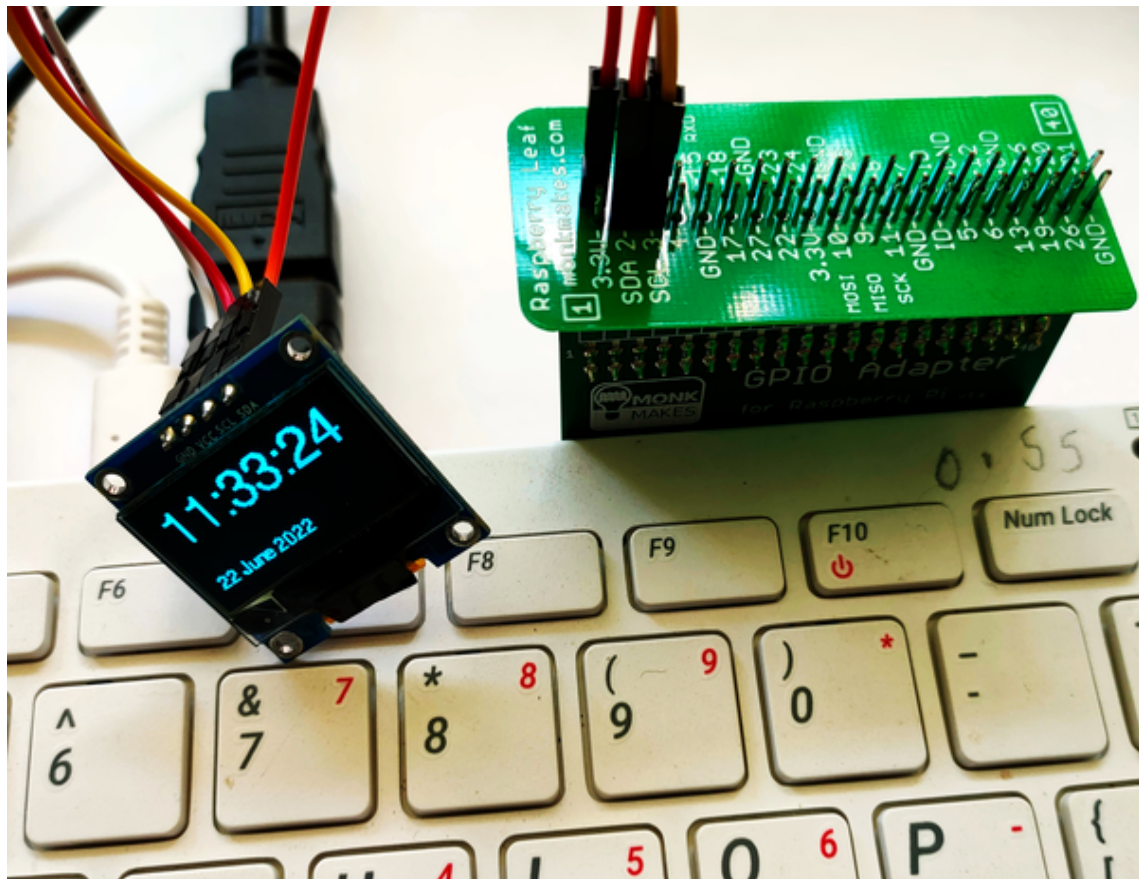


Figure 15-4. An I2C OLED display on a Raspberry Pi 400

To make this recipe, you will need the following:

- Four female-to-female jumper wires (see “[Prototyping Equipment and Kits](#)”)
- I2C OLED display 128×64 pixels (see “[Modules](#)”)

The connections between the Raspberry Pi and the module are as follows:

- VCC on the display to 5V on the Raspberry Pi GPIO connector
- GND on the display to GND on the Raspberry Pi GPIO connector
- SDA on the display to GPIO 2 (SDA) on the Raspberry Pi GPIO connector
- SCL on the display to GPIO 3 (SCL) on the Raspberry Pi GPIO connector

For this recipe to work, you will also need to set up your Raspberry Pi for I2C, so follow [Recipe 10.4](#) first.

Adafruit has a library for these displays, which you can install using these commands:

```
$ cd ~
$ pip3 install adafruit-blinka
$ pip3 install adafruit-circuitpython-ssd1306
```

This library uses the Python Image Library (PIL) and NumPy modules, which you can install using the following command:

```
$ sudo apt install python3-pil
$ sudo apt install python3-numpy
```

The code example *ch\_15\_oled\_clock.py* displays the time and date on the OLED display:

```
import board
from PIL import Image, ImageDraw, ImageFont
import adafruit_ssd1306
from time import sleep
from datetime import datetime

# Set up display
i2c = board.I2C()
disp = adafruit_ssd1306.SSD1306_I2C(128, 64, i2c, addr=0x3C)
small_font = ImageFont.truetype('FreeSans.ttf', 12)
large_font = ImageFont.truetype('FreeSans.ttf', 33)
disp.fill(0)
disp.show()

# Make an image to draw on in 1-bit color.
width = disp.width
height = disp.height
image = Image.new('1', (width, height))
draw = ImageDraw.Draw(image)

# Display a message on 3 lines, first line big font
def display_message(top_line, line_2):
    draw.rectangle((0,0,width,height), outline=0, fill=0)
```

```
draw.text((0, 0), top_line, font=large_font, fill=255)
draw.text((0, 50), line_2, font=small_font, fill=255)
disp.image(image)
disp.show()
```

```
while True:
    now = datetime.now()
    date_message = '{:%d %B %Y}'.format(now)
    time_message = '{:%H:%M:%S}'.format(now)
    display_message(time_message, date_message)
    sleep(0.1)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

## Discussion

Every I2C slave device has an address, which is set in the line:

```
disp = adafruit_ssd1306.SSD1306_I2C(128, 64, i2c, addr=0x3C)
```

This is fixed at 3C (hex) for many of the low-cost I2C modules, but may vary, so you should check any documentation that comes with the device or use I2C tools ([Recipe 10.5](#)) to list all the I2C devices attached to the bus so that you can see the address of your display.

The preceding code example uses a technique called *double buffering*. This involves preparing what is to be displayed and then switching it onto the image in one go. This prevents the display from flickering.

You can see the code for this in the `display_message` function. First it draws onto the image a blank rectangle the entire size of the display. It then draws the text onto the image, and then sets the display content to be image using `disp.image(image)`. The display isn't actually updated until the function `disp.show()` is called.

Small OLED displays are cheap, don't use much current, and have high resolution despite their diminutive size. They are replacing LCD displays in many consumer products.

## See Also

The instructions here are for four-pin I2C interfaces. If you really want to use the SPI interface, take a look at [Adafruit's tutorial](#).

# 15.5 Using Addressable RGB LED Strips

## Problem

You want to connect an RGB LED strip (NeoPixels) to your Raspberry Pi.

## Solution

Use an LED strip based on the WS2812 RGB LED chips on your Raspberry Pi.

Using these LED strips ([Figure 15-5](#)) can be really easy, with a direct connection to the Raspberry Pi and power for the LEDs supplied by the Raspberry Pi's 5V supply. This is the “Happy Day” scenario that should work just fine, but do see the Discussion for providing external power so that your LED strip use is trouble-free.

### **Don't Power the LEDs from the Pi's 3.3V Supply**

Although it might be tempting to power the LEDs from the 3.3V supply pin on the GPIO connector, do not do this—it can supply only low currents (see [Recipe 10.3](#)). Using this pin could easily damage your Raspberry Pi.

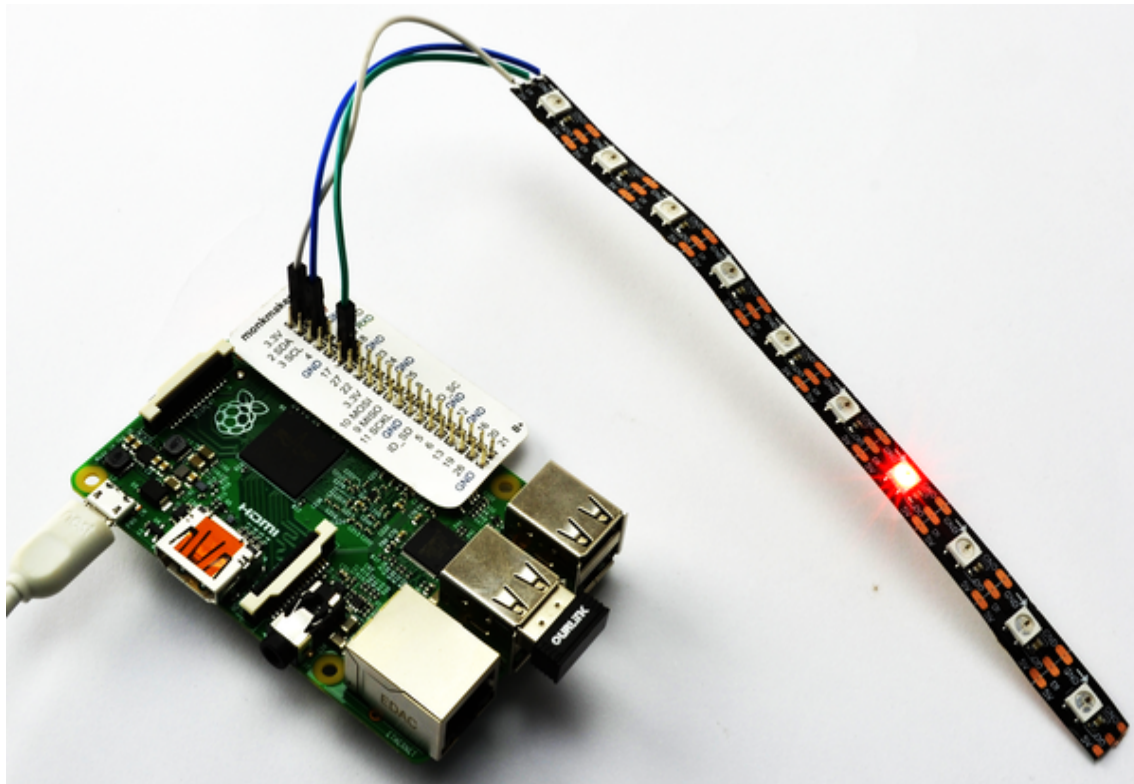


Figure 15-5. An LED strip of 10 LEDs

The LED strip used in [Figure 15-5](#) is cut from a reel. In this case, there are 10 LEDs. Because each LED can use up to 60mA, 10 is probably a sensible limit for the number of LEDs that can be used without arranging for a separate power supply for the LED strip (see the Discussion section).

To connect the strip to the Raspberry Pi, jumper wires with female connectors on one end were cut and the wire ends soldered to the three connections on the LED strip (see [Recipe 10.3](#)): GND, DI (data in), and 5V. These can then be attached to GPIO pins GND, GPIO 18, and 5V, respectively.

Notice that the LED strip has right-facing arrows printed on it ([Figure 15-6](#)). Make sure that when you solder leads to the LED strip, you start from the cut end to the *left* of the arrows.



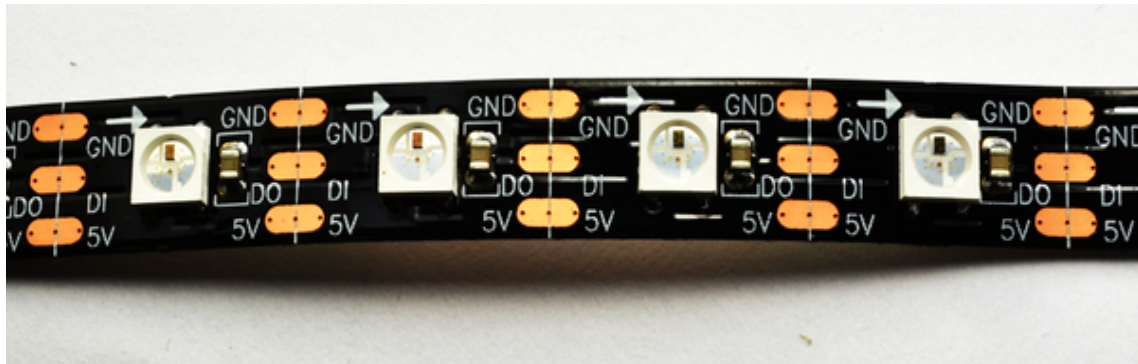


Figure 15-6. An LED strip close-up

We are going to use the Adafruit software to control our addressable LEDs. To install it, run the following commands:

```
$ pip3 install adafruit-blinka
$ sudo pip3 install rpi_ws281x adafruit-circuitpython-neopixel
```

The following example program (*ch\_15\_neopixel.py*) will set the LED red in successive positions along the strip:

```
import time
import board
from neopixel import NeoPixel

led_count = 5
red = (100, 0, 0)
no_color = (0, 0, 0)

strip = NeoPixel(board.D18, led_count, auto_write=False)

def clear():
    for i in range(0, led_count):
        strip[i] = no_color
    strip.show()

i = 0
while True:
    clear()
    strip[i] = red
    strip.show()
    time.sleep(1)
```

```
i += 1
if i >= led_count:
    i = 0
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

Run the program with Python 3 and use the `sudo` command like this:

```
$ sudo python3 ch_15_neopixel.py
```

If you have a different number of LEDs in your strip, modify `led_count`. The rest of the constants do not need to be changed.

You can use only GPIOs 10, 12, 18, and 21 with NeoPixels. If you want to use a different GPIO pin, change `board.D18` to whichever pin you are using.

You can set the color of each of the LEDs independently. The colors are set as a tuple of red, green, and blue intensities. To change the color of an LED at a particular position, use the `strip` variable as if it were an array and set that element to the color you want. For example (`strip[i] = red`).

The changes to the LED colors are not actually updated until the method `show` is called.

## Discussion

Each LED in the strip can use a maximum of about 60mA. They will do this only if all three color channels (red, green, and blue) are at maximum brightness (255). If you plan to use a lot of LEDs, you'll need to use a separate 5V power supply sufficient to supply all the LEDs in your strip.

[Figure 15-7](#) shows how you would wire up a separate power supply. A female direct current (DC) jack-to-screw terminal adapter (see [“Prototyping Equipment and Kits”](#)) makes it easy to connect an external power supply to a breadboard.

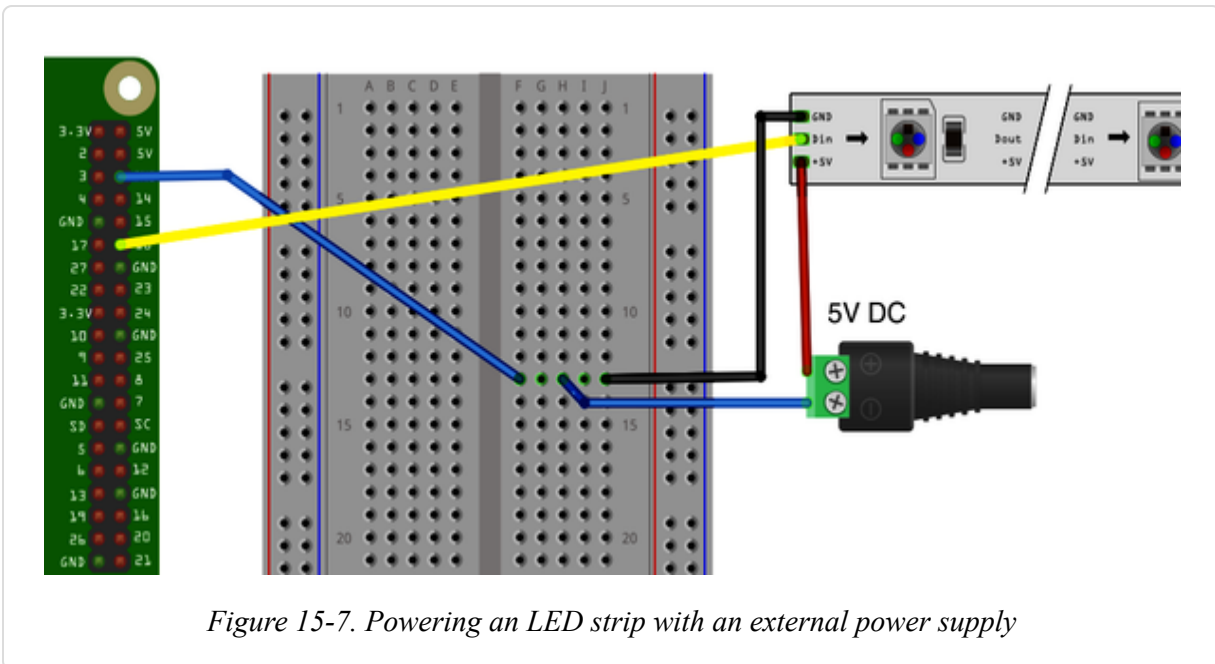


Figure 15-7. Powering an LED strip with an external power supply

## See Also

NeoPixels are also available in [ring format](#).

You can read more about the Adafruit approach to NeoPixels at <https://oreil.ly/-oA90>.

## 15.6 Using the Pimoroni Unicorn HAT

### Problem

You want an RGB LED matrix display for your Raspberry Pi.

### Solution

Use a Pimoroni Unicorn HAT to provide an 8x8 LED matrix ([Figure 15-8](#)).

Start by installing the Unicorn HAT software from Pimoroni:

```
$ curl https://get.pimoroni.com/unicornhat | bash
```

You will be asked several times to confirm various bits of software to install and finally to reboot your Raspberry Pi.

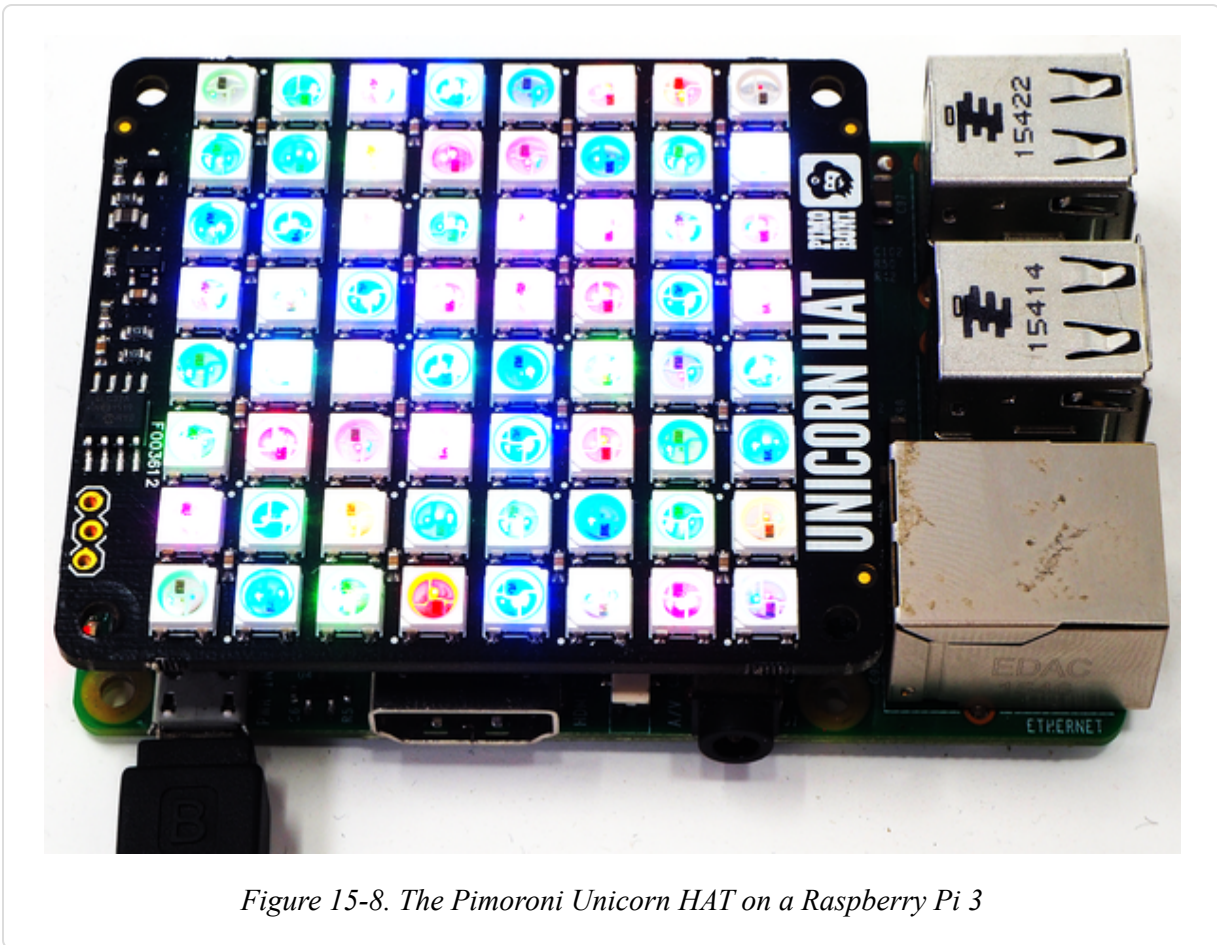


Figure 15-8. The Pimoroni Unicorn HAT on a Raspberry Pi 3

When you've completed the installation, here's a colorful program (`ch_15_unicorn.py`) to run on it. It will repeatedly set the color of a random pixel to a random color. Run the program using the `sudo` command:

```
import time
import unicornhat as unicorn
from random import randint

unicorn.set_layout(unicorn.AUTO)
unicorn.rotation(0)
unicorn.brightness(1)
width, height = unicorn.get_shape()

while True:
    x = randint(0, width)
```

```
y = randint(0, height)
r, g, b = (randint(0, 255), randint(0, 255), randint(0, 255))
unicorn.set_pixel(x, y, r, g, b)
unicorn.show()
time.sleep(0.01)

time.sleep(1)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

## Discussion

The Unicorn HAT makes a very convenient way to attach a matrix of addressable LEDs. You will also find other chains of addressable LEDs laid out in various configurations, including matrices with even more LEDs on them. Generally, such matrices of addressable LEDs are actually arranged electrically as a long chain of LEDs.

## See Also

For more information on addressable (NeoPixel) LEDs, see [Recipe 15.5](#).

# 15.7 Using an ePaper Display

## Problem

You want to use your Raspberry Pi to control an ePaper display.

## Solution

Attach an Inky pHAT or Inky wHAT module to your Raspberry Pi ([Figure 15-9](#)). Download the Pimoroni software for this using the command:

```
$ curl https://get.pimoroni.com/inkyphat | bash
```

If (when prompted) you accept the option to fetch examples and documentation, this will take quite some time to install.

As an example program (*ch\_15\_phat.py*), let's have the Inky pHAT display the Raspberry Pi's IP address:

```
from inky import InkyPHAT
from PIL import Image, ImageFont, ImageDraw
from font_fredoka_one import FredokaOne
import subprocess

inky_display = InkyPHAT("red")
inky_display.set_border(inky_display.WHITE)

img = Image.new("P", (inky_display.WIDTH, inky_display.HEIGHT))
draw = ImageDraw.Draw(img)
font = ImageFont.truetype(FredokaOne, 22)

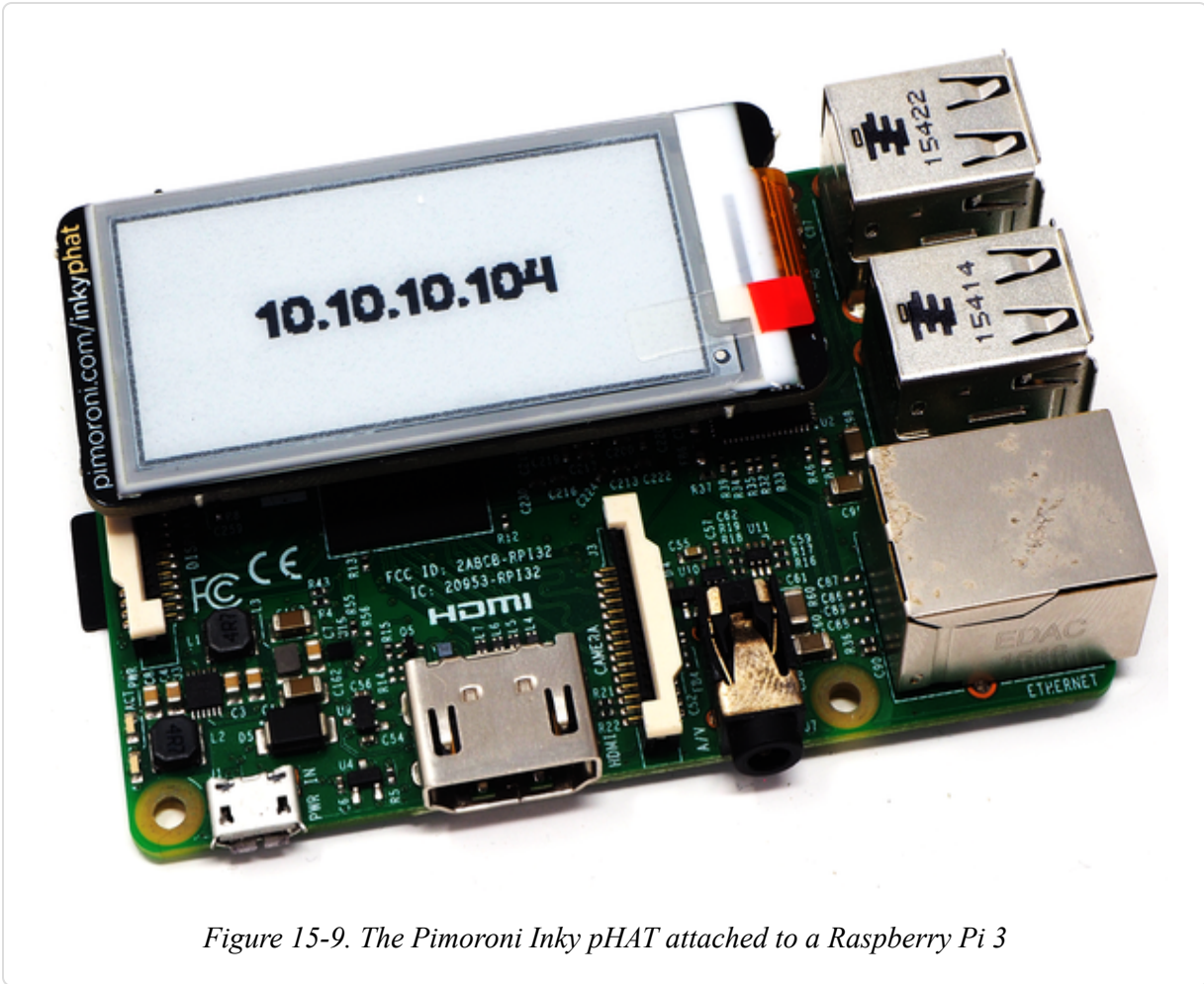
message = str(subprocess.check_output(['hostname', '-I'])).split()[0][2:]
print(message)

w, h = font.getsize(message)
x = (inky_display.WIDTH / 2) - (w / 2)
y = (inky_display.HEIGHT / 2) - (h / 2)

draw.text((x, y), message, inky_display.RED, font)
inky_display.set_image(img)
inky_display.show()
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

The example program uses the subprocess library ([Recipe 7.15](#)) to obtain the IP address assigned to the Raspberry Pi.



*Figure 15-9. The Pimoroni Inky pHAT attached to a Raspberry Pi 3*

## Discussion

You won't be playing any video games on these displays because they take a few seconds to update but, after they update, the ePaper keeps whatever you have drawn on it even when the power is removed. Pimoroni sells small displays like the Inky pHAT used here, and also a display twice the size that is pretty much as big as the Raspberry Pi (Inky wHAT).

## See Also

More information is available in the [full documentation of Inky pHAT](#).

# Chapter 16. Sound

---

## 16.0 Introduction

In this chapter, you learn how to use sound with your Raspberry Pi. There are recipes both for playing sounds in various ways—using loudspeakers or a buzzer—and for using a microphone to record sounds.

## 16.1 Connecting a Loudspeaker

### Problem

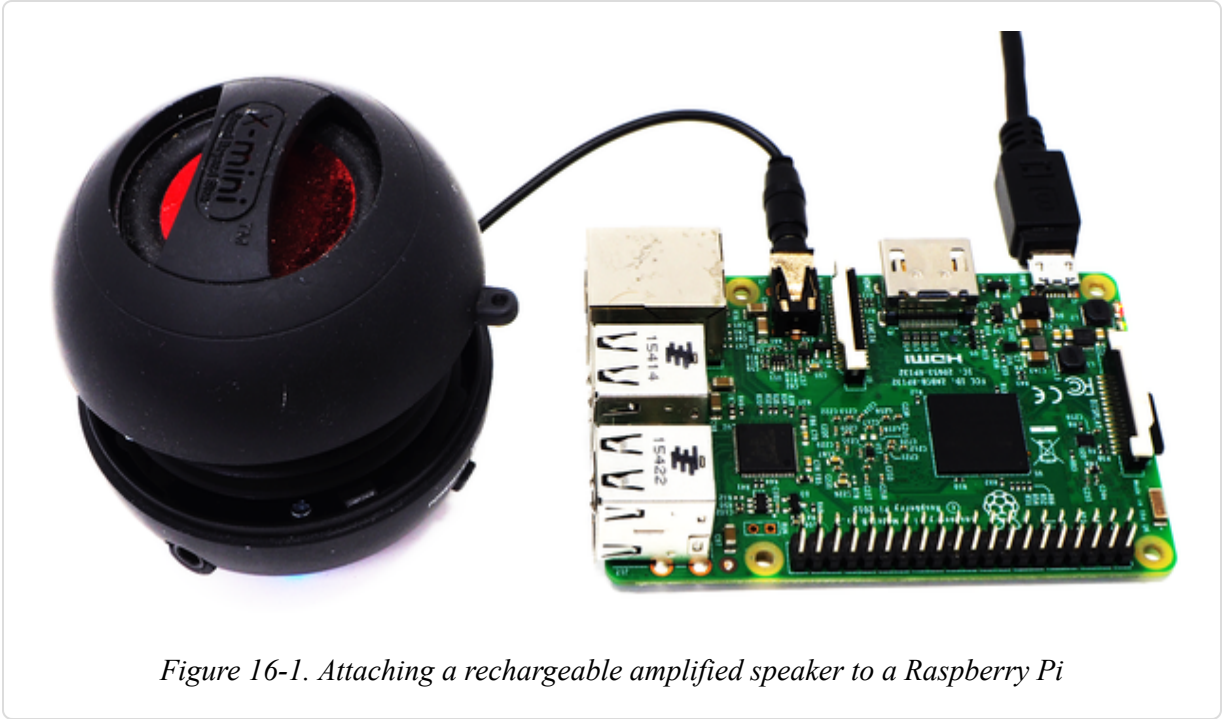
You want to play sounds from your Raspberry Pi.

### Solution

Attach a powered speaker such as the one shown in [Figure 16-1](#).

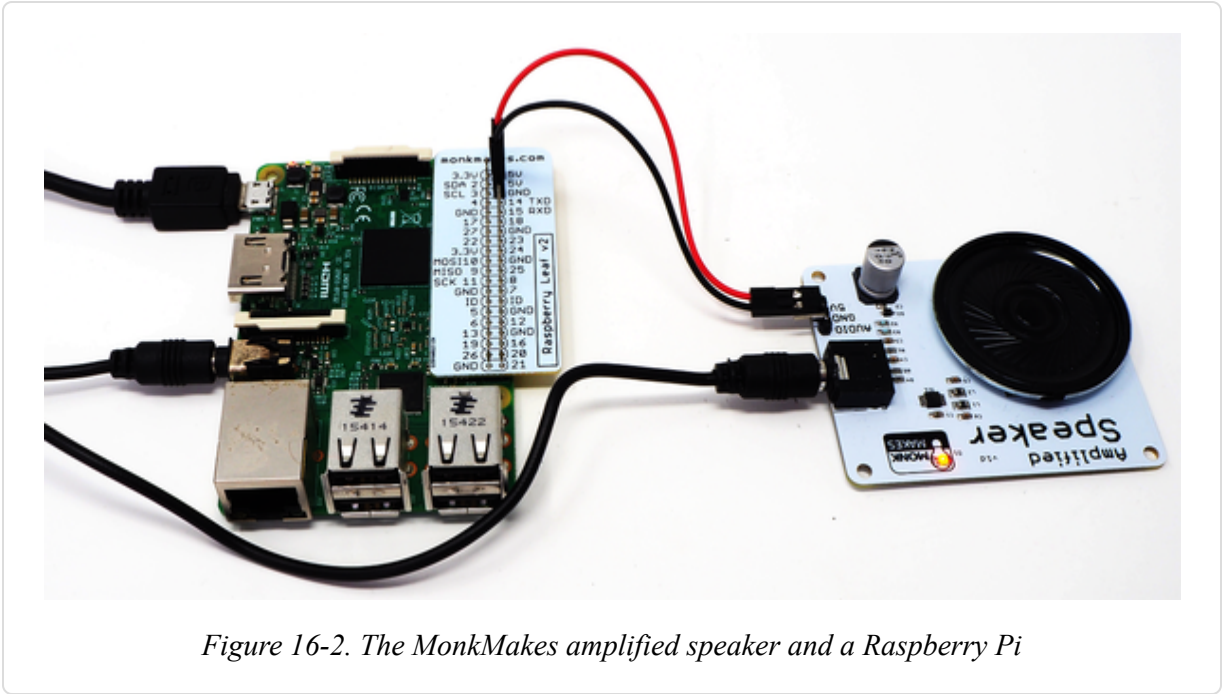
Having attached the speaker to the audiovisual socket, you will also need to configure the Raspberry Pi to play through the audiovisual jack, not HDMI, using [Recipe 16.2](#).





*Figure 16-1. Attaching a rechargeable amplified speaker to a Raspberry Pi*

An alternative to using a general-purpose speaker like the one shown in [Figure 16-1](#) is to use a speaker kit designed specifically for the Raspberry Pi, such as the MonkMakes Speaker Kit for Raspberry Pi, shown in [Figure 16-2](#).



*Figure 16-2. The MonkMakes amplified speaker and a Raspberry Pi*

This speaker is connected using the audio lead supplied with the kit, and the speaker uses the Raspberry Pi's 5V power supply, connected using female-to-female leads.

Raspberry Pi OS comes with a handy program to test that your speaker is working. Type the following command in the Terminal:

```
$ speaker-test -t wav -c 2

speaker-test 1.1.3

Playback device is default
Stream parameters are 48000Hz, S16_LE, 2 channels
WAV file(s)
Rate set to 48000Hz (requested 48000Hz)
Buffer size range from 256 to 32768
Period size range from 256 to 32768
Using max buffer size 32768
Periods = 4
was set period_size = 8192
was set buffer_size = 32768
 0 - Front Left
 1 - Front Right
Time per period = 2.411811
 0 - Front Left
 1 - Front Right
```

When you run this command, you will hear a voice saying “front left,” “front right,” and so on. This is designed to test a stereo audio setup.

## Discussion

As well as connecting amplified loudspeakers of various types, you can also directly connect a pair of headphones to a Raspberry Pi's audio socket.

By default, a Raspberry Pi's output is via the HDMI cable. So if your Pi is connected to a TV or a monitor with built-in speakers, all you need to do to hear sounds from your Raspberry Pi is to turn up the volume on your TV or monitor. However, if you are using the Raspberry Pi without a monitor, or your monitor doesn't have speakers, you can follow this recipe.

## See Also

To pair your Raspberry Pi with a Bluetooth speaker, see [Recipe 1.17](#).

To specify where sound is output, see [Recipe 16.2](#).

To play a test sound, see [Recipe 16.5](#).

For more information on the Speaker Kit for Raspberry Pi, see <https://oreil.ly/Q73vu>.

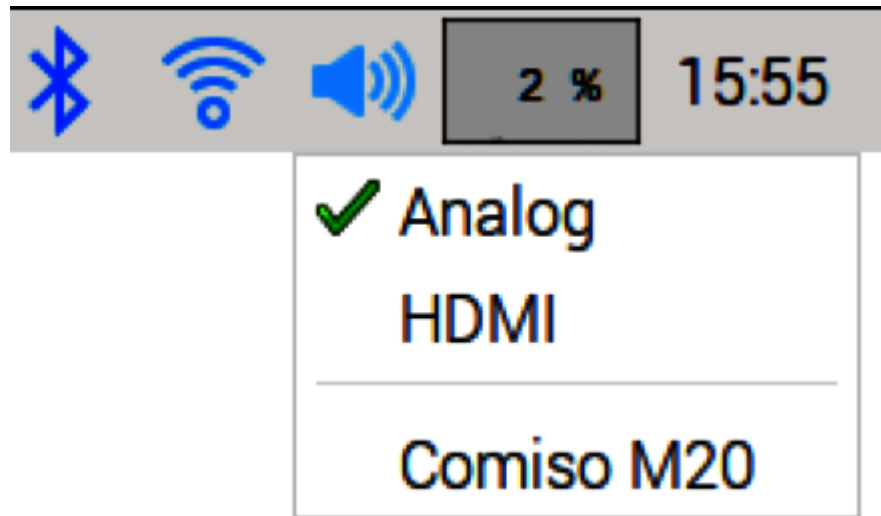
## 16.2 Controlling Where Sound Is Output

### Problem

You want to control which of several output options the Raspberry Pi will use when making sounds.

### Solution

The simplest way to set the audio output is to use the selector available when you right-click the Speaker icon in the upper-right corner of your desktop ([Figure 16-3](#)).



*Figure 16-3. Changing the audio output device*

Note that the entry Comiso M20 in [Figure 16-3](#) is for a Bluetooth speaker that has already been paired with the Raspberry Pi (see [Recipe 1.17](#)).

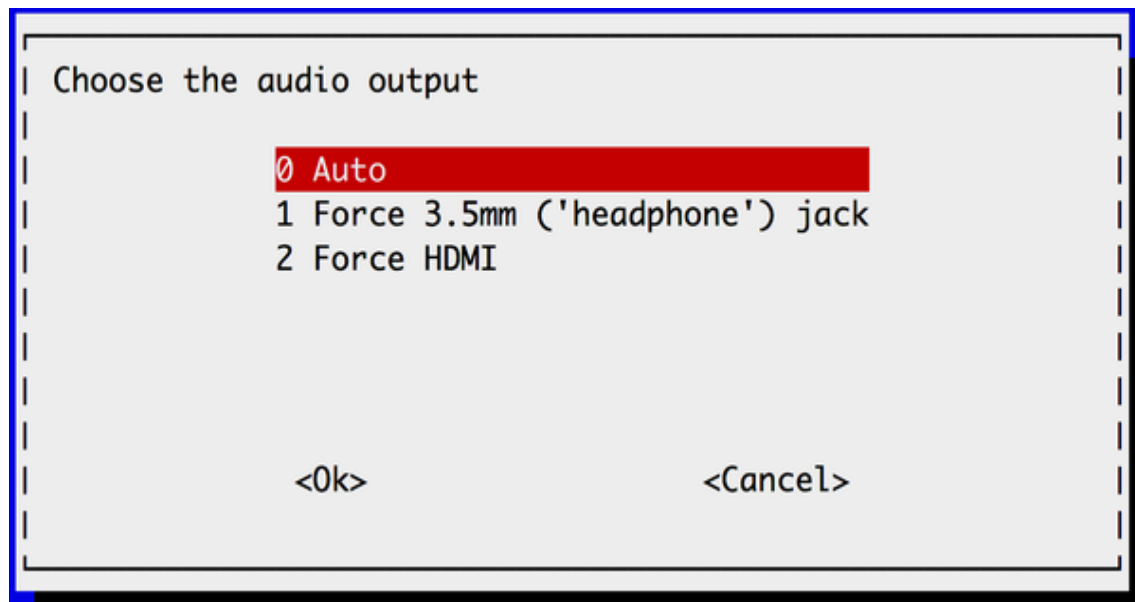
## Discussion

If you are running your Raspberry Pi headless (without a monitor), you can still control where the sound is output using the `raspi-config` command-line utility.

Open a Terminal session and enter the following command:

```
$ sudo raspi-config
```

Then select the option System, followed by Audio. You can then select the option you want, as shown in [Figure 16-4](#).



*Figure 16-4. Using `raspi-config` to change the audio output device*

## See Also

To learn about connecting a speaker to your Raspberry Pi, see [Recipe 16.1](#). For more sound input and output options, you can use the Audio Device Settings tool; see [Recipe 16.6](#).

## 16.3 Playing Audio on a Raspberry Pi Without an Audio Socket

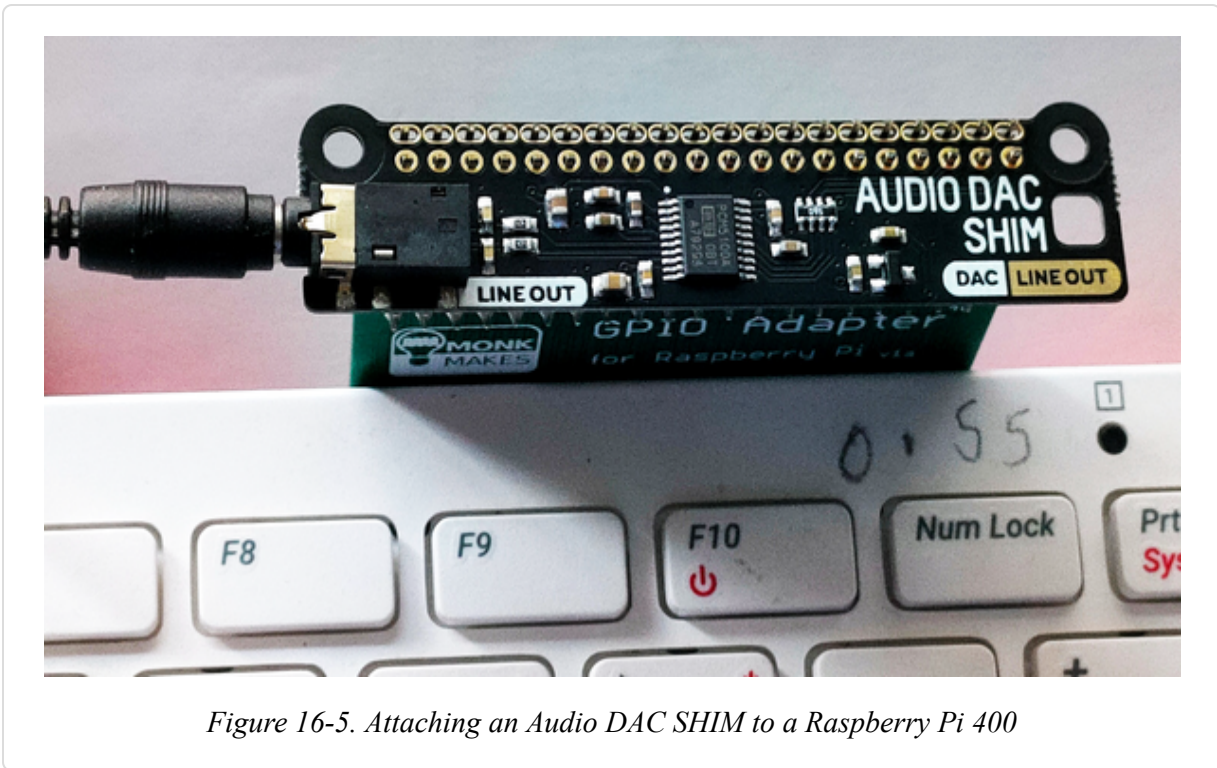
### Problem

You want high-quality audio output on a Raspberry Pi without an audio jack, like a Pi Zero or Pi 400.

### Solution

Attach a Pimoroni Audio DAC Shim as shown in [Figure 16-5](#) to the GPIO connector and connect the line out to an audio amplifier. If you are using a

Raspberry Pi 400, you'll need a GPIO adapter to gain access to the GPIO pins.

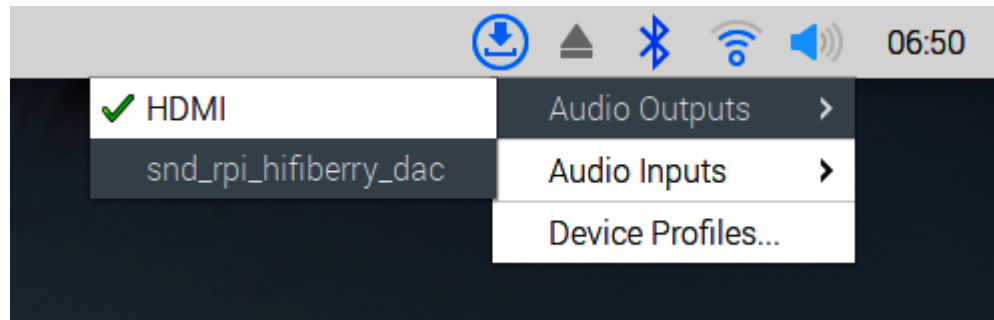


Pimoroni's SHIM concept is a clever one, which allows small add-on boards to be attached to the GPIO connector without the use of soldering or a connector socket. It just slides over the GPIO connector as a tight fit, slightly sprung to make good contact with the pins.

For the Audio DAC card to be recognized, you need to install some software from Pimoroni, by running the following commands:

```
$ cd ~  
$ git clone https://github.com/pimoroni/pirate-audio  
$ cd pirate-audio/mopidy  
$ sudo ./install.sh
```

Once the install has finished, restart your Raspberry Pi, and you should find that when you right-click on the Audio icon on the top right of the desktop, the new sound interface will be available to select (Figure 16-6).



*Figure 16-6. Selecting the Audio DAC SHIM for output*

## Discussion

The output from the Audio DAC SHIM may be the same connector as a headphone jack, but you should not connect headphones directly. This connector is just for connecting to your HiFi and is not powerful enough to drive headphones directly.

A more conventional way to add a new audio interface to your Raspberry Pi is to attach a USB sound card. Most of these that claim to work with Linux will work fine on a Raspberry Pi.

## See Also

As well as configuring the Pi for the Audio DAC, the installation script in this recipe also installs [Mopidy](#), a version of MPD (Music Player Daemon).

## 16.4 Playing Sound from the Command Line

### Problem

You want to be able to play a sound file from the Raspberry Pi's command line.

### Solution

Use the built-in VLC software from the command line. To try this out, locate the file called *school\_bell.mp3* in this book's downloads in the *python* directory. You can play this license-free sound using the following command. Note that on the command line, the command `cvlc`, not `vlc`, is used:

```
$ cvlc ~/raspberrypi_cookbook_ed4/python/school_bell.mp3
```

This will play the sound file through the current audio output device.

## Discussion

VLC will play most types of sound files, including MP3, WAV, AIFF, AAC, and OGG. However, if you want to play only uncompressed WAV files, you can also use the lighter-weight `aplay` command:

```
$ aplay ~/raspberrypi_cookbook_ed4/python/school_bell.mp3
```

## See Also

VLC is also used in [Recipe 4.9](#).

Documentation for `aplay` is available at <https://oreil.ly/Fbs94>.

# 16.5 Playing Sound from Python

## Problem

You want to play a sound file from your Python program.

## Solution

If you need to play the sound from a Python program, you can use the Python `subprocess` module (*ch\_16\_play\_sound.py*), as shown here:



```
import subprocess

sound_file = 'school_bell.mp3'

subprocess.run(['cvlc', sound_file])
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

## Discussion

You can also play sounds using the `pygame` library (*ch\_16\_play\_sound\_pygame.py*), as illustrated here:

```
import pygame

sound_file =
    '/home/pi/raspberrypi_cookbook_ed4/python/school_bell.wav'

pygame.mixer.init()
pygame.mixer.music.load(sound_file)
pygame.mixer.music.play()

while pygame.mixer.music.get_busy() == True:
    continue
```

The sound file must be either an OGG file or an uncompressed WAV file. I found that the sound would play only through the HDMI channel.

## See Also

To play sounds directly from the command line, see [Recipe 16.4](#).

You also can use `pygame` for intercepting keypresses ([Recipe 13.11](#)) and mouse movements ([Recipe 13.12](#)).

## 16.6 Using a USB Microphone

## Problem

You want to connect a microphone to your Raspberry Pi to capture sound.

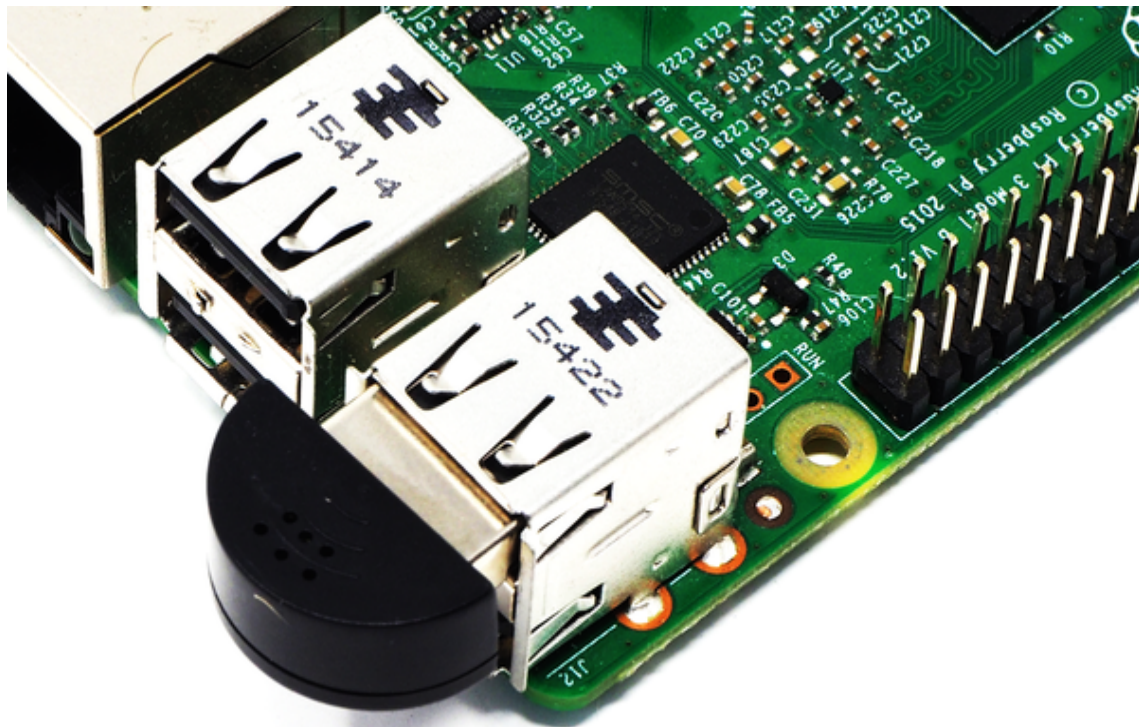
## Solution

Use a USB microphone like the one shown in [Figure 16-7](#), or the microphone of a USB webcam, or something a bit more substantial and of better quality.

These devices come in various shapes and sizes. Some fit directly into the USB socket, others have a USB lead attached to them, and some are part of a headset.

Having plugged in your USB microphone, check that Raspberry Pi OS is aware of it by running this command, which lists available devices from which you can record:

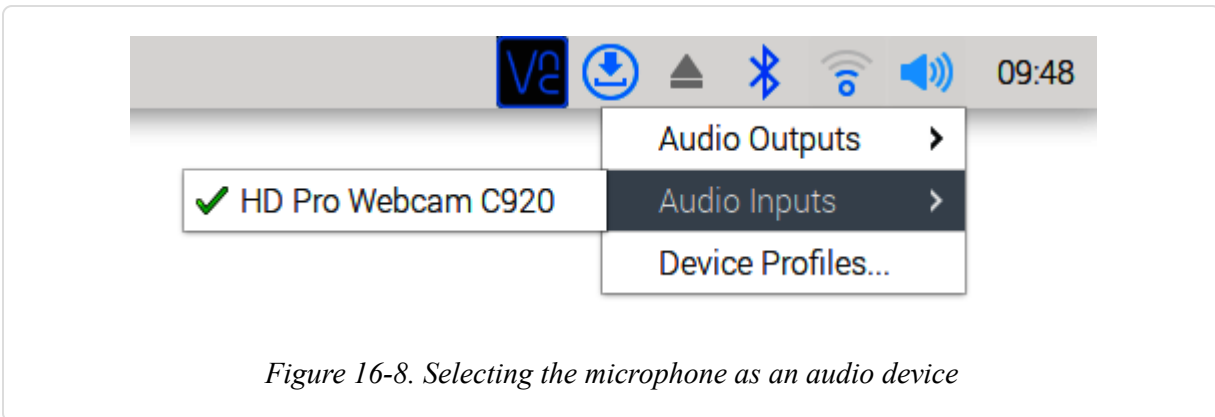
```
$ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: H340 [Logitech USB Headset H340], device 0: USB Audio
[USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```



*Figure 16-7. Attaching a USB microphone to a Raspberry Pi*

In this case, I'm using a microphone that's part of a Logitech USB headset with microphone. As the previously described output shows, this is card 1, subdevice 0. We'll need this information when we come to recording sounds.

To make the microphone active, you need to right-click on the speaker icon in the top right of the screen. Then select your USB mic device from the Audio Inputs list, as shown in [Figure 16-8](#). Here we are selecting the microphone of a USB webcam.



You can now try to make a recording from the command line and then play it back. To record, use the following command:

```
$ arecord -d 3 test.wav
```

The `-d` parameter specifies the duration of the recording in seconds.

To play back the sound you have just recorded, use the following:

```
$ aplay test.wav
Playing WAVE 'test.wav' : Unsigned 8 bit, Rate 8000 Hz, Mono
```

You can interrupt the sound playing by pressing Ctrl-C.

## Discussion

I have had mixed results with the tiny push-in microphones like the one shown in [Figure 16-7](#). Some work and some don't. So, if you have followed the instructions here and you still can't get your microphone to record anything, try a different one.

The `arecord` command has optional parameters that enable you to control the sample rate and format of the audio you record. So, for example, if you want to record at 16,000 samples a second rather than the default of 8,000, use the following command:

```
$ arecord -r 16000 -d 3 test.wav
```

When you play back the sound, `aplay` will automatically detect the sample rate, so you can just do this:

```
$ aplay test.wav  
Playing WAVE 'test.wav' : Unsigned 8 bit, Rate 16000 Hz, Mono
```

## See Also

For ways of attaching a speaker to your Raspberry Pi, see [Recipe 16.1](#).

For other methods of playing a sound file, see [Recipes 16.4](#) and [16.5](#).

## 16.7 Making a Buzzing Sound

### Problem

You want to make a buzzing sound with the Raspberry Pi.

### Solution

Use a piezoelectric buzzer connected to a GPIO pin.

Most small piezo buzzers work just fine using the arrangement shown in [Figure 16-9](#). The one I used is an Adafruit-supplied component (see [“Miscellaneous”](#)). You can connect the buzzer pins directly to the Raspberry Pi using female-to-female headers (see [“Prototyping Equipment and Kits”](#)).

These buzzers use very little current. However, if you have a large buzzer or just want to play it safe, put a  $470\Omega$  resistor between the GPIO pin and the buzzer lead.

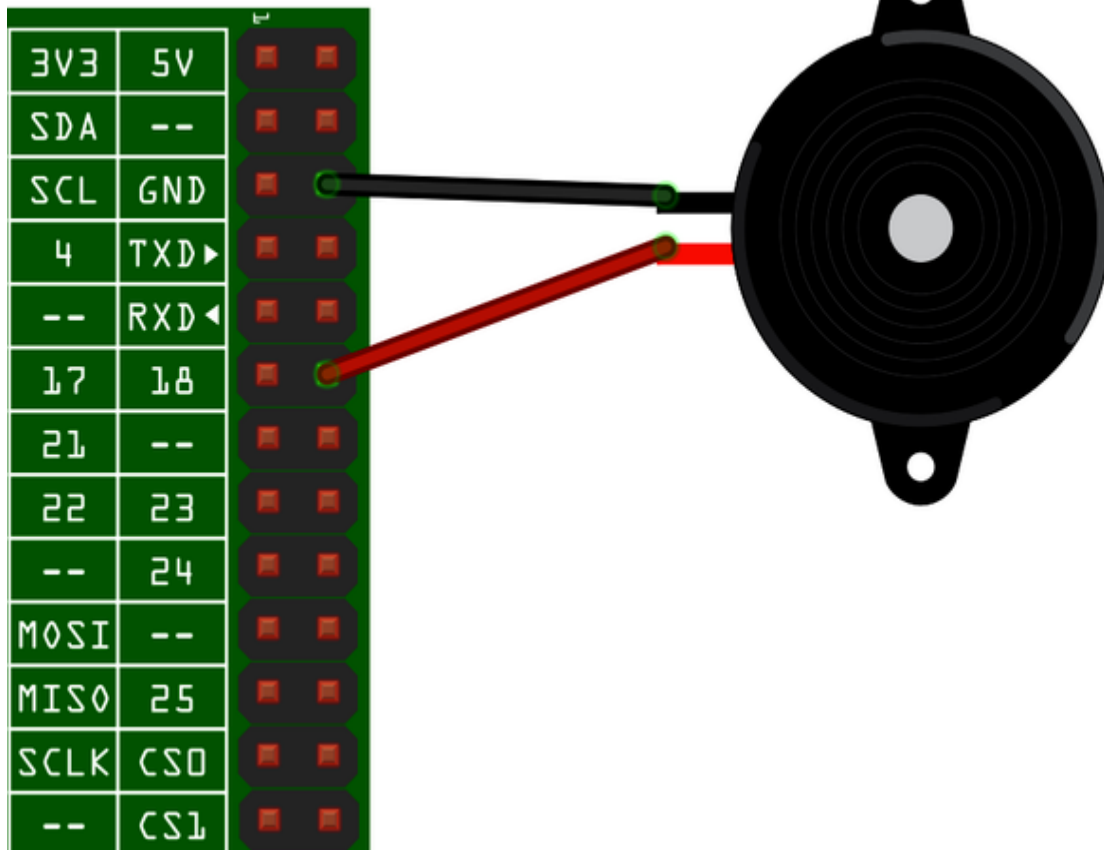


Figure 16-9. Connecting a piezo buzzer to a Raspberry Pi

Paste the following code into an editor (*ch\_16\_buzzer.py*):

```
from gpiozero import Buzzer

buzzer = Buzzer(18)

def buzz(pitch, duration):
    period = 1.0 / pitch
    delay = period / 2
    cycles = int(duration * pitch)
    buzzer.beep(on_time=period, off_time=period, n=int(cycles/2))

while True:
    pitch_s = input("Enter Pitch (200 to 2000): ")
    pitch = float(pitch_s)
    duration_s = input("Enter Duration (seconds): ")
```

```
duration = float(duration_s)
buzz(pitch, duration)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

When you run the program, it first prompts you for the pitch in Hz and then for the duration of the buzz in seconds:

```
$ python3 ch_16_buzzer.py
Enter Pitch (200 to 10000): 2000
Enter Duration (seconds): 20
```

## Discussion

Piezo buzzers don't have a wide range of frequencies, nor is the sound quality remotely good. However, you can vary the pitch a little. The frequency generated by the code is not very accurate, and you might hear a bit of warbling.

The program works by using the `gpiozero Buzzer` class to toggle GPIO pin 18 on and off, with a short delay in between. The `delay` is calculated from the `pitch`. The higher the `pitch` (frequency), the shorter the delay needs to be.

## See Also

Take a look at the [datasheet for the piezo buzzer](#).

For better audio output options, see [Recipe 16.1](#).

# Chapter 17. The Internet of Things

---

## 17.0 Introduction

The *Internet of Things* (IoT) is the rapidly growing network of devices (things) connected to the internet. That doesn't just mean more and more computers using browsers, but actual appliances and wearable and portable technology. This includes all sorts of home automation, from smart appliances and lighting to security systems and even internet-operated pet feeders, as well as lots of less practical but fun projects.

In this chapter, you learn how your Raspberry Pi can participate in the IoT in various ways.

## 17.1 Controlling GPIO Outputs Using a Web Interface

### Problem

You want to control general-purpose input/output (GPIO) outputs using a web interface to your Raspberry Pi.

### Solution

Use the `bottle` Python web server library ([Recipe 7.17](#)) to create an HTML web interface to control the GPIO port.

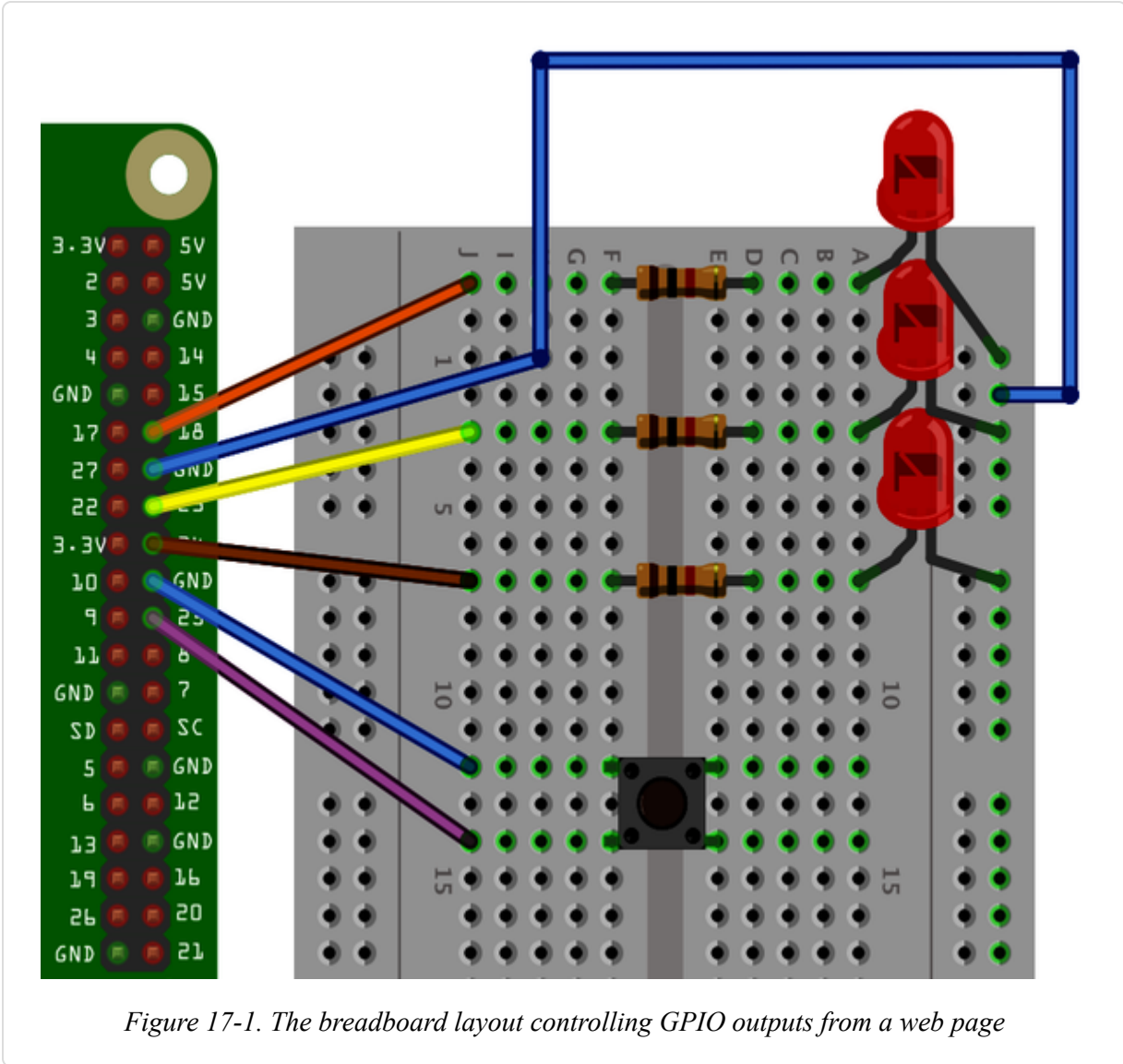
To make this recipe, you will need the following:

- Breadboard and jumper wires (see [“Prototyping Equipment and Kits”](#))
- Three 1k $\Omega$  resistors (see [“Resistors and Capacitors”](#))



- Three LEDs (see “[OptoElectronics](#)”)
- Tactile push switch (see “[Miscellaneous](#)”)

Figure 17-1 shows the breadboard layout for this.



An alternative to using a breadboard is to attach a Raspberry Squid and Squid Button (see Recipes 10.10 and 10.11). You can plug these directly into the GPIO pins of the Raspberry Pi, as shown in Figure 17-2.

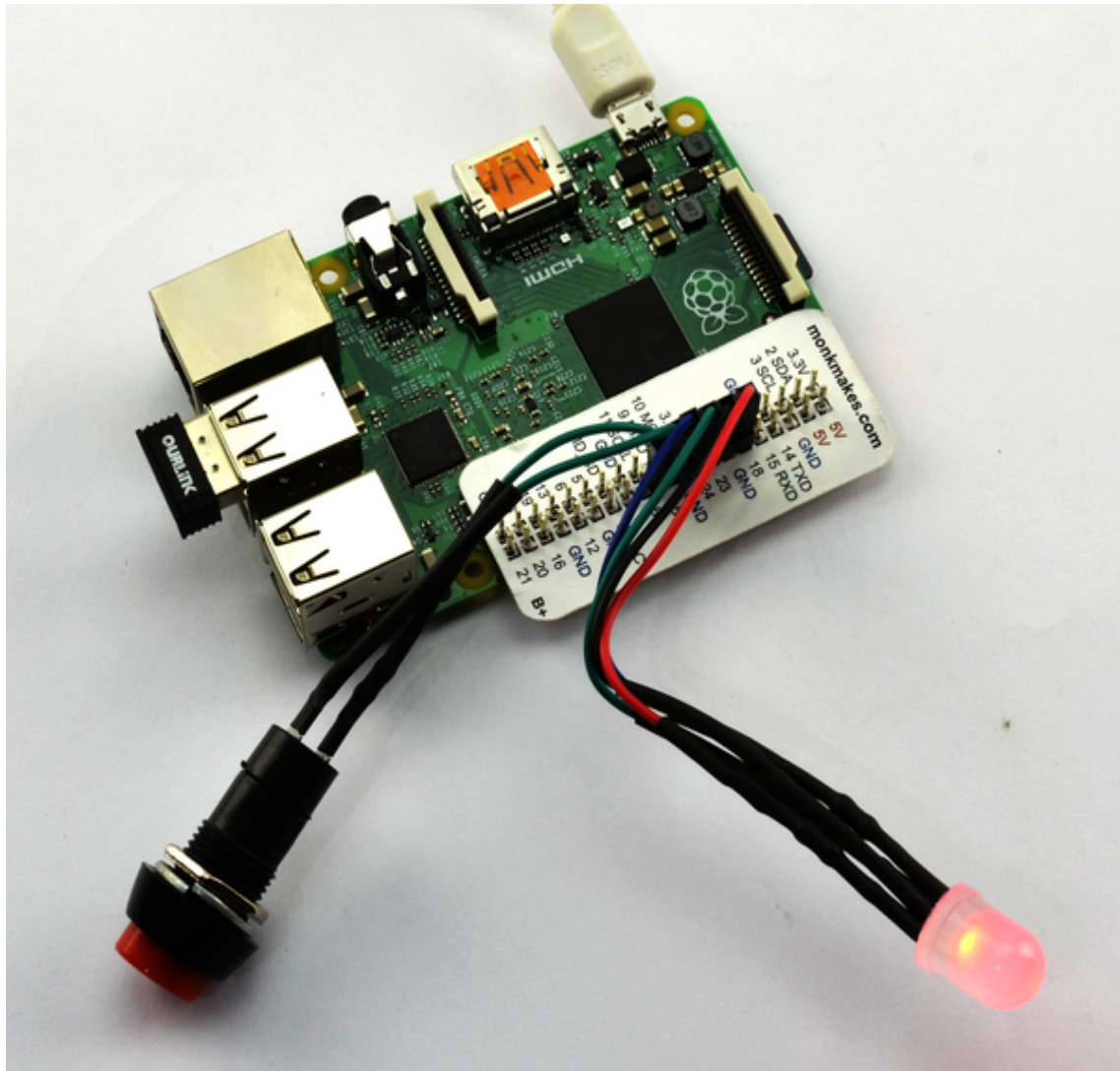


Figure 17-2. Raspberry Squid and Squid Button

To install the `bottle` library, see [Recipe 7.17](#).

Open an editor and paste in the following code (`ch_17_web_control.py`):

```
from bottle import route, run
from gpiozero import LED, Button

leds = [LED(18), LED(23), LED(24)]
switch = Button(25)

def switch_status():
    if switch.is_pressed:
```

```

        return 'Down'
    else:
        return 'Up'

def html_for_led(led_number):
    i = str(led_number)
    result = " <input type='button'
        onClick='changed(" + i + ")' value='LED " + i
        + "'/>"
    return result

@route('/')
@route('/<led_number>')
def index(led_number="n"):
    if led_number != "n":
        leds[int(led_number)].toggle()
    response = "<script>"
    response += "function changed(led) "
    response += "{"
    response += "    window.location.href='/' + led"
    response += "}"
    response += "</script>"

    response += '<h1>GPIO Control</h1>'
    response += '<h2>Button=' + switch_status() + '</h2>'
    response += '<h2>LEDs</h2>'
    response += html_for_led(0)
    response += html_for_led(1)
    response += html_for_led(2)
    return response

run(host='0.0.0.0', port=80)

```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

You must run the program as superuser:

```
$ sudo python3 ch_17_web_control.py
```

If it starts correctly, you should see a message like this:

```
Bottle server starting up (using WSGIRefServer())...
Listening on http://0.0.0.0:80/
Hit Ctrl-C to quit.
```

If you see error messages, make sure that you are using the `sudo` command.

Open a browser window from any machine on your network, even the Raspberry Pi itself, and navigate to the IP address of the Raspberry Pi (see [Recipe 2.2](#)). The web interface shown in [Figure 17-3](#) should appear.



*Figure 17-3. A web interface to GPIO*

If you click one of the three LED buttons at the bottom of the screen, you should find that the appropriate LED toggles on and off.

Also, if you hold down the button as you reload the web page, you should see that the text next to “Button” says “Down” rather than “Up.”

## Discussion

To understand how the program works, we first need to look at how a web interface works. All web interfaces rely on a server somewhere (in this case, a program on the Raspberry Pi) responding to requests from a web browser.

When the server receives a request, it looks at the information that comes with the request and formulates some HyperText Markup Language (HTML) in response.

If the web request is just to the root page (for my Raspberry Pi, the root page is *http://192.168.1.8/*), `led_number` will be given a default value of `n`. However, if we were to browse the URL *http://192.168.1.8/2*, the 2 on the end of the URL would be assigned to the `led_number` parameter.

The `led_number` parameter is then used to determine that LED 2 should be toggled.

To be able to access this LED-toggling URL, we need to arrange things so that when the button for LED 2 is clicked, the page is reloaded with this extra parameter on the end of the URL. The trick here is to include a JavaScript function in HTML that is returned to the browser. When the browser runs this function, it causes the page to be reloaded with the appropriate extra parameter.

This all means that we have a rather mind-bending situation in which the Python program is generating code in JavaScript to be run later by the browser. The lines that generate this JavaScript function are as follows:

```
response = "<script>"
response += "function changed(led) "
response += "{"
response += "    window.location.href='/' + led"
response += "}"
response += "</script>"
```

We need to also generate the HTML that will eventually call this script when a button is clicked. Rather than repeat the HTML for each of the web page buttons, this is generated by the function `html_for_led`:

```
def html_for_led(led):
    i = str(led)
    result = " <input type='button' onClick='changed(" + i + ")' "
    value = 'LED ' + i + "'/>"
    return result
```

This code is used three times, once for each button, and links a button click with the `changed` function. The function is also supplied with the LED

number as its parameter.

The process of reporting the state of the button tests to see whether the button is clicked and reports the appropriate HTML.

## See Also

For more information on using `bottle`, see the [bottle documentation](#).

## 17.2 Displaying Sensor Readings on a Web Page

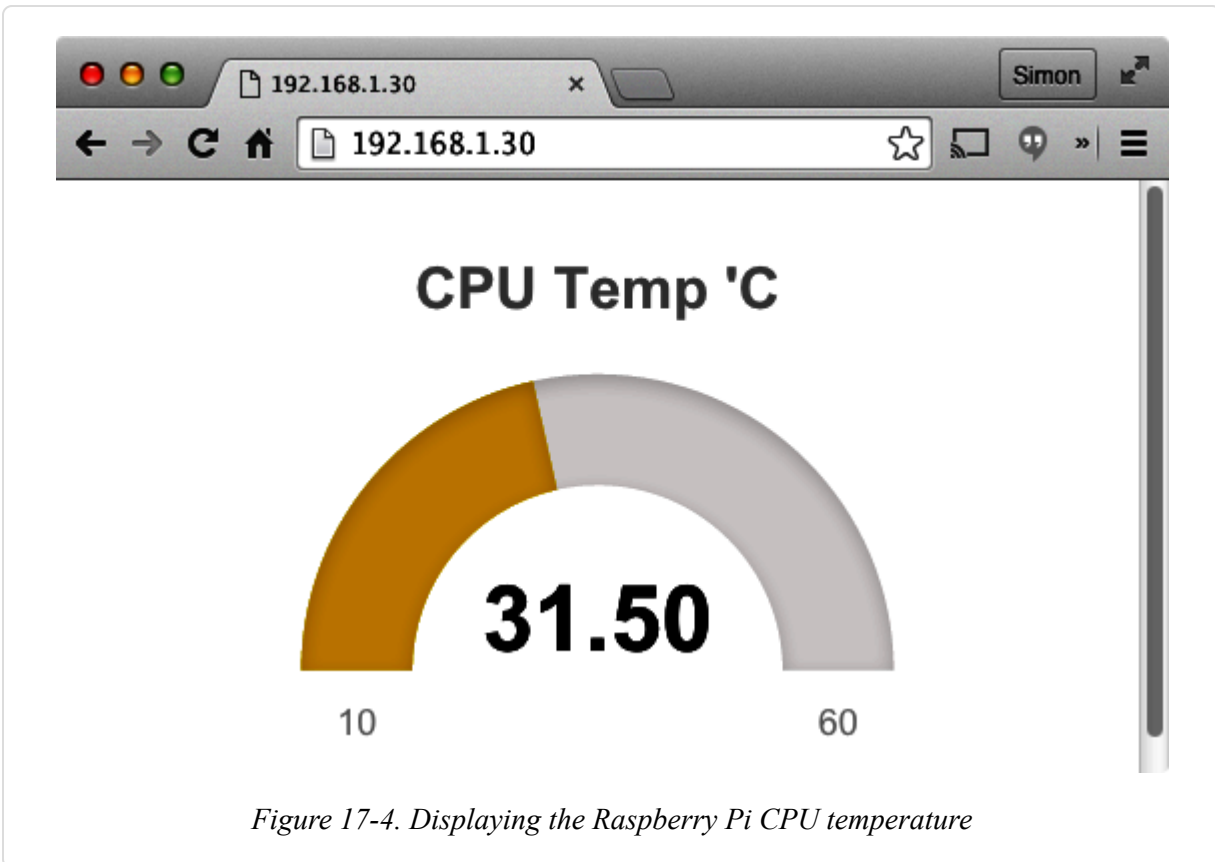
### Problem

You want to display sensor readings from your Raspberry Pi on a web page that automatically updates.

### Solution

Use the `bottle` web server and some fancy JavaScript to automatically update your display.

The example shown in [Figure 17-4](#) displays the Raspberry Pi's CPU temperature using its built-in sensor.



To install the `bottle` library, see [Recipe 7.17](#).

All four files for this example are contained in the folder `ch_17_web_sensor`:

`web_sensor.py`

Contains the Python code for the `bottle` server.

`main.xhtml`

Contains the web page that will be displayed in your browser.

`justgage.1.0.1.min.js`

A third-party JavaScript library that displays the temperature meter.

`raphael.2.1.0.min.js`

A library used by the `justgage` library.

To run the program, change directory to *ch\_17\_web\_sensor.py* and then run the Python program using the following:

```
$ sudo python3 web_sensor.py
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

Then open a browser, either on the same Raspberry Pi or on any computer on the same network as the Raspberry Pi, and enter the IP address of the Raspberry Pi into the browser's address bar. The page shown in [Figure 17-4](#) should appear.

## Discussion

The main program (*web\_sensor.py*) is quite concise:

```
import os, time
from bottle import route, run, template

def cpu_temp():
    cpu_temp = CPUtemperature()
    return str(cpu_temp.temperature)

@route('/temp')
def temp():
    return cpu_temp()

@route('/')
def index():
    return template('main.xhtml')

@route('/raphael')
def index():
    return template('raphael.2.1.0.min.js')

@route('/justgage')
def index():
    return template('justgage.1.0.1.min.js')

run(host='0.0.0.0', port=80)
```



The function `cpu_temp` reads the temperature of the Raspberry Pi's CPU, as described in [Recipe 14.11](#).

Four routes are then defined for the `bottle` web server. The first (`/temp`) returns a string containing the CPU temperature in degrees C. The root route (`/`) returns the main HTML template for the page (`main.xhtml`). The other two routes provide access to copies of the `raphael` and `justgage` JavaScript libraries.

The file `main.xhtml` mostly contains the JavaScript to render the user interface:

```
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min
.js"
type="text/javascript" charset="utf-8"></script>
<script src="raphael"></script>
<script src="justgage"></script>

<script>
function callback(tempStr, status){
if (status == "success") {
    temp = parseFloat(tempStr).toFixed(2);
    g.refresh(temp);
    setTimeout(getReading, 1000);
}
else {
    alert("There was a problem");
}
}

function getReading(){
$.get('/temp', callback);
}
</script>
</head>

<body>
<div id="gauge" class="200x160px"></div>

<script>
var g = new JustGage({
    id: "gauge",
```

```
        value: 0,  
        min: 10,  
        max: 60,  
        title: "CPU Temp 'C"  
    });  
    getReading();  
</script>  
  
</body>  
</html>
```

The `jquery`, `raphael`, and `justgage` libraries are all imported (jquery from <https://oreil.ly/nfC6s>, and the other two from local copies).

Getting a reading from the Raspberry Pi to the browser window is a two-stage process. First, the function `getReading` is called. This sends a web request with the route `/temp` to `web_sensor.py` and specifies a function called `callback` to be run when the web request completes. The `callback` function is then responsible for updating the `justgage` display before setting a timeout to call `getReading` again after a second.

## See Also

For an example of using a Python app to display sensor values in an application rather than a web page, see [Recipe 14.23](#).

The `justgage` library has all sorts of useful options for displaying sensor values.

## 17.3 Getting Started with Node-RED

### Problem

You want to create simple IoT workflows, such as sending a tweet when a button is pressed on your Raspberry Pi.

### Solution

Use the Node-RED system that can be found in the Programming section of the Recommended Software tool (Recipe 4.2). Use the following example to start a Node-RED server:

```
$ node-red-pi --max-old-space-size=256
```

Then connect to the server using a browser. This can be on the Raspberry Pi itself, in which case you can connect to the URL `http://127.0.0.1:1880/`—or if you are connecting from another computer on your network, change `127.0.0.1` to the local IP address of your Raspberry Pi (Recipe 2.2).

Figure 17-5 shows the sort of thing you can expect to see in your browser when you connect to the Node-RED server.

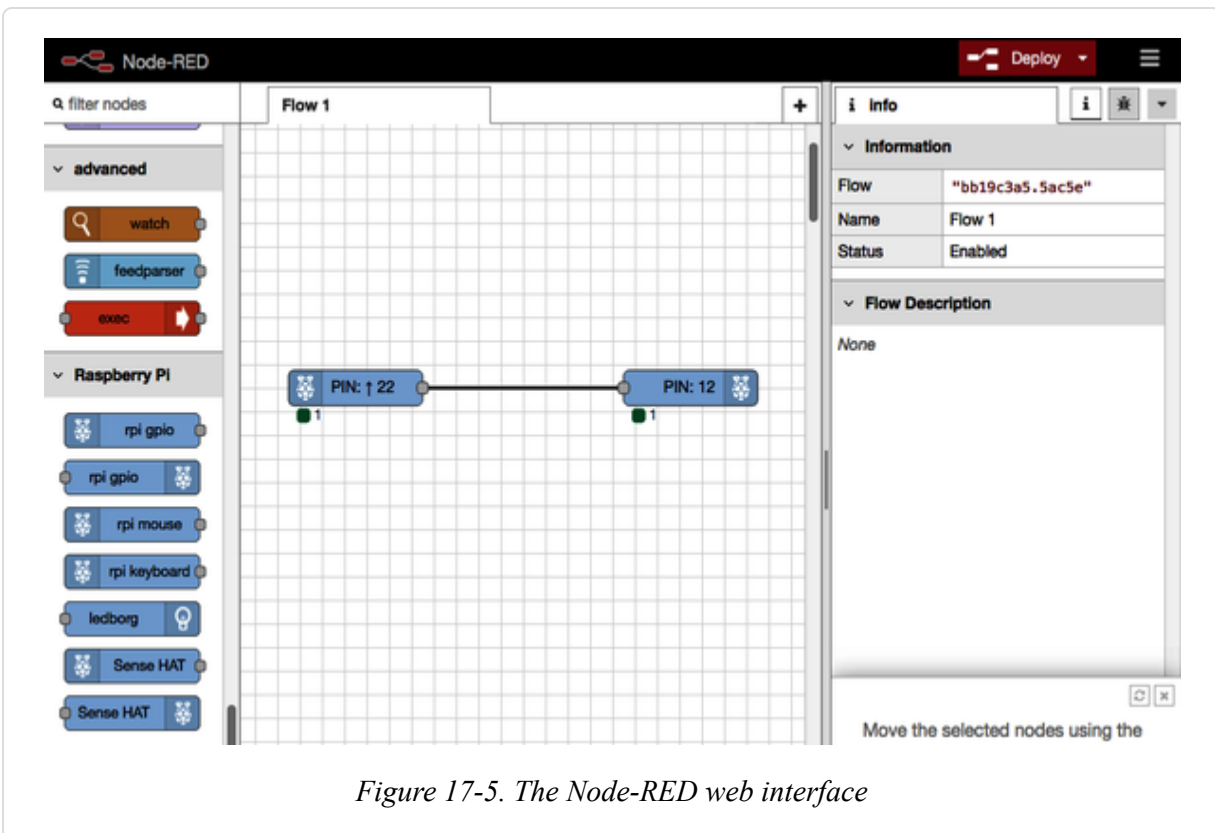


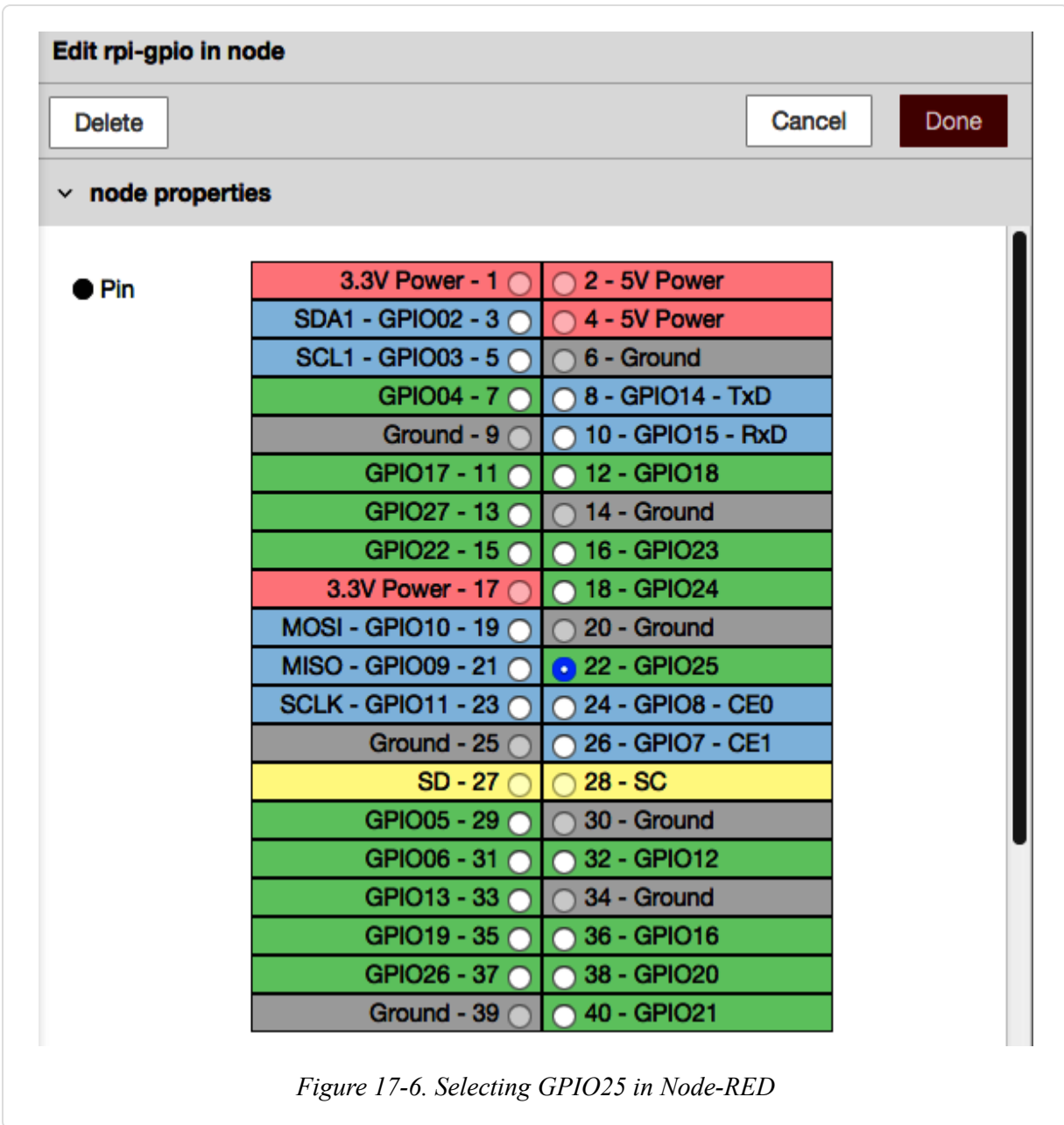
Figure 17-5. The Node-RED web interface

The idea behind Node-RED is that you draw your program (called a *flow*) rather than write code. To do this, you drag *nodes* onto the editor area and then connect them together. For example, Figure 17-5 shows a minimal flow with two Raspberry Pi pins linked together. One pin acts as an input

and might be connected to a switch (let's pick GPIO 25 as an example), and the other is connected to an LED (let's assume GPIO 18). You can accomplish this using a Squid LED and button as in [Recipe 17.1](#).

The node on the left, labeled PIN 22, is the input (connected to the switch on GPIO 25), and PIN 12 is the output (connected to the LED on GPIO 18). Node-RED uses the pin positions (where the top left pin is pin 1, its neighbor to the right is pin 2, and so on) rather than the GPIO names.

To create this flow in your editor, scroll down the list of nodes on the left until you get to the Raspberry Pi section. Drag the "rpi gpio" node that has a Raspberry Pi icon on the left side to the editor area. This will be the input. Double-click it to open the window shown in [Figure 17-6](#). Select "22 - GPIO25" and click Done.



Next, select the other “rpi gpio” node type (with the Raspberry Pi icon on the right) and drag it to the editor area, and then double-click it to open the window shown in [Figure 17-6](#); this time select “12 - GPIO18,” and click Done.

Link the two nodes together by dragging out from the round connector on the right of the input node, so that the flow looks like [Figure 17-5](#).

Assuming that you have a push switch and LED connected to your Raspberry Pi, you can now run this flow by clicking the Deploy button. The LED should light, and when you press the button on the switch connected to your Raspberry Pi, the LED should turn off. The logic of this is inverted, but let's leave that for now. We revisit this in the next chapter, where you learn a lot more about Node-RED.

This is all very neat, but as yet it does not have much to do with the IoT. This is where some of the other node types of Node-RED come into play. If you browse the list, you'll find all sorts of nodes, including the Tweet node. You can connect this node as a second output to the PIN 22 node, so that the flow now looks like [Figure 17-7](#). Double-click the Tweet node to configure it with your Twitter credentials. Now when you press the button, a tweet will be sent.

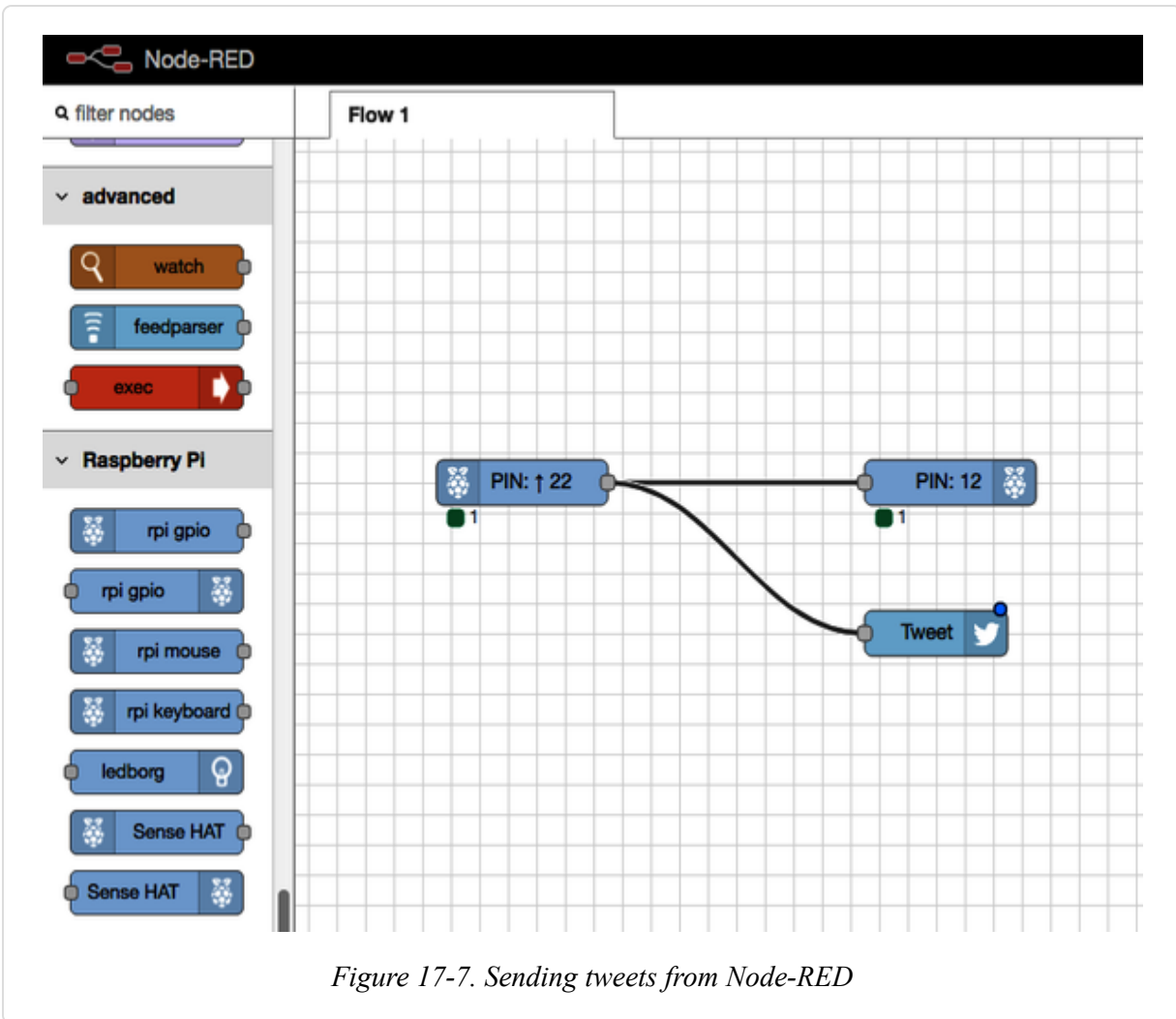


Figure 17-7. Sending tweets from Node-RED

## Discussion

Node-RED is an extremely powerful system, and consequently, becoming familiar with all its features and its odd quirks will take some time. As well as creating direct flows like the one we made here, you can also introduce switching code (like an `if` statement) and functions that transform the messages being passed between the nodes.

We have only just touched on Node-RED here; if you want to learn more, I suggest working through the documentation mentioned in the See Also section and much of [Chapter 18](#).

Having played with Node-RED and decided that you like it, you may well want to ensure that it starts automatically whenever your Raspberry Pi

reboots by issuing the following commands:

```
$ sudo systemctl enable nodered.service
$ sudo systemctl start nodered.service
```

## See Also

More information is available in the [full documentation on using Node-RED on the Raspberry Pi](#).

Find some good video introductions to Node-RED at <https://oreil.ly/uxGHZ> and <https://oreil.ly/ZSDNi>.

## 17.4 Sending Email and Other Notifications with IFTTT

### Problem

You want a flexible way for your Raspberry Pi to send notifications through email, Facebook, Twitter, or Slack.

### Solution

Have your Raspberry Pi send requests to the If This Then That (IFTTT) *Maker* channel to trigger configurable notifications.

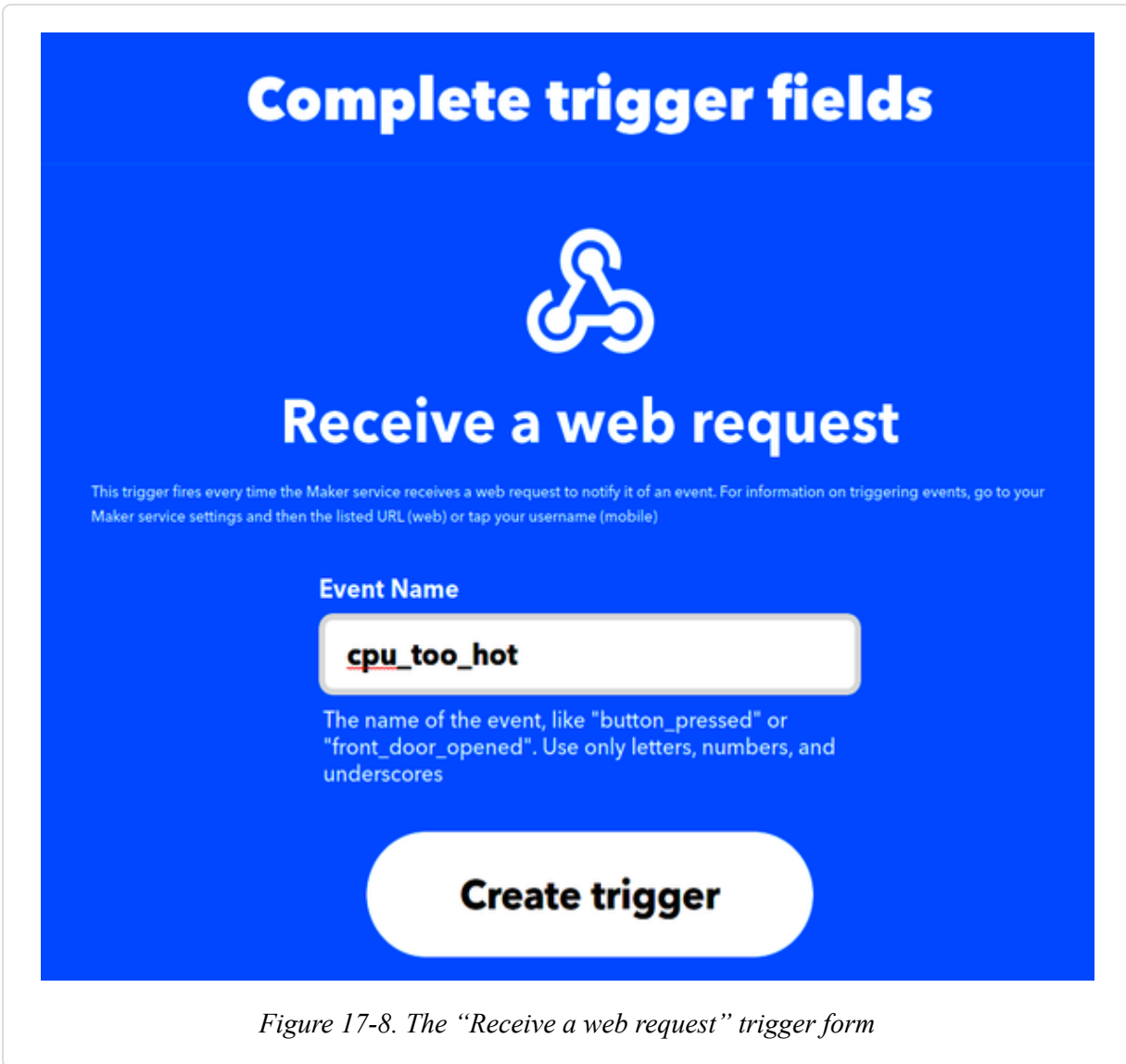
This recipe is illustrated with an example that sends you an email when the CPU temperature of your Raspberry Pi exceeds a threshold.

You need to create an account with IFTTT before you can begin using it, so visit [www.ifttt.com](http://www.ifttt.com), sign up for a free account, and log in.

The next step is to create a new IFTTT *applet* on the IFTTT website. An applet is like a rule, such as *When I get a web request from a Raspberry Pi, send an email*. Click the CREATE button on the web page. This will prompt you to first enter the IF THIS part of the recipe and later the THEN THAT part.



In this case, the IF THIS part (the trigger) is going to be the receipt of a web request from your Raspberry Pi, so click THIS and then enter **Webhooks** into the search field to find the Webhooks channel. Select the Webhooks channel and, when prompted, select the option “Receive a web request.” Clicking Create should bring up something like [Figure 17-8](#).



*Figure 17-8. The “Receive a web request” trigger form*

In the Event Name field, enter the text **cpu\_too\_hot** and then click “Create trigger.”

This will now move you to the THEN THAT portion of the recipe, the *action*, and you will need to select an action channel. There are many

options, but for this example, you will use the Email channel, so in the search field, type **Email** and then select the Email channel.

Having selected the Email channel, select the action “Send me an email,” which displays the form shown in [Figure 17-9](#).

Change the text so that it appears as shown in [Figure 17-9](#). Note that the special values OccurredAt and Value1 will both be surrounded by {{ and }}. These values are called ingredients and are variable values that will be taken from the web request and substituted into the email subject and body.

Click “Create action” and then Finish to complete the recipe creation.



# Complete action fields

Step 5 of 6

## Send me an email

This Action will send you an HTML based email. Images and links are supported.

### Subject

Raspberry Pi CPU Temperature  
Warning

**Add ingredient**

### Body

Raspberry Pi CPU temperature  
warning at **OccurredAt** .  
Temperature measured as  
**Value1** degrees C

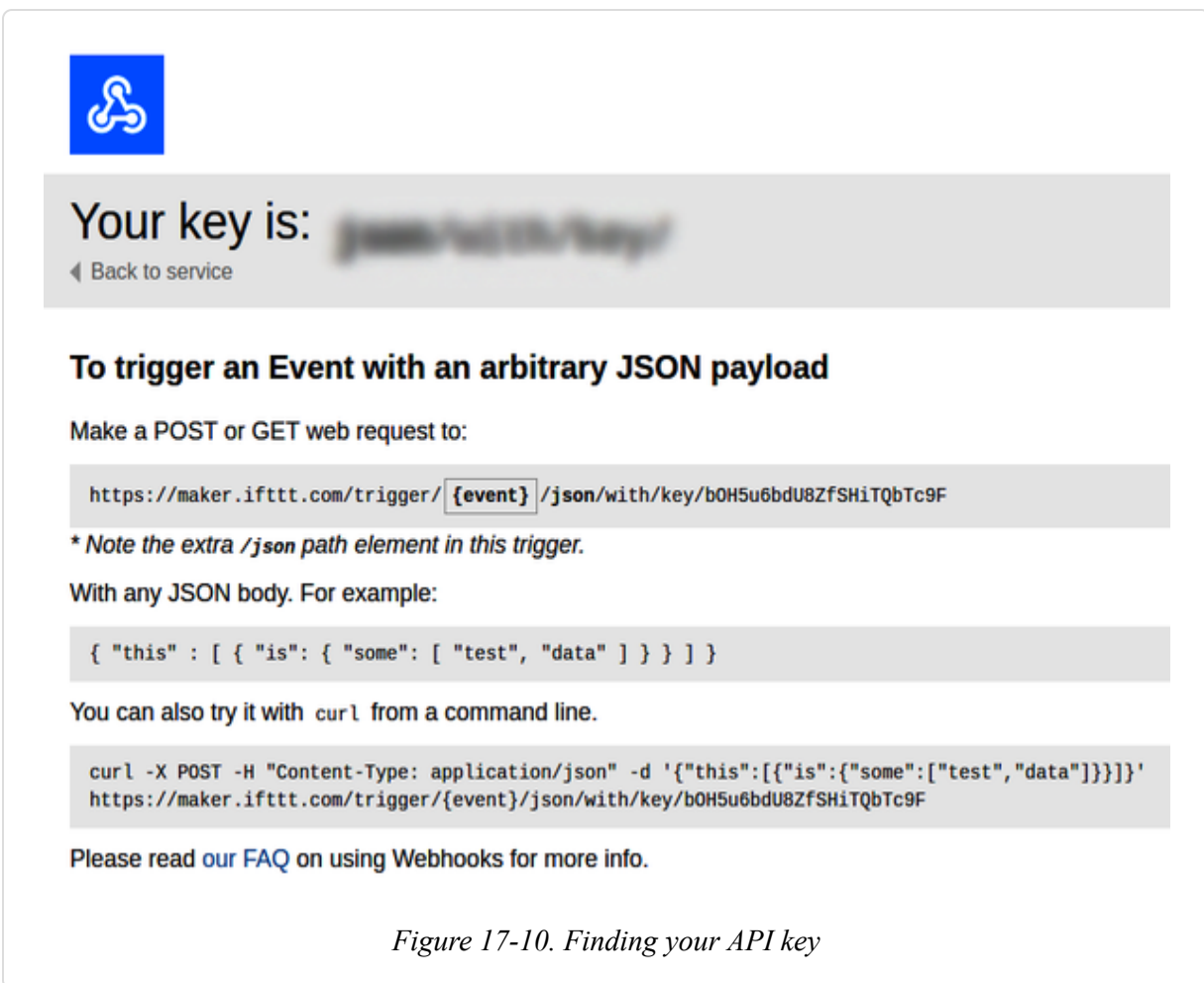
**Add ingredient**

**Create action**

Figure 17-9. Completing the action fields in IFTTT

One final piece of information that we need is the API key for the Webhooks channel. This is so that other people can't bombard you with emails about their Raspberry Pi's CPU temperature.

To find this key on the IFTTT website, click the My Services option on the menu that appears when you click next to your user icon, and then find Webhooks. On the Webhooks page, click Documentation, and a page like **Figure 17-10** displays; here you can see your key (which is intentionally obscured in this figure). You will need to paste this key into the code that follows in the line `KEY = 'your_key_here'`.



The screenshot shows the IFTTT Webhooks documentation page. At the top left is the IFTTT logo. Below it, a grey box contains the text "Your key is:" followed by a blurred API key. A link "Back to service" is visible below the key. The main heading is "To trigger an Event with an arbitrary JSON payload". Below this, it says "Make a POST or GET web request to:" followed by a code block containing the URL: `https://maker.ifttt.com/trigger/{event}/json/with/key/b0H5u6bdU8ZfSHiTQbTc9F`. A note states: "\* Note the extra /json path element in this trigger." Below this, it says "With any JSON body. For example:" followed by a code block containing a JSON payload: `{ "this" : [ { "is": { "some": [ "test", "data" ] } } ] }`. It then says "You can also try it with curl from a command line." followed by a code block containing a curl command: `curl -X POST -H "Content-Type: application/json" -d '{"this":[{"is":{"some":["test","data"]}]}' https://maker.ifttt.com/trigger/{event}/json/with/key/b0H5u6bdU8ZfSHiTQbTc9F`. At the bottom, it says "Please read our FAQ on using Webhooks for more info."

*Figure 17-10. Finding your API key*

The Python program to send the web request is called `ch_17_ifttt_cpu_temp.py`:

```

import time
from gpiozero import CPUTemperature
import requests

MAX_TEMP = 33.0
MIN_T_BETWEEN_WARNINGS = 60 # Minutes

EVENT = 'cpu_too_hot'
BASE_URL = 'https://maker.ifttt.com/trigger/'
KEY = 'your_key_here'

def send_notification(temp):
    data = {'value1' : temp}
    url = BASE_URL + EVENT + '/with/key/' + KEY
    response = requests.post(url, json=data)
    print(response.status_code)

def cpu_temp():
    cpu_temp = CPUTemperature().temperature
    return cpu_temp

while True:
    temp = cpu_temp()
    print("CPU Temp (C): " + str(temp))
    if temp > MAX_TEMP:
        print("CPU TOO HOT!")
        send_notification(temp)
        print("No more notifications for: " +
str(MIN_T_BETWEEN_WARNINGS)
        + " mins")
        time.sleep(MIN_T_BETWEEN_WARNINGS * 60)
        time.sleep(1)

```

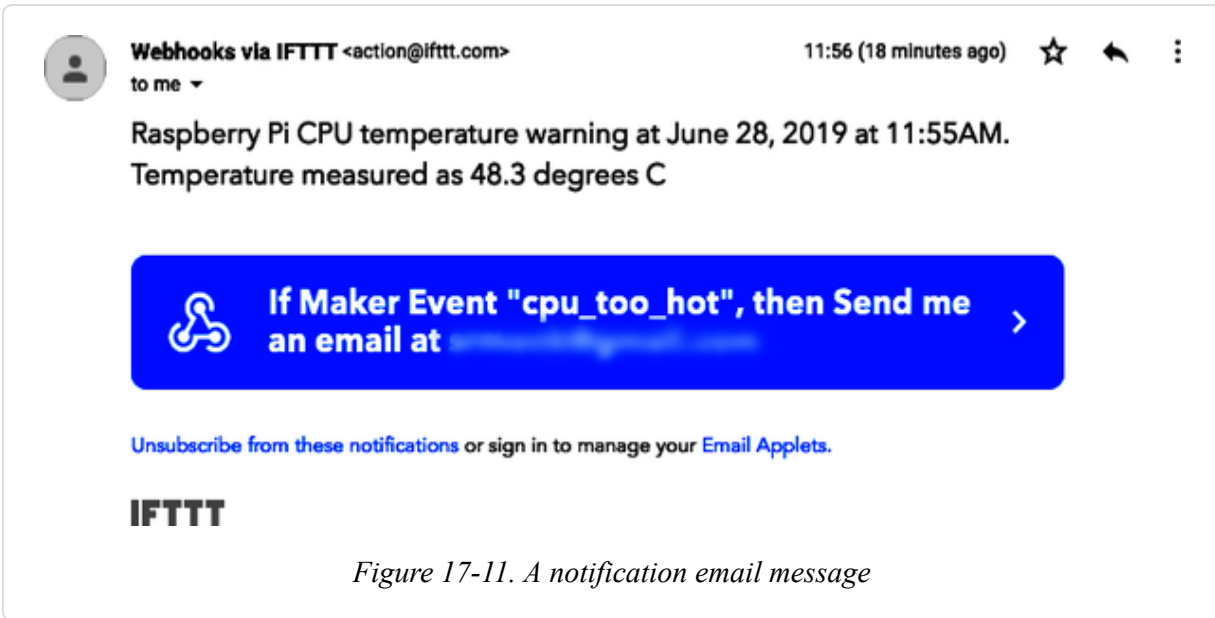
As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

I have left the `MAX_TEMP` deliberately low for testing purposes. If you live somewhere hot, you'll want to bump this number up to 60 or 70.

Paste the key into `ch_17_ifttt_cpu_temp.py` on the line that starts `KEY=` and then run the program using:

```
$ python3 ch_17_ifttt_cpu_temp.py
```

You can increase your CPU temperature by playing a video or *temporarily* wrapping your Raspberry Pi in bubble wrap. When the event is triggered, you should receive an email that looks like **Figure 17-11**. Notice how the values have been substituted into the email (again, the obscuring is intentional).



*Figure 17-11. A notification email message*

## Discussion

Most of the action for this program takes place in the `send_notification` function. This function first constructs a URL that includes the key and request parameter `value1` (containing the temperature) and then uses the Python `requests` library to send the web request to IFTTT.

The main loop continually checks the CPU temperature against the `MAX_TEMP`; if the temperature exceeds `MAX_TEMP`, the web request is sent, and a long sleep is started as specified by `MIN_T_BETWEEN_WARNINGS`. The sleep prevents your inbox from being flooded with notifications.

As an alternative to using IFTTT, you could, of course, just send an email directly using **Recipe 7.16**. However, by using IFTTT to send the messages,

you are not restricted to email notifications—you could use any of the action channels available in IFTTT without having to write any code.

## See Also

To send an email directly from Python, see [Recipe 7.16](#).

The code to measure the CPU temperature is described in [Recipe 14.11](#).

## 17.5 Sending Tweets Using ThingSpeak

### Problem

You want to automatically send tweets from your Raspberry Pi—for example, to irritate people by telling them the temperature of your CPU.

### Solution

You could just use [Recipe 17.4](#) and change the action channel to Twitter. However, the ThingSpeak service is an alternative way of doing this.

**ThingSpeak** is similar to IFTTT but is aimed squarely at IoT projects. It allows you to create channels that can store and retrieve data using web requests, and it also has a number of actions, including ThingTweet, which provides a web services wrapper around Twitter. This is easier to use than the Twitter API, which requires you to register your application with Twitter.

Start by visiting <https://thingspeak.com> and signing up. Note that this also involves creating a MATLAB account for no particularly good reason.

Next, select the ThingTweet action from the Apps menu. You are prompted to log in to Twitter, and then your action will become activated ([Figure 17-12](#)).

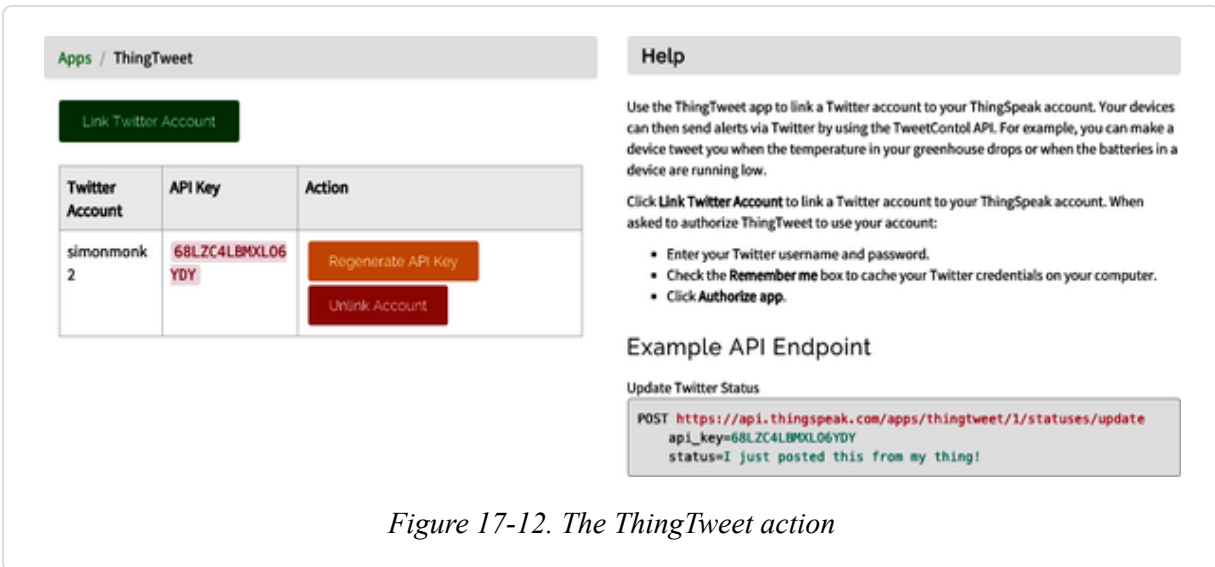


Figure 17-12. The ThingTweet action

The Python program to send the web request that triggers the tweet is called `ch_17_send_tweet.py`:

```
import time
from gpiozero import CPUtemperature
import requests

MAX_TEMP = 35.0
MIN_T_BETWEEN_WARNINGS = 60 # Minutes

BASE_URL =
'https://api.thingspeak.com/apps/thingtweet/1/statuses/update/'
KEY = 'your_key_here'

def send_notification(temp):
    status = 'Thingtweet: Raspberry Pi getting hot. CPU temp=' +
str(temp)
    data = {'api_key' : KEY, 'status' : status}
    response = requests.post(BASE_URL, json=data)
    print(response.status_code)

def cpu_temp():
    cpu_temp = CPUtemperature().temperature
    return cpu_temp

while True:
    temp = cpu_temp()
    print("CPU Temp (C): " + str(temp))
    if temp > MAX_TEMP:
        print("CPU TOO HOT!")
```



```
        send_notification(temp)
        print("No more notifications for: " +
str(MIN_T_BETWEEN_WARNINGS)
        + " mins")
        time.sleep(MIN_T_BETWEEN_WARNINGS * 60)
        time.sleep(1)
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

As with [Recipe 17.4](#), you need to paste your key from [Figure 17-12](#) into the code before you run the program. Run and test the program in the same way you did in [Recipe 17.4](#).

## Discussion

The code is very similar to [Recipe 17.4](#). The main difference is in the function `send_notification`, which constructs the tweet and then sends the web request with the message as the parameter `status`.

## See Also

More information is available in the [full documentation of the ThingSpeak service](#).

In [Recipe 17.6](#), you use the popular CheerLights, implemented in ThingSpeak; in [Recipe 17.7](#), you learn how to use ThingSpeak to collect sensor data.

# 17.6 Changing LED Color Using CheerLights

## Problem

You want to hook your Raspberry Pi up to an RGB LED and participate in the popular CheerLights project.

CheerLights is a web service that, when anyone sends a tweet to `@cheerlights` containing the name of a color, will record that color as being

*the* CheerLights color. Around the world, many people have CheerLights projects that use a web service to request the last color and set their lighting to that color. So when anyone tweets, everyone's lights change color.

## **Solution**

Use a Raspberry Squid RGB LED connected to your Raspberry Pi (Figure 17-13) and run the test program called *ch\_17\_cheerlights.py* (shown after Figure 17-13):

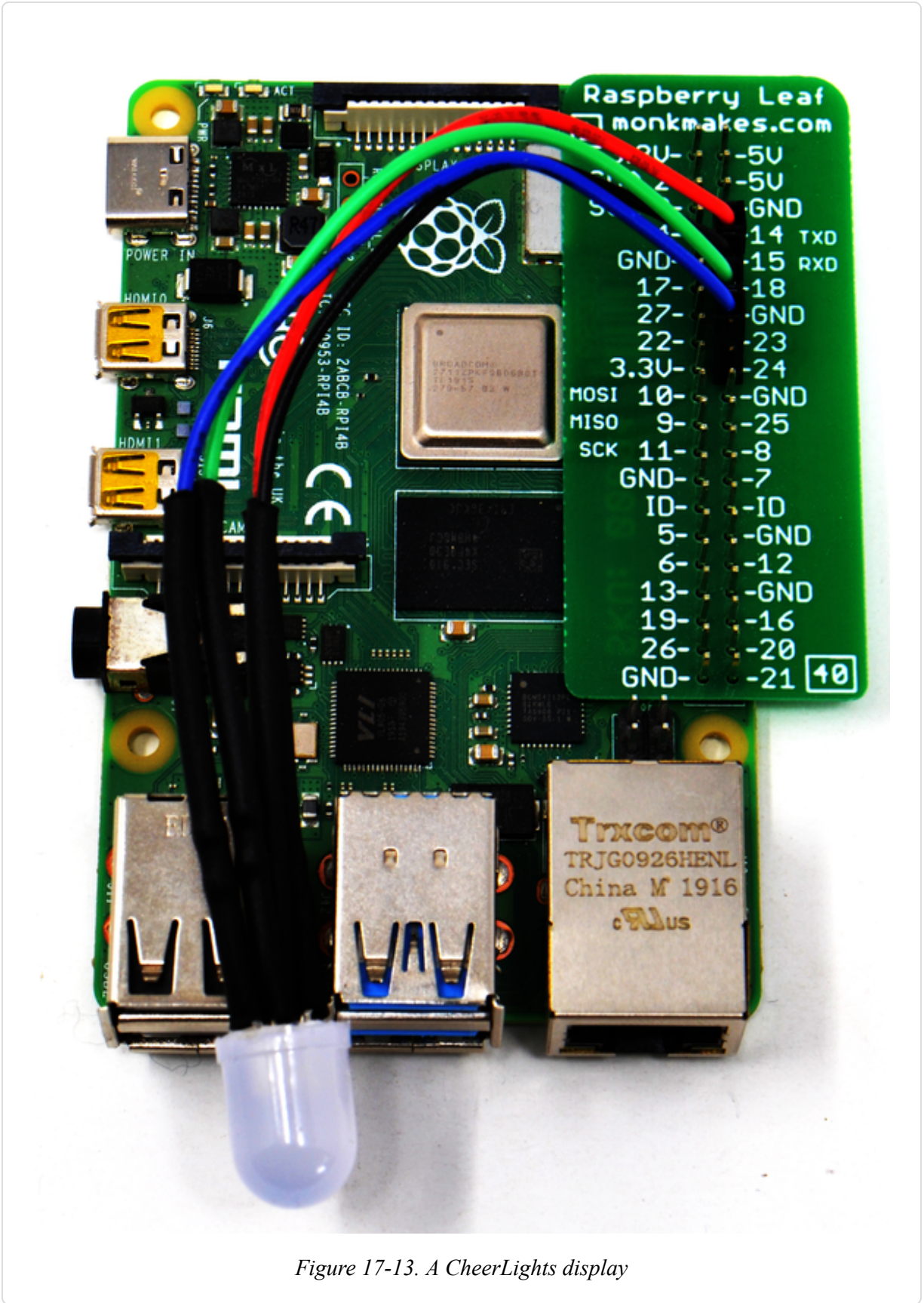


Figure 17-13. A CheerLights display

```

from gpiozero import RGBLED
from colorzero import Color
import time, requests

led = RGBLED(18, 23, 24)
cheerlights_url =
"http://api.thingspeak.com/channels/1417/field/2/last.txt"

while True:
    try:
        cheerlights = requests.get(cheerlights_url)
        c = cheerlights.content
        print(c)
        led.color = Color(c)
    except Exception as e:
        print(e)
        time.sleep(2)

```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

When you run the program, your LED should immediately set itself to a color. It will probably change color after a while when someone tweets; if it doesn't, try tweeting a message such as “@cheerlights red,” and the color of your LED and the rest of the world's LEDs should change. Valid color names for CheerLights are red, green, blue, cyan, white, oldlace, purple, magenta, yellow, orange, and pink.

## Discussion

The code just sends a web request to ThingSpeak, which returns a string of colors as a six-digit hexadecimal number. This is then used to set the LED color.

The `try/except` code is used to ensure that the program doesn't crash if a temporary network outage occurs.

## See Also

CheerLights uses ThingSpeak to store the last color in a channel. In [Recipe 17.7](#), a channel is used to record sensor data.

If you don't have a Squid, you can use an RGB on a breadboard (see [Recipe 11.11](#)), or you can even adapt [Recipe 15.5](#) to control an entire LED strip.

## 17.7 Sending Sensor Data to ThingSpeak

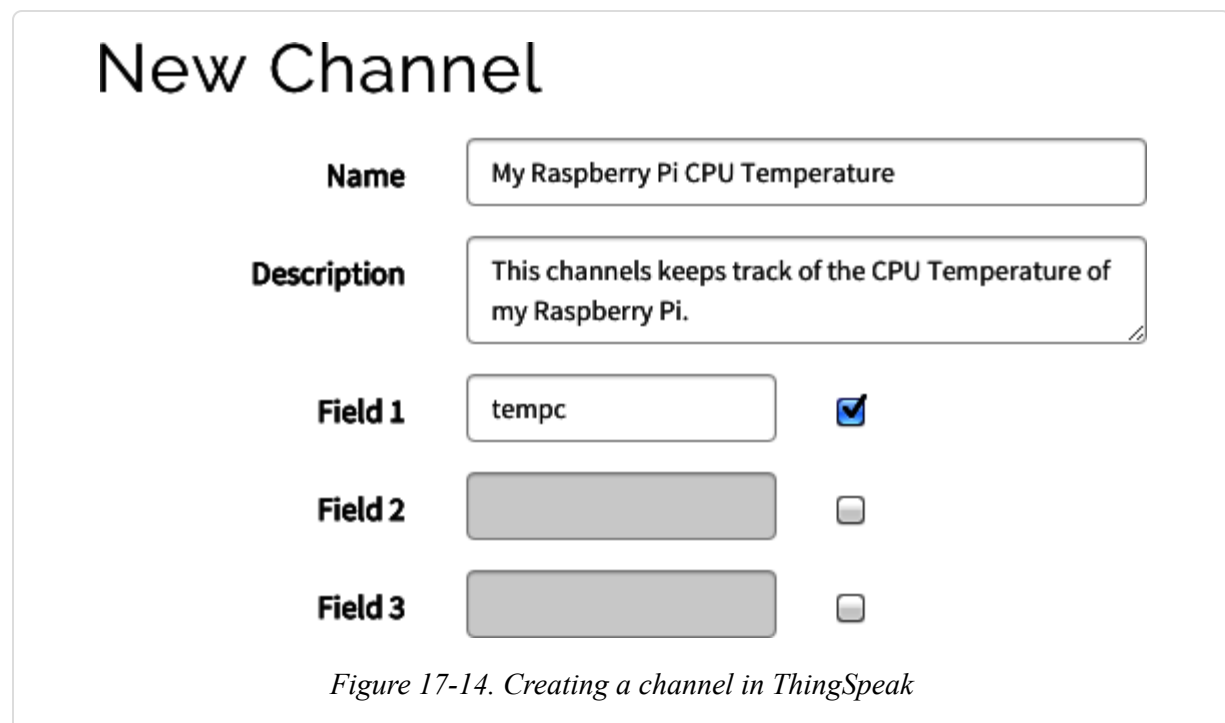
### Problem

You want to log sensor data to ThingSpeak and then see charts of the data over time.

### Solution

Log in to ThingSpeak and, from the Channels drop-down, select My Channels. Next, create a new channel by completing the top of the form, as shown in [Figure 17-14](#).

You can leave the rest of the form blank. When you have finished editing, click Save Channel at the bottom of the page. Click the API Keys tab to find a summary of the web requests that you can use, along with keys for the channel you just created ([Figure 17-15](#)).



**New Channel**

**Name** My Raspberry Pi CPU Temperature

**Description** This channels keeps track of the CPU Temperature of my Raspberry Pi.

**Field 1** tempc

**Field 2**

**Field 3**

*Figure 17-14. Creating a channel in ThingSpeak*

### Update Channel Feed - GET

```
GET https://api.thingspeak.com/update?api_key=DYHHDDKCLU80V58T&field1=0
```

### Update Channel Feed - POST

```
POST https://api.thingspeak.com/update.json
api_key=DYHHDDKCLU80V58T
field1=73
```

### Get a Channel Feed

```
GET https://api.thingspeak.com/channels/77483/feeds.json?results=2
```

### Get a Channel Field Feed

```
GET https://api.thingspeak.com/channels/77483/fields/1.json?results=2
```

### Get Status Updates

```
GET https://api.thingspeak.com/channels/77483/status.json
```

Figure 17-15. Specifying a ThingSpeak channel

You will need to copy your API key (*api\_key* in Update Channel Feed - POST as shown in Figure 17-15) as the value for `KEY` in the program that follows.

To send data to the channel, you must send a web request. The Python program to send the web request is called *ch\_17\_thingspeak\_data.py*:

```
import time
from gpiozero import CPUtemperature
import requests

PERIOD = 60 # seconds
BASE_URL = 'https://api.thingspeak.com/update.json'
KEY = 'your key goes here'

def send_data(temp):
```

```

data = {'api_key' : KEY, 'field1' : temp}
response = requests.post(BASE_URL, json=data)

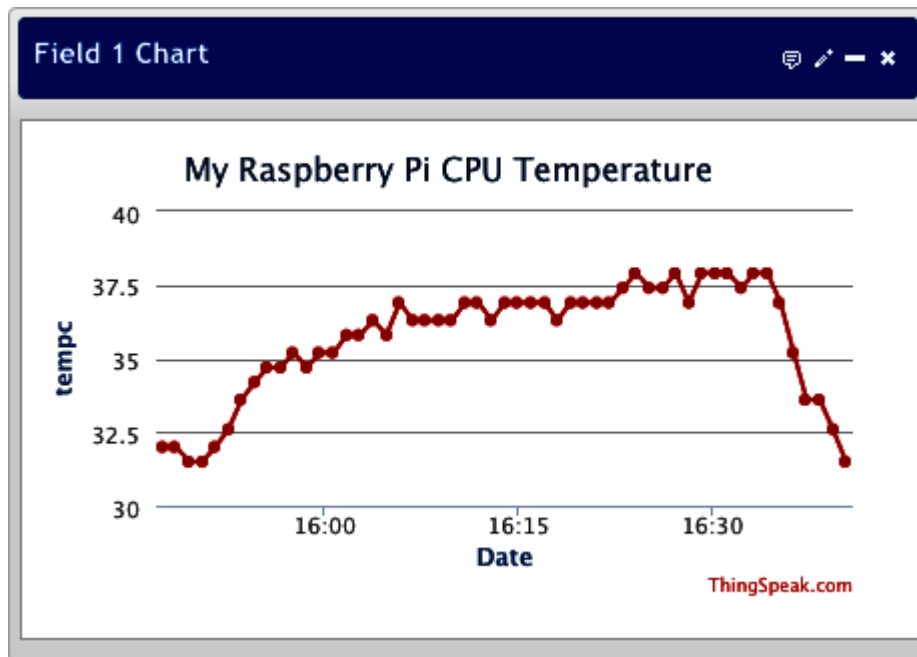
def cpu_temp():
    cpu_temp = CPUtemperature().temperature
    return cpu_temp

while True:
    temp = cpu_temp()
    print("CPU Temp (C): " + str(temp))
    send_data(temp)
    time.sleep(PERIOD)

```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

Run the program. On the ThingSpeak channel page, on the Private View tab, you should see a graph like the one shown in [Figure 17-16](#).



*Figure 17-16. Charting the sensor data in ThingSpeak*

This will update every minute as each new reading arrives.

## Discussion

The variable `PERIOD` is used to determine the time interval in seconds after each sending of the temperature.

The `send_data` function constructs the web request, supplying the temperature in a parameter called `field1`.

If your data might be something of public interest—say, accurate environmental readings—you might want to make the channel public so that anyone can make use of it. This probably isn't the case for your Pi's CPU temperature.

## See Also

For an explanation of the code that reads the CPU temperature, see [Recipe 14.11](#).

For an example of exporting sensor data into a spreadsheet, see [Recipe 14.24](#).

## 17.8 Responding to Tweets Using Dweet and IFTTT

### Problem

You want your Raspberry Pi to perform some action in response to a certain hashtag or mention in a tweet.

[Recipe 17.6](#) does this, but it does it very inefficiently because it relies on you continually polling with web requests to see whether the color has changed.

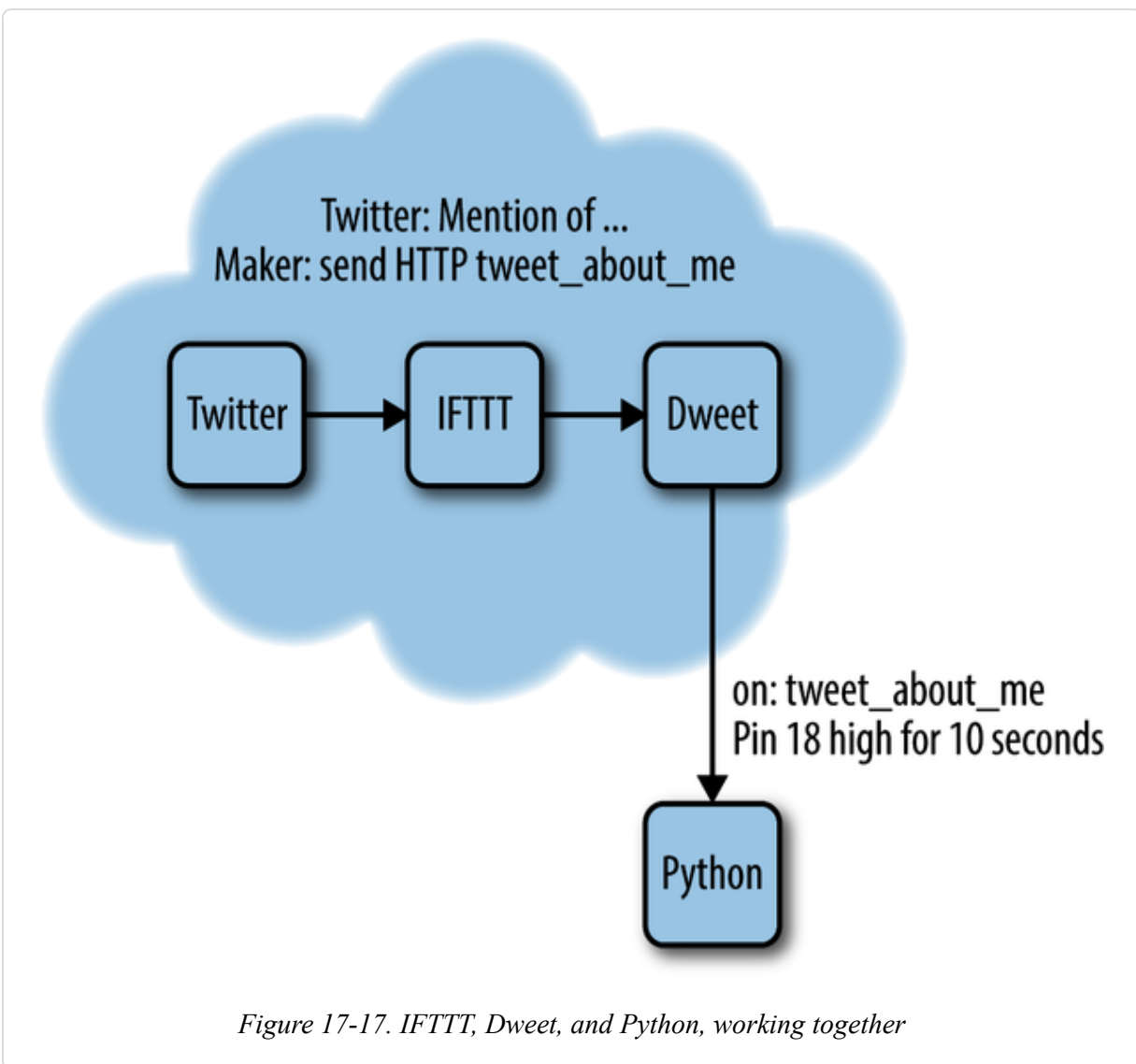
### Solution

An efficient mechanism for monitoring tweets that does not rely on polling is to use IFTTT (see [Recipe 17.4](#)) to spot tweets of interest and then send a web request to a service called Dweet that can push notifications to a Python program running on your Raspberry Pi ([Figure 17-17](#)).



For example, you could flash an LED for 10 seconds every time your username is mentioned on Twitter by using a Raspberry Squid or an LED attached to a breadboard.

As far as the hardware goes, this recipe just requires some electronics that do something noticeable when GPIO 18 goes high. This could be one channel of a Raspberry Squid (see [Recipe 10.10](#)) or a single LED attached to breadboard (see [Recipe 11.1](#)) or, for ultimate flexibility, a relay (see [Recipe 11.5](#)). Using a relay would allow you to create a project like [Bubblino](#), the bubble-blowing Arduino bot.



*Figure 17-17. IFTTT, Dweet, and Python, working together*

The first step is to log in to IFTTT (see [Recipe 17.4](#)) and then create a new applet. Choose an action channel of “New Mention of You” and then click “Create trigger.” For the recipe’s action channel, select Webhooks, and then select the action “Make a web request” and complete the fields, as shown in [Figure 17-18](#).

The URL includes a request parameter with the ingredient of “text.” This will contain the body of the tweet. Although this will not be used other than to print it in the console, you might have the message displayed on an LCD screen for a more sophisticated project, so it is useful to know how to pass data from a tweet to the Python program.

Finally, click “Create recipe” to take the IFTTT recipe live.

The `dweet.io` web service operates rather like Twitter for IoT things. It has a web interface that allows you to both post and listen for *dweets*. Dweet doesn’t require an account or any login details to use it; you can just have one thing (IFTTT, in this case) send a message to Dweet and have another thing (your Raspberry Pi Python program) wait for a notification from the service that something you are interested in has happened. In this case, the token that links the two is `tweet_about_me`. This is not very unique, and if several people are trying out this example from the book at the same time, you will all get one another’s messages. To avoid this, use a more unique token (say, by adding a random string of letters and numbers to the message).



# Complete action fields

Step 5 of 6

**Make a web request**

This action will make a web request to a publicly accessible URL. NOTE: Requests may be rate limited.

**URL**

`https://dweet.io/dweet/for/tweet_about_me?text=`

Surround any text with "<<>>" to escape the content

**Method**

The method of the request e.g. GET, POST, DELETE

**Content Type**

Optional

**Body**

Surround any text with "<<>>" to escape the content

Figure 17-18. Completing the "Make a web request" action channel in IFTTT

To access Dweet from your Python program, you need to install the `dweepy` library using the following command:

```
$ sudo pip3 install dweepy
```

The program for this recipe is called `ch_17_twitter_trigger.py`:

```
import time
import dweepy
from gpiozero import LED

KEY = 'tweet_about_me'
led = LED(18)

while True:
    try:
        for dweet in dweepy.listen_for_dweets_from(KEY):
            print('Tweet: ' + dweet['content']['text'])
            led.on()
            time.sleep(10)
            led.off()
    except Exception:
        pass
```

As with all the program examples in this book, you can also download this program (see [Recipe 3.22](#)).

After it's running, try mentioning yourself in a tweet, and the LED should light for 10 seconds.

## Discussion

The program uses the `listen_for_dweets_from` method to leave an open connection to the `dweet.io` server, listening for any push messages from the server as a result of a dweet arriving from IFTTT in response to a tweet. The `try/except` block ensures that, if any communication outage occurs, the program will just start the listening process again.

## See Also

For a similar project using a different approach, see [Recipe 17.6](#).

# Chapter 18. Home Automation

---

## 18.0 Introduction

As a low-cost and low-power device, a Raspberry Pi makes a great home automation hub that you can leave running without fear of huge electricity bills. For the recipes described in this chapter, you don't need the power of a Raspberry Pi 4 or 400. In fact, a Raspberry Pi 2 or 3 will be plenty fast enough and will run cooler and use less electricity than a Raspberry Pi 4.

We start with Message Queuing Telemetry Transport (MQTT), the basic communication mechanism for most home automation systems, and then move on to look at using Node-RED (which you first met in [Chapter 17](#)) as a basis for home automation.

Strictly speaking, home automation is all about making your home smarter and more able to do things for itself—for example, to turn on a light for a set amount of time when movement is detected, or to automatically turn everything off at bedtime. But most people who are interested in home automation are also interested in remote control of the parts of their home that have been automated. We also look at remote control by smartphone in this chapter.

## 18.1 Making a Raspberry Pi into a Message Broker with Mosquitto

### Problem

You want to make your Raspberry Pi a hub for your home automation system capable of using the Mosquitto MQTT software.

### Solution

Install the Mosquitto software so that your Raspberry Pi can act as an MQTT broker.

Run the following commands to install Mosquitto and start it as a service so that it automatically starts when your Raspberry Pi reboots:

```
$ sudo apt update
$ sudo apt install -y mosquitto mosquitto-clients
$ sudo systemctl enable mosquitto.service
```

You can check to see whether everything is working by running the following command:

```
$ mosquitto -v
1656413574: mosquitto version 2.0.11 starting
1656413574: Using default config.
1656413574: Starting in local only mode. Connections will only be
possible
  from clients running on this machine.
1656413574: Create a configuration file which defines a listener
to allow
  remote access.
1656413574: For more details see
https://mosquitto.org/documentation
/authentication-methods/
1656413574: Opening ipv4 listen socket on port 1883.
1656413574: Error: Address already in use
1656413574: Opening ipv6 listen socket on port 1883.
1656413574: Error: Address already in use
$
```

The error message is not really an error; it just means that Mosquitto is already running because we started it as a service.

## Discussion

MQTT is a way of passing messages between one program and another. It has two parts:

Server

The central place where the passing of messages is controlled and messages are routed to the right recipient.

## Client

A program that sends and receives messages to and from the server. Normally more than one client will be in a system.

Messages are passed using what is called a *publish and subscribe* model. That is, a client that has something interesting to say (e.g., a sensor reading) tells the server by publishing the reading. Every few seconds, the client might take another reading and publish that, too.

Messages have a *topic* and a *payload*. In a system for automating lights, the topic might be *bedroom\_light* and the payload *on* or *off*.

You can try this out by opening two Terminals at the same time. One Terminal will act as the publisher and the other as a subscriber. You can see this in action in [Figure 18-1](#).

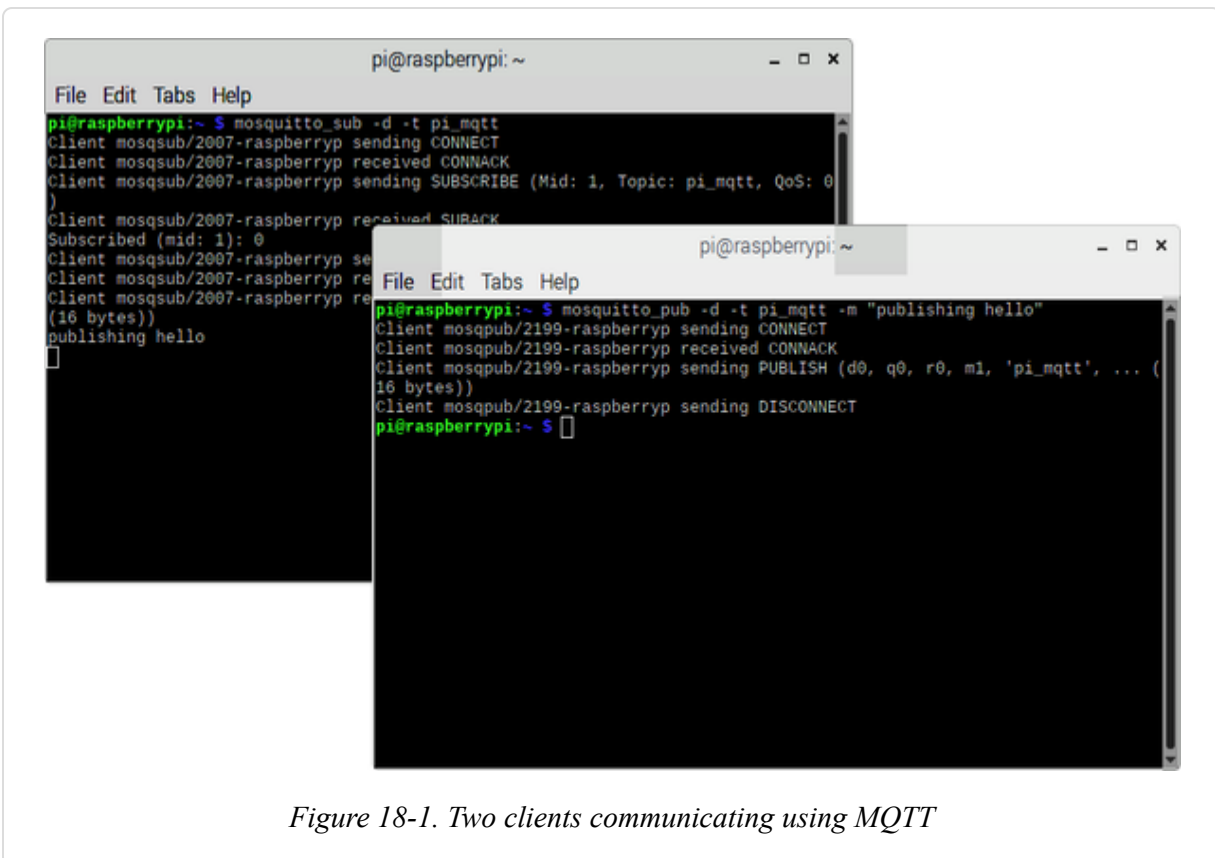


Figure 18-1. Two clients communicating using MQTT



Let's break this down. The Terminal on the left is the subscriber, and here we issue the command:

```
$ mosquitto_sub -d -t pi_mqtt
Client mosqsub/2007-raspberryp sending CONNECT
Client mosqsub/2007-raspberryp received CONNACK
Client mosqsub/2007-raspberryp sending SUBSCRIBE (Mid: 1, Topic:
pi_mqtt, QoS: 0)
Client mosqsub/2007-raspberryp received SUBACK
Subscribed (mid: 1): 0
```

The `mosquitto_sub` command subscribes the client. The `-d` option specifies debug mode, which just means you will see a lot more output about what the client and server are doing; this is useful while you are making sure everything is working. The `-t pi_mqtt` option specifies the topic that the client is interested in as `pi_mqtt`.

The debug trace shows that the client is connecting to the server without problems and that the client has requested to subscribe and the server has acknowledged the subscription.

Leave this Terminal session running and open a second Terminal session to act as a client that's going to publish something on the topic `pi_mqtt`. Enter the following command in this new Terminal window:

```
$ mosquitto_pub -d -t pi_mqtt -m "publishing hello"
Client mosqpub/2199-raspberryp sending CONNECT
Client mosqpub/2199-raspberryp received CONNACK
Client mosqpub/2199-raspberryp sending PUBLISH (d0, q0, r0, m1,
'pi_mqtt',
... (16 bytes))
Client mosqpub/2199-raspberryp sending DISCONNECT
```

Again, the `-d` and `-t` options specify debug mode and the topic, but this time there is an extra option of `-m`, which specifies a message to include in the publication. If the publisher were a sensor, the message might be the sensor reading.

As soon as the `mosquitto_pub` command is sent, text like the following will appear in the first Terminal window (the subscriber):

```
Client mosqsub/5170-raspberryp received PUBLISH (d0, q0, r0, m0,
'pi_mqtt',
... (16 bytes))
publishing hello
```

## See Also

For information on Mosquitto, see <https://mosquitto.org>.

For other MQTT recipes in this chapter, see Recipes [18.2](#) and [18.5](#).

## 18.2 Using Node-RED with an MQTT Server

### Problem

You want to combine Node-RED with an MQTT server to be able, for instance, to control a general-purpose input/output (GPIO) pin in response to MQTT message publications.

### Solution

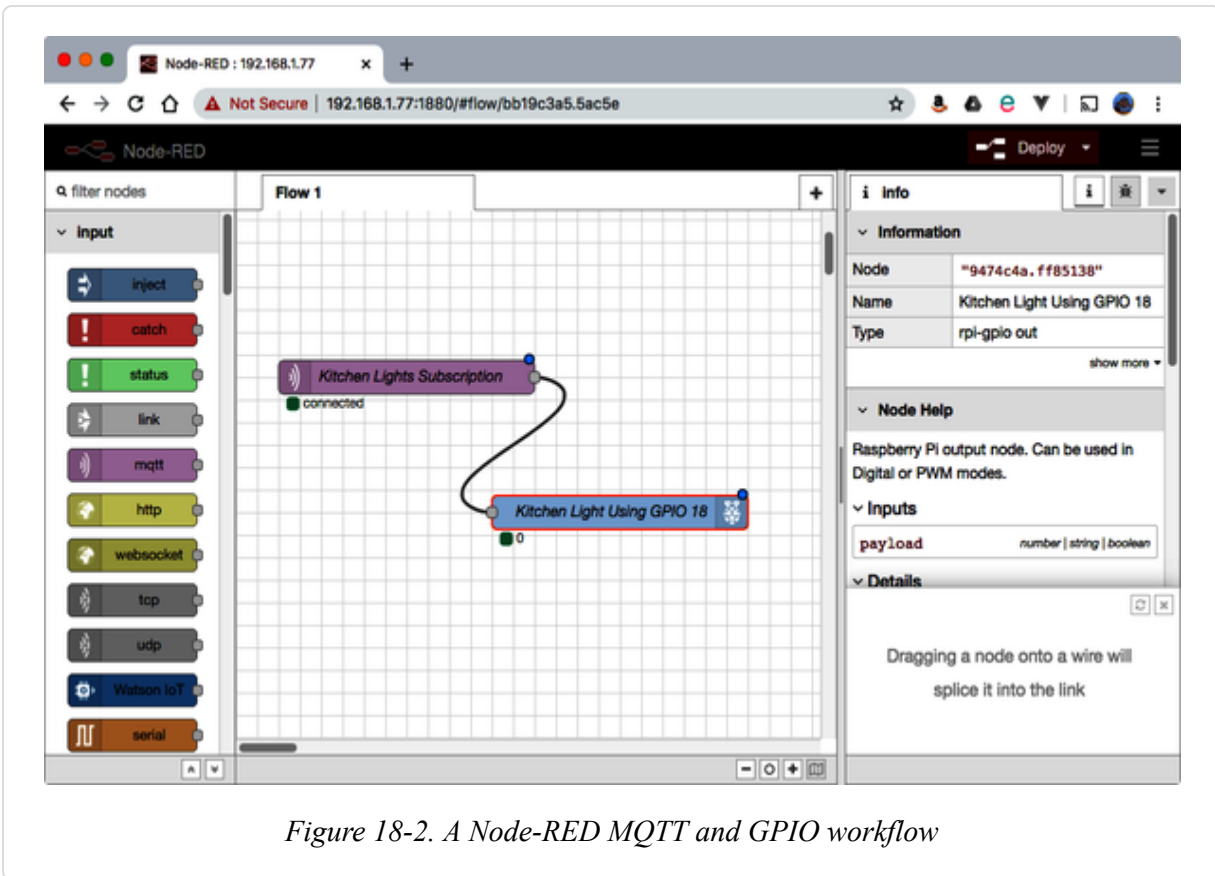
Use a Node-RED “mqtt” node and an “rpi gpio” node in a Node-RED flow like the one shown in [Figure 18-2](#).

If you have not installed Node-RED, you can find instructions in [Recipe 17.3](#).

After you deploy this, you will be able to turn GPIO 18 on and off by sending `mosquitto_pub` commands. For testing purposes, attach an LED or Raspberry Squid LED to pin 18 ([Recipe 11.1](#)).

The flow refers to kitchen lights, as this example pretends that GPIO 18 is being used with something like a PowerSwitch Tail ([Recipe 11.7](#)) to switch lighting on and off.

Let's build the example up one node at a time.



Start by adding an “mqtt” node from the “input” category of Node-RED. Double-click the node to edit it (Figure 18-3).

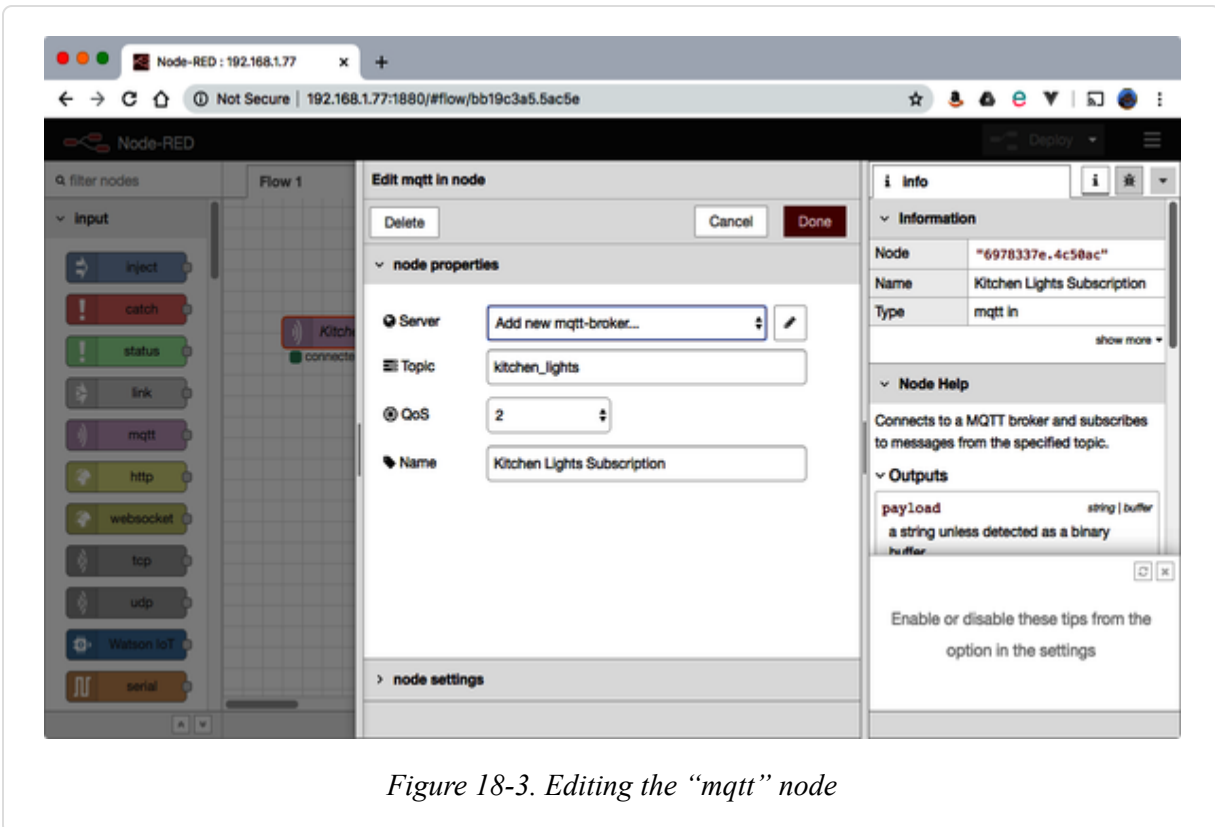


Figure 18-3. Editing the “mqtt” node

Notice that the Server field just has the option to “Add new mqtt-broker.” We will return to this in a moment. For now, specify a topic (`kitchen_lights`) and give the node a meaningful name. The QoS field allows you to set the quality of service and determines how persistent the MQTT server is at getting messages to their intended destination. Level 2 means guaranteed delivery.

We now need to define an MQTT server for Node-RED, so click the edit (pencil) button next to the Server field. This opens the window shown in [Figure 18-4](#).

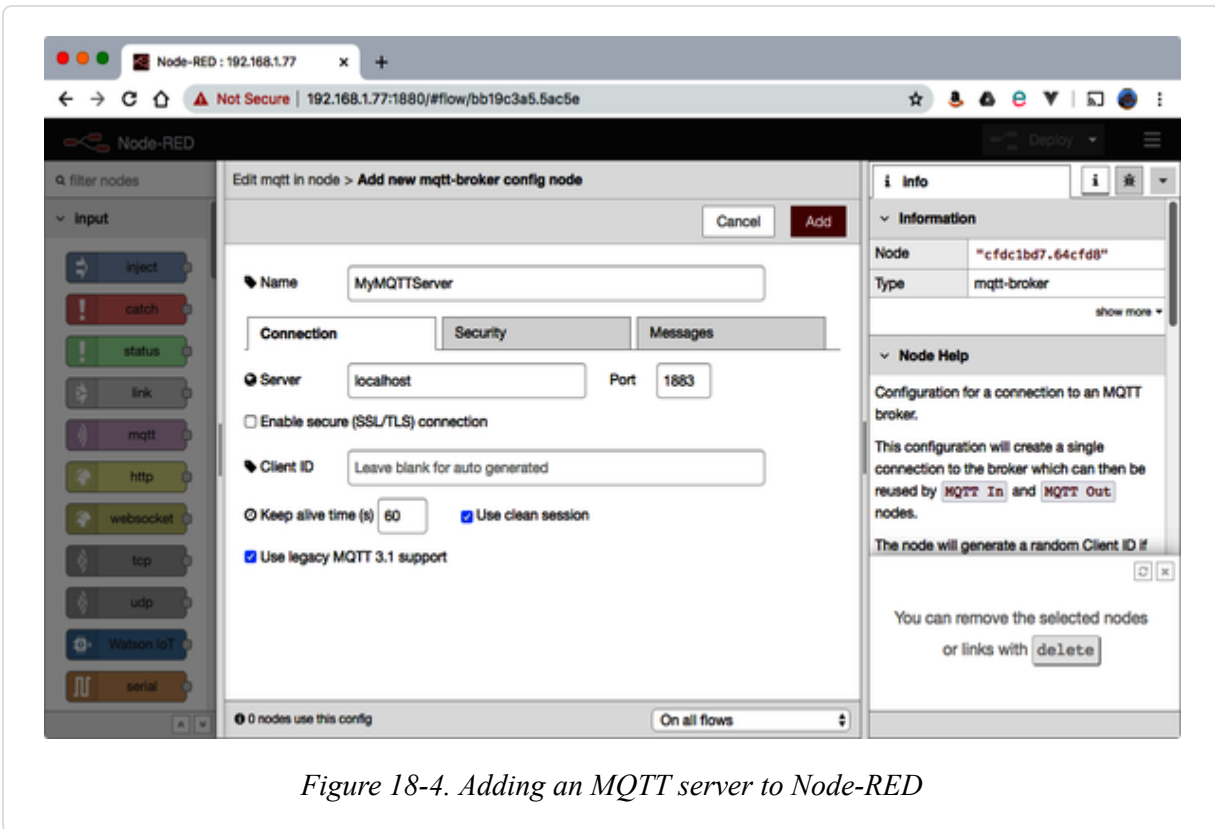


Figure 18-4. Adding an MQTT server to Node-RED

Give the server a name and enter `localhost` in the Server field. We can do this because Node-RED and the MQTT server are running on the same Raspberry Pi.

Now we can add the “rpi-gpio out” node. You will find this in the Raspberry Pi section. After you have added it to the flow, open it (Figure 18-5).

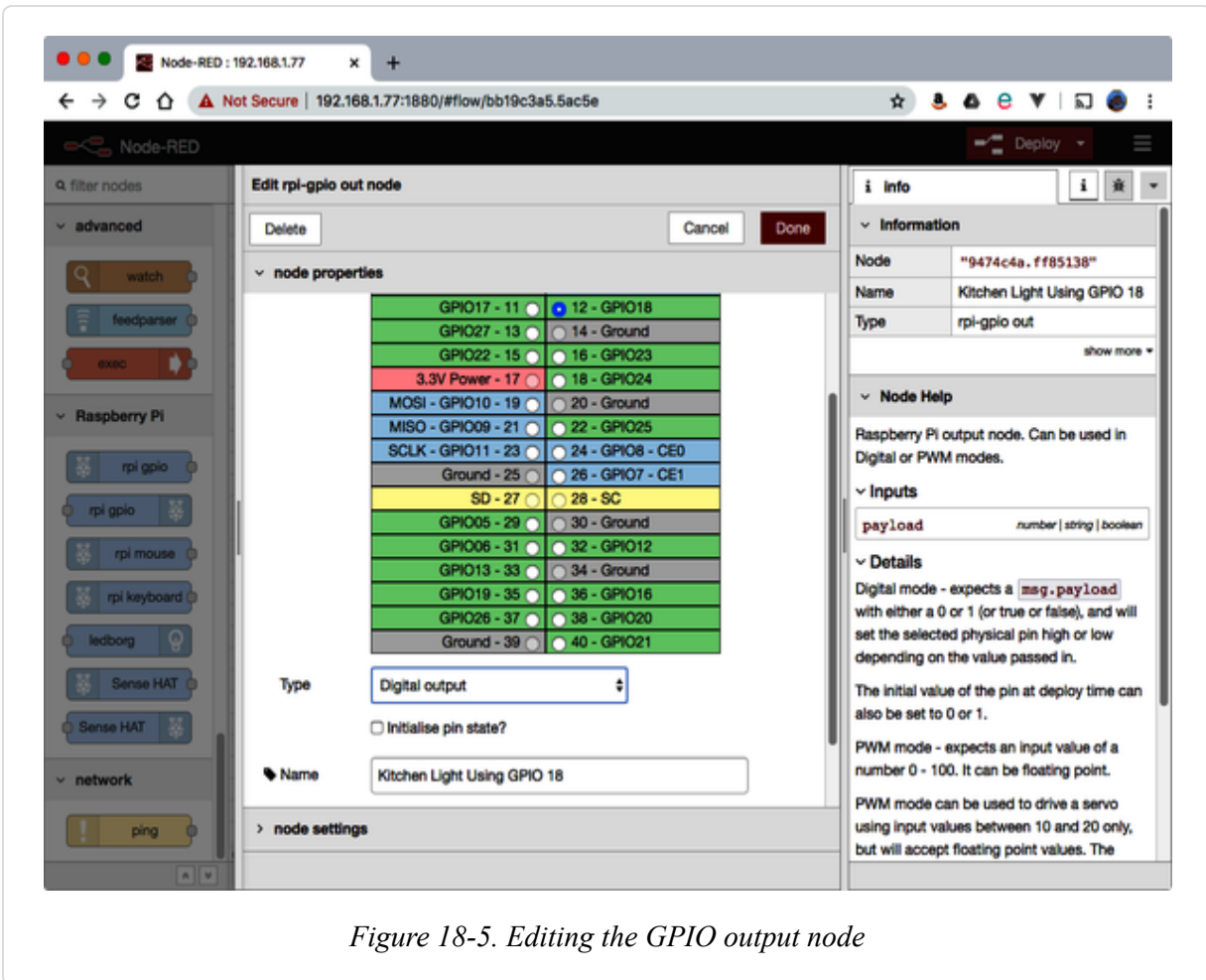


Figure 18-5. Editing the GPIO output node

Select “12 - GPIO 18” and give the node a name before clicking Done. Drag the connector from the “mqtt in” node to the Raspberry Pi GPIO node so that the flow looks like [Figure 18-2](#).

Click the Deploy button and then open a Terminal session on the Raspberry Pi to test the flow.

Type the following command in the Terminal window to publish a request to turn on the light:

```
$ mosquitto_pub -d -t kitchen_lights -m 1
```

The LED on pin 18 should light. Then, to turn off the LED, send:

```
$ mosquitto_pub -d -t kitchen_lights -m 0
```

## Discussion

This recipe works only if the Raspberry Pi happens to be right next to the thing you want to control. In reality, you are more likely to want to use a wireless switch.

It is, however, useful to know how to control GPIO pins through MQTT and Node-RED.

Node-RED has the ability to import and export flows as JSON text. All the flows used in this chapter are available on the book's [GitHub pages](#).

To import one of these flows into Node-RED, visit the GitHub page and then click the recipe number corresponding to the flow that you want to import. For example, in [Figure 18-6](#) I have clicked *recipe\_18\_10.json*.

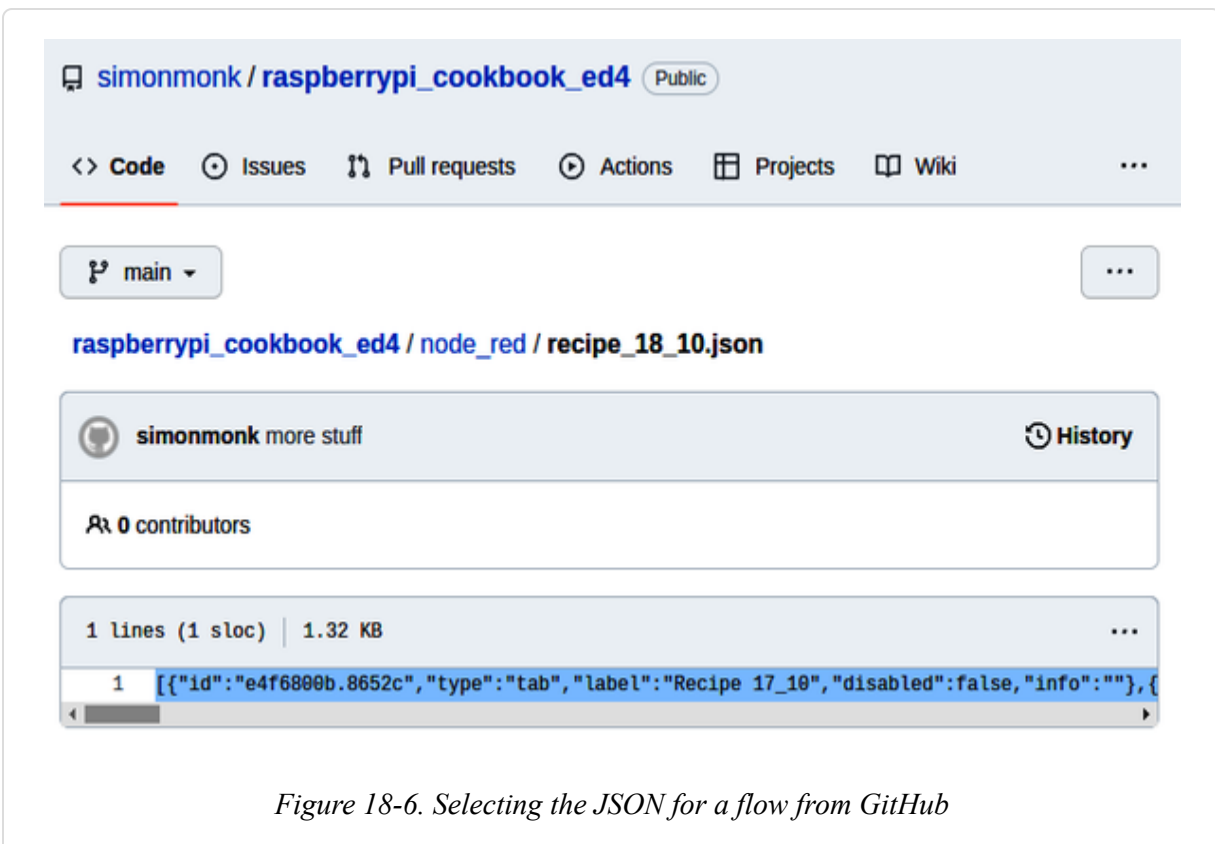
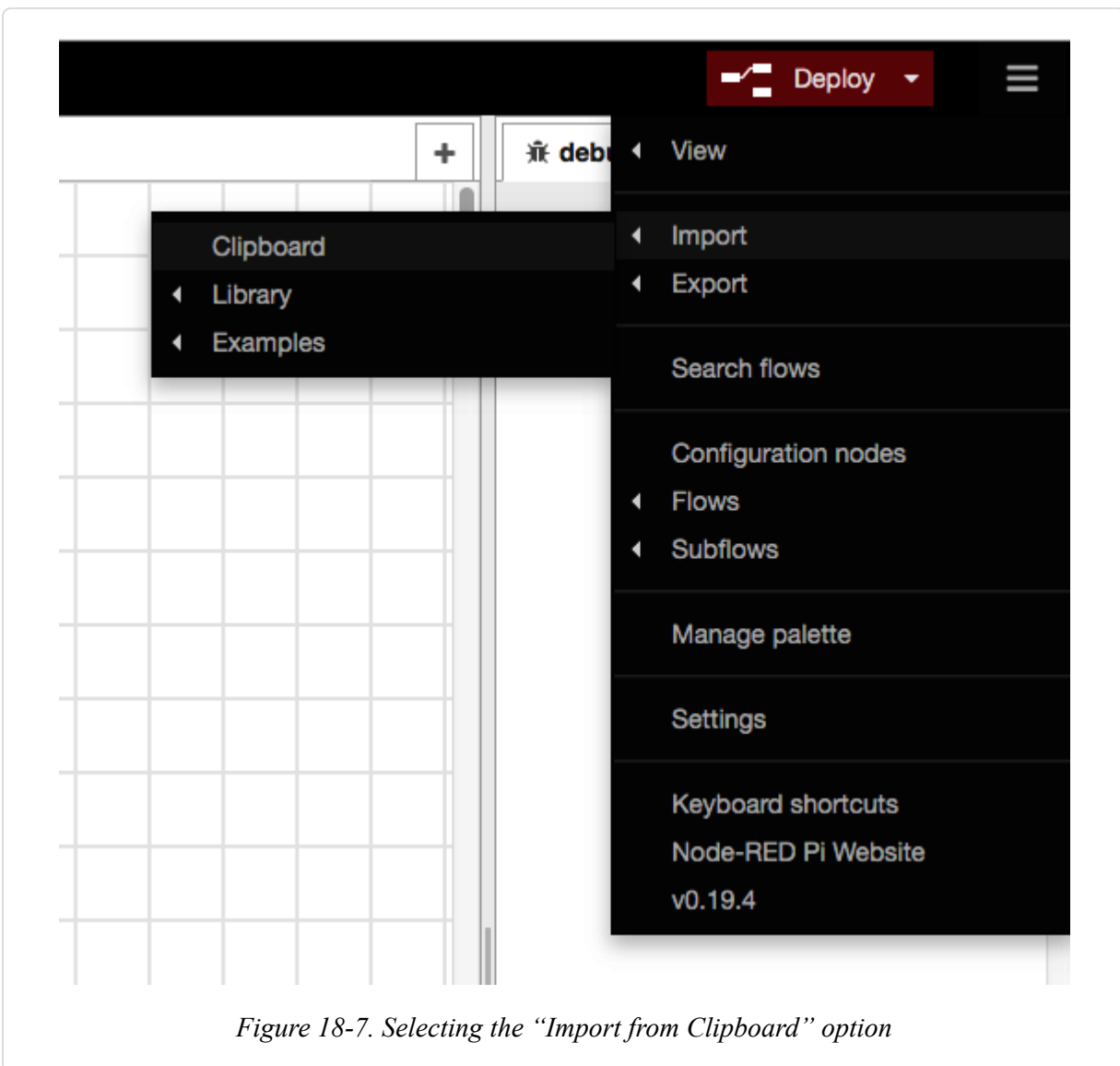


Figure 18-6. Selecting the JSON for a flow from GitHub

Select the entire line in the code area and copy it to your clipboard. (Clicking the Raw button just above the code can make it easier to select all of the code for copying.) Then switch back to your Node-RED web page and select Import, and then Clipboard from the Node-RED menu, as shown in [Figure 18-7](#).

Then paste the code you copied from GitHub into the window shown in [Figure 18-8](#) and select “new flow.”



*Figure 18-7. Selecting the “Import from Clipboard” option*



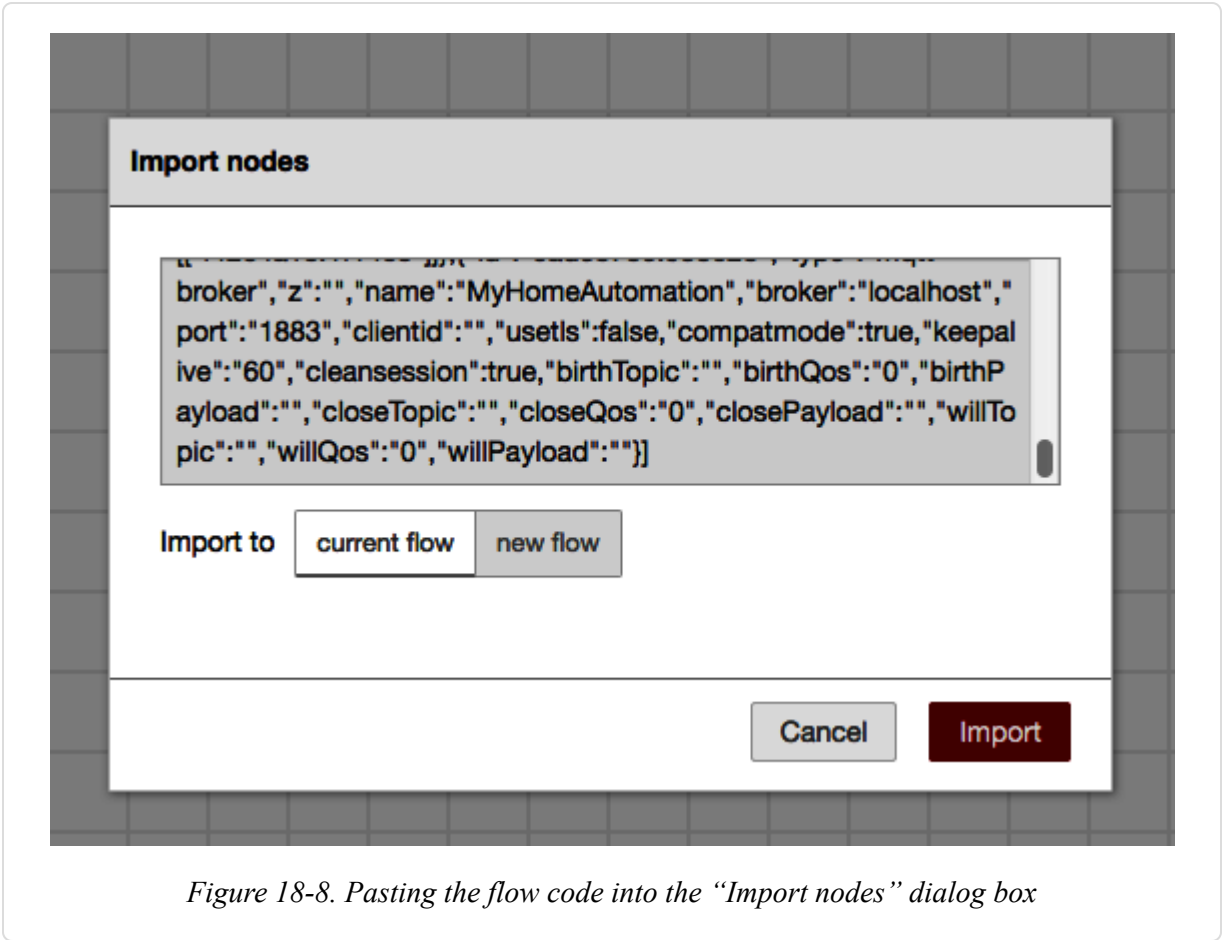


Figure 18-8. Pasting the flow code into the “Import nodes” dialog box

When you click Import, the flow appears in a new tab (Figure 18-9).

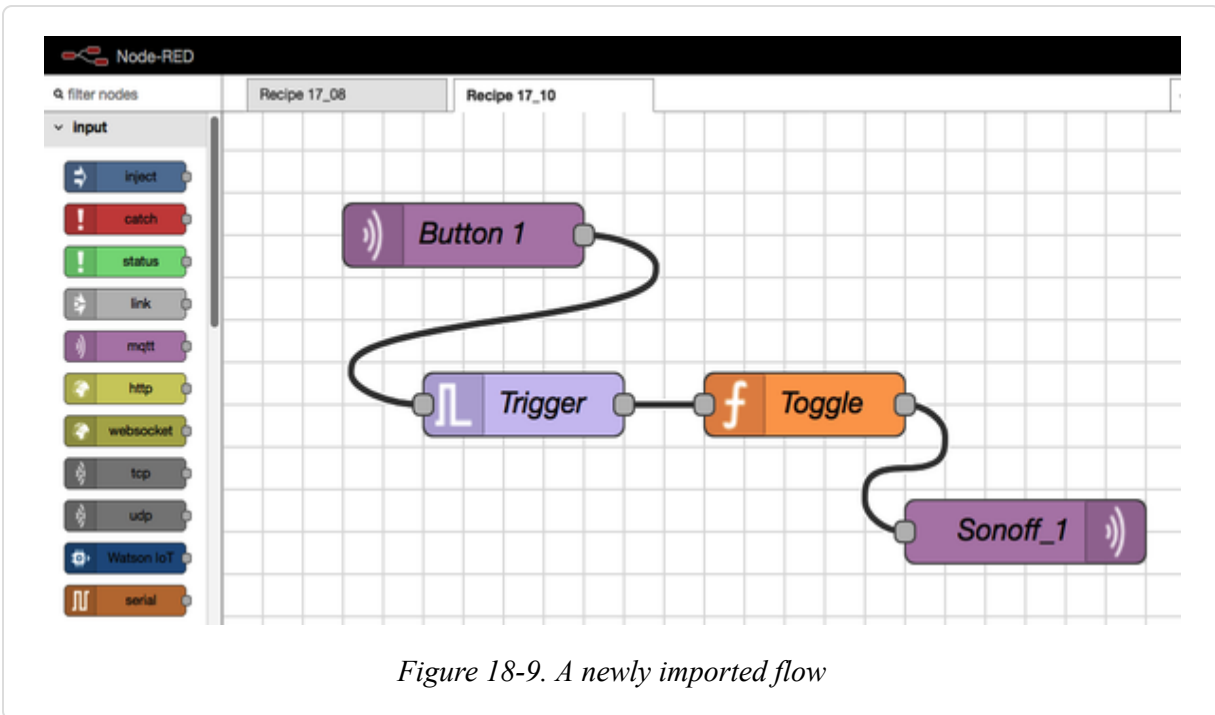


Figure 18-9. A newly imported flow

## See Also

To see how you can control a WiFi switch using MQTT, see [Recipe 18.5](#).  
To see a similar recipe that uses Node-RED instead, see [Recipe 18.6](#).

Read more about [MQTT QoS levels](#).

For more information, see the [full documentation on Node-RED](#).

## 18.3 Flashing a Sonoff WiFi Smart Switch for MQTT Use

### Problem

You want to use a WiFi smart switch with MQTT so that you can control domestic accessories directly from your Raspberry Pi.

### Solution

Flash (install) new firmware (Tasmota) onto a low-cost Sonoff WiFi switch, configure the switch through a web interface, and then control it using MQTT.

The Sonoff web switches (shown in [Figure 18-10](#)) offer an extremely low-cost way of turning lighting and other appliances on and off wirelessly.



*Figure 18-10. The Sonoff WiFi switch*

However, the firmware pre-installed on the Sonoff switches is proprietary and relies on servers in China for communication to the internet. If you would prefer to have local control of your device and actually improve over the original firmware, you should follow this recipe to flash new open source firmware onto your Sonoff.

You can do all of this from your Raspberry Pi, but you'll need a few things:

- A Sonoff Basic web switch (see [“Modules”](#))
- A row of four header pins (see [“Miscellaneous”](#))

- Four female-to-female jumper wires (see “Prototyping Equipment and Kits”)
- Soldering equipment and solder (see “Prototyping Equipment and Kits”)

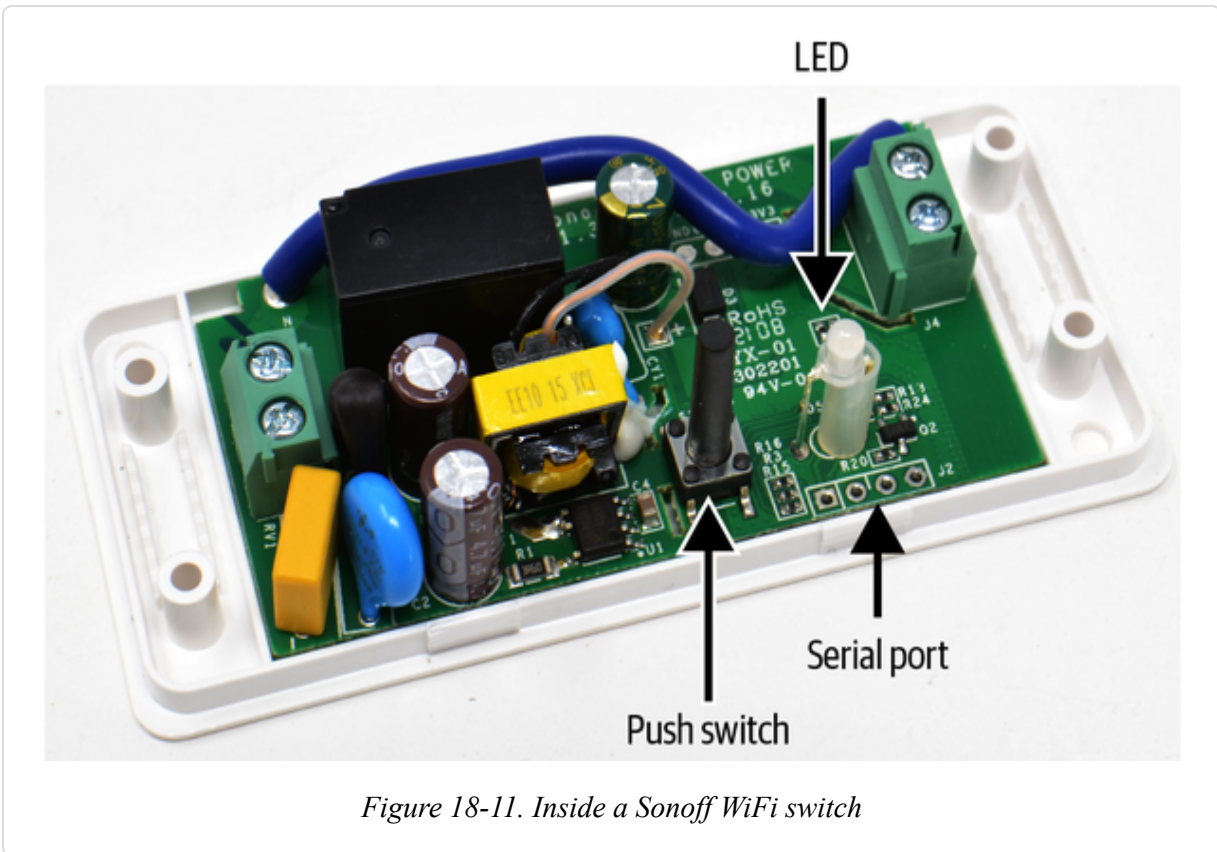
You will also need a Raspberry Pi 2 or later because earlier Raspberry Pis are not able to provide enough current at 3.3V to power the Sonoff.

### **Danger: High Voltage**

Switching alternating current (AC) using a Sonoff requires the connection of live wires to the Sonoff’s screw terminals. This is electrician’s work and should be done only by someone qualified to do so.

Flash the Sonoff fresh out of its box, *before* it is wired into your household electricity. You can configure it without connecting it to an AC supply. The Raspberry Pi will power it.

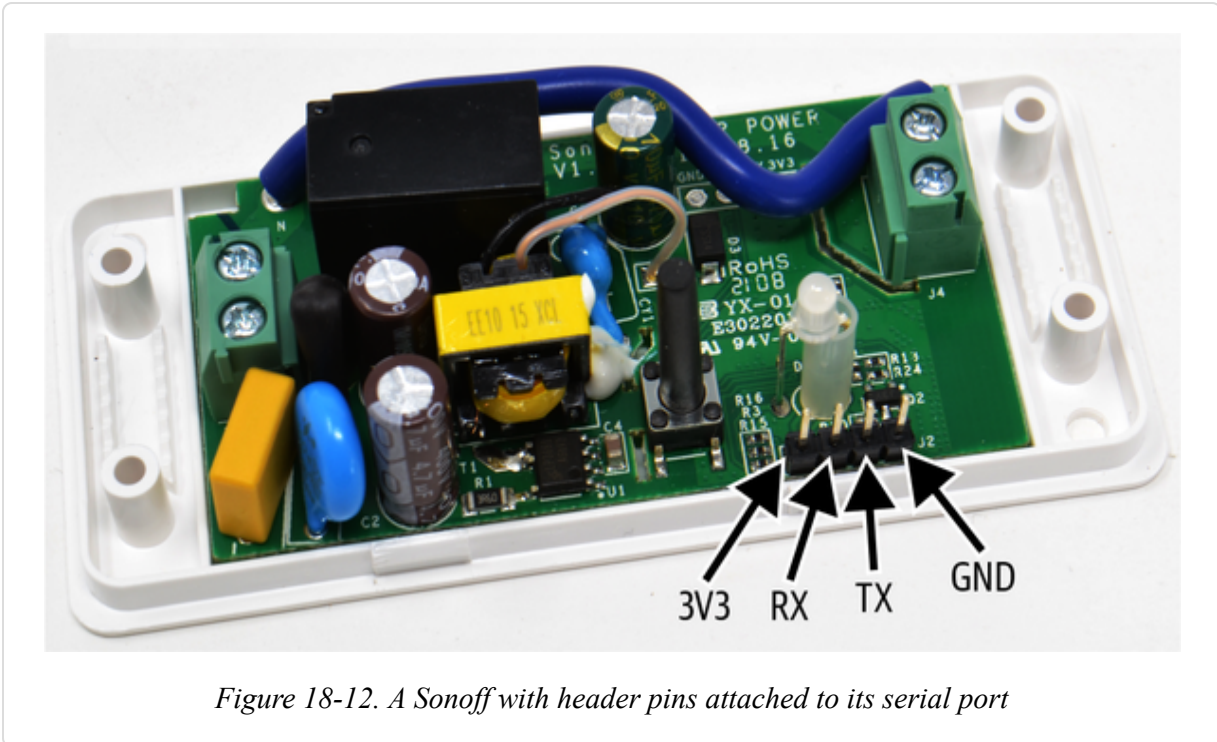
Before you connect your Sonoff to AC, take it apart, because you are going to need to solder a strip of four header pins into the holes supplied on the Sonoff’s circuit board. **Figure 18-11** shows the position of the holes and also marks off the roles of the four header pins we are interested in. In fact, they are a serial port of the Sonoff.



*Figure 18-11. Inside a Sonoff WiFi switch*

Note that different versions of the Sonoff have the serial interface in slightly different places. You may have to look on the underside of the board for pads marked 3V, RX, TX, and GND to find the serial interface.

Once the header pins are soldered in place, your Sonoff will look something like [Figure 18-12](#).



*Figure 18-12. A Sonoff with header pins attached to its serial port*

You now need to connect the header pins on the Sonoff to the GPIO pins on your Raspberry Pi as follows (use [Figure 18-13](#) as a reference).

- Sonoff 3.3V to Raspberry Pi 3.3V
- Sonoff RX to Raspberry Pi TXD
- Sonoff TX to Raspberry Pi RXD
- Sonoff GND to Raspberry Pi GND

[Figure 18-13](#) shows a Raspberry Pi 400 connected to the Sonoff using a GPIO adapter.

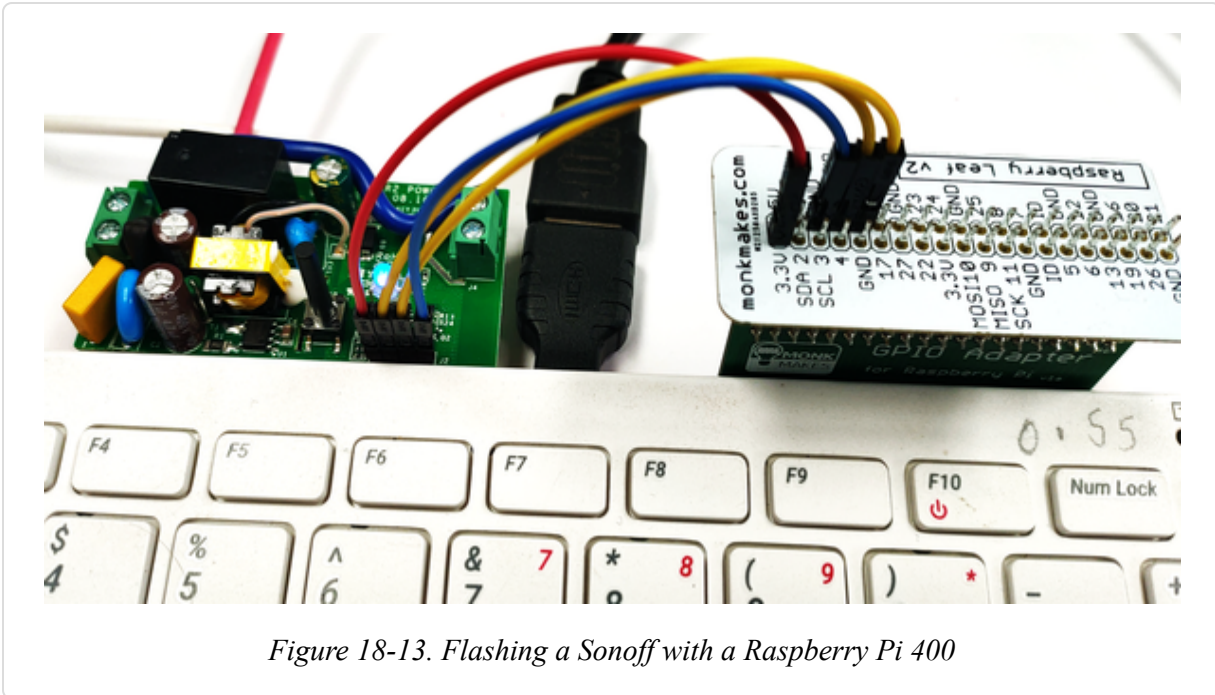


Figure 18-13. Flashing a Sonoff with a Raspberry Pi 400

Changing the firmware (*flashing*) the Sonoff requires a piece of Python software called `esptool`. To download it onto your Raspberry Pi, run the following command:

```
$ git clone https://github.com/espressif/esptool.git
$ cd esptool
```

Having downloaded the software, we also need to get the replacement Tasmota firmware to flash onto the Sonoff. To do this, run the following command from within the *esptool* directory.

```
$ wget https://github.com/arendst/Sonoff-
Tasmota/releases/download/v6.6.0/
sonoff-basic.bin
```

This will fetch a file called *sonoff-basic.bin* into the *esptools* directory.

The remainder of the process is as follows:

1. Put the Sonoff into *flash* mode.

At the moment, the LED on your Sonoff will probably be happily flashing away. To put the Sonoff into flash mode, disconnect the GND lead, press the Sonoff's push switch ([Figure 18-11](#)) and reconnect the GND lead to power up the Sonoff from your Raspberry Pi. Once the Sonoff is powered up, release the push switch after a few seconds.

The Sonoff's LED should no longer be blinking. It's now in flash mode, ready to receive a new program.

## 2. Erase the Sonoff.

Make sure that you are still in the *esptools* directory and run the command to erase the Sonoff. You should see messages in the Terminal like those shown here:

```
$ cd ~/esptools
$ python3 esptool.py --port /dev/serial0 erase_flash
esptool.py v4.3-dev
Serial port /dev/serial0
Connecting...
Failed to get PID of a device on /dev/ttyS0, using standard
reset
sequence.
.....
Detecting chip type... Unsupported detection protocol,
switching and
trying again...
Connecting...
Failed to get PID of a device on /dev/ttyS0, using standard
reset
sequence.

Detecting chip type... ESP8266
Chip is ESP8285N08
Features: WiFi, Embedded Flash
Crystal is 26MHz
MAC: c4:4f:33:eb:0f:73
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 1.4s
Hard resetting via RTS pin...
```



### 3. Flash the Tasmota software onto the Sonoff.

Run the command to flash the *sonoff-basic.bin* file that you fetched earlier onto the Sonoff:

```
$ python3 esptool.py --port /dev/serial0 write_flash -fs 1MB
-fm dout
    0x0 sonoff-basic.bin
esptool.py v4.3-dev
Serial port /dev/serial0
Connecting...
Failed to get PID of a device on /dev/ttyS0, using standard
reset
    sequence.
.....
Detecting chip type... Unsupported detection protocol,
switching and
    trying again...
Connecting...
Failed to get PID of a device on /dev/ttyS0, using standard
reset
    sequence.

Detecting chip type... ESP8266
Chip is ESP8285N08
Features: WiFi, Embedded Flash
Crystal is 26MHz
MAC: c4:4f:33:eb:0f:73
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00000000 to 0x00069fff...
Compressed 432432 bytes to 300963...
Wrote 432432 bytes (300963 compressed) at 0x00000000 in 27.0
seconds
    (effective 128.0 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

## Discussion

When all of this is complete, you can disconnect the jumper wires and get a qualified person to install your Sonoff so that it is supplied from AC and

ready to switch whatever you have in mind for it to switch.

However, it might be wise to do a bit more testing of the newly flashed Sonoff before connecting it to the AC or putting it somewhere inaccessible. So you can, if you prefer, continue to power the Sonoff from your Raspberry Pi by leaving the 3.3V and GND jumpers in place. The LED will light when the Sonoff is switched on.

In addition to the Sonoff model that I used here, many other models are available, including some that look like regular light switches but contain a WiFi module.

## See Also

For more information on Tasmota, see <https://oreil.ly/aevZD>.

Having flashed your Sonoff, follow the next recipe ([Recipe 18.4](#)) to configure it.

## 18.4 Configuring a Sonoff WiFi Smart Switch

### Problem

You need to join your Sonoff WiFi switch to your home WiFi network.

### Solution

First, flash the Tasmota firmware onto your Sonoff using [Recipe 18.4](#). If you have the Tasmota firmware on your Sonoff and it's powered up, either using the 3.3V supply of your Raspberry Pi (Pi 2 or later) or in situ with an AC supply, you can now configure the Sonoff by connecting to the wireless access point that it will be running. At the time of writing, you won't be able to do this with your Raspberry Pi because, after you join it, the wireless access point does not trigger the welcome page for the access point to be opened in the same way as it does if you connect using a Mac or Windows PC. Instead, you'll need to connect to the WiFi access point

called something like Sonoff-2500 on your PC or Mac, or even on your smartphone (Figure 18-14).

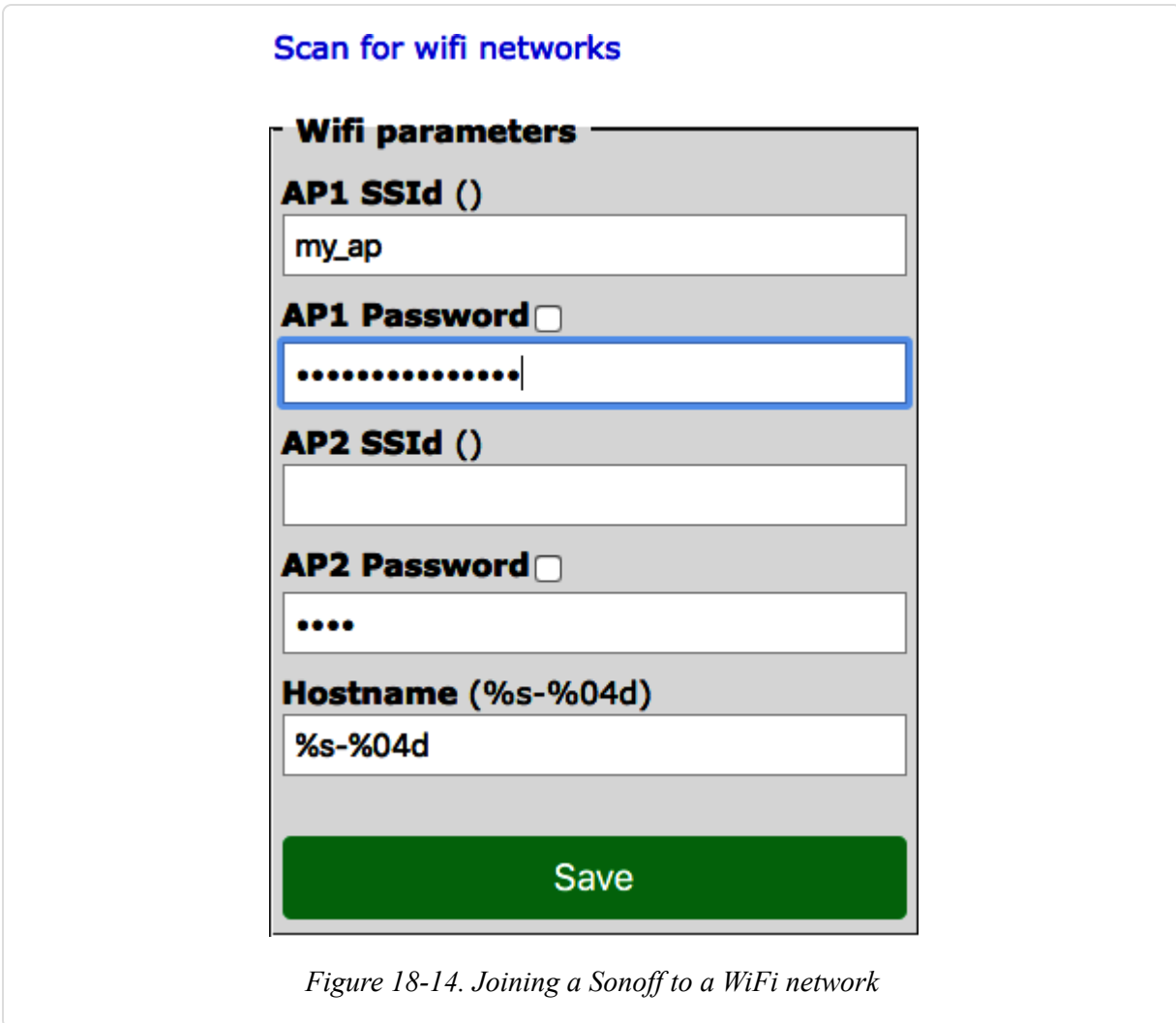


Figure 18-14. Joining a Sonoff to a WiFi network

You have the option to enter credentials for two wireless access points. But assuming you have just one, either use the “Scan for wifi networks” link at the top of the page or else just enter your access point name in the AP1 SSID field and your password in the AP1 Password field and then click Save.

The Sonoff will reboot and, if you entered the access point credentials correctly, it will reboot connecting to your network.

Now you have the problem of finding the Sonoff’s IP address. A tool like Fing for Android phones or Discovery for iOS will do this. As you can see

from **Figure 18-15**, in my case the Sonoff has been assigned the IP address 192.168.1.84.

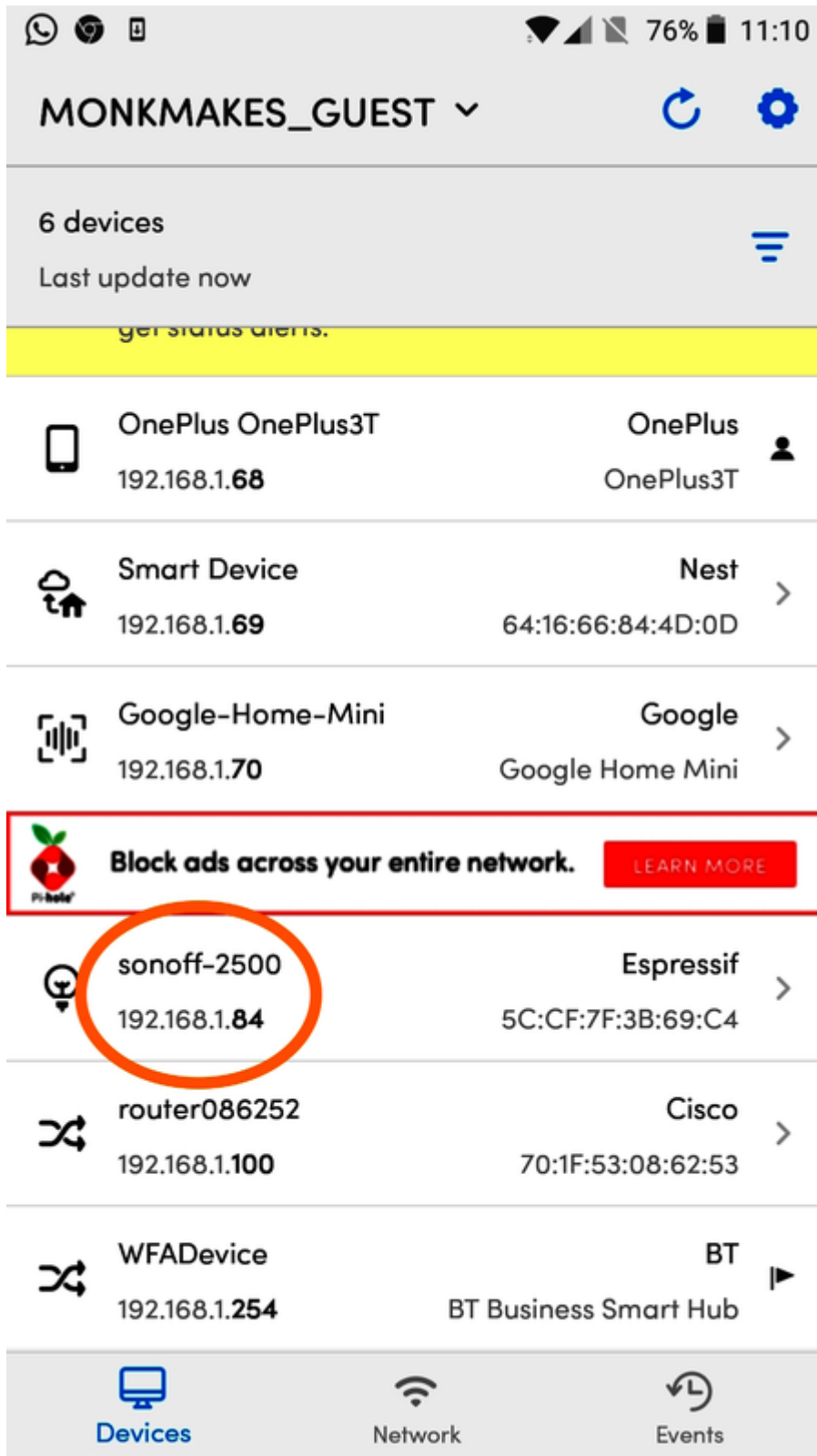
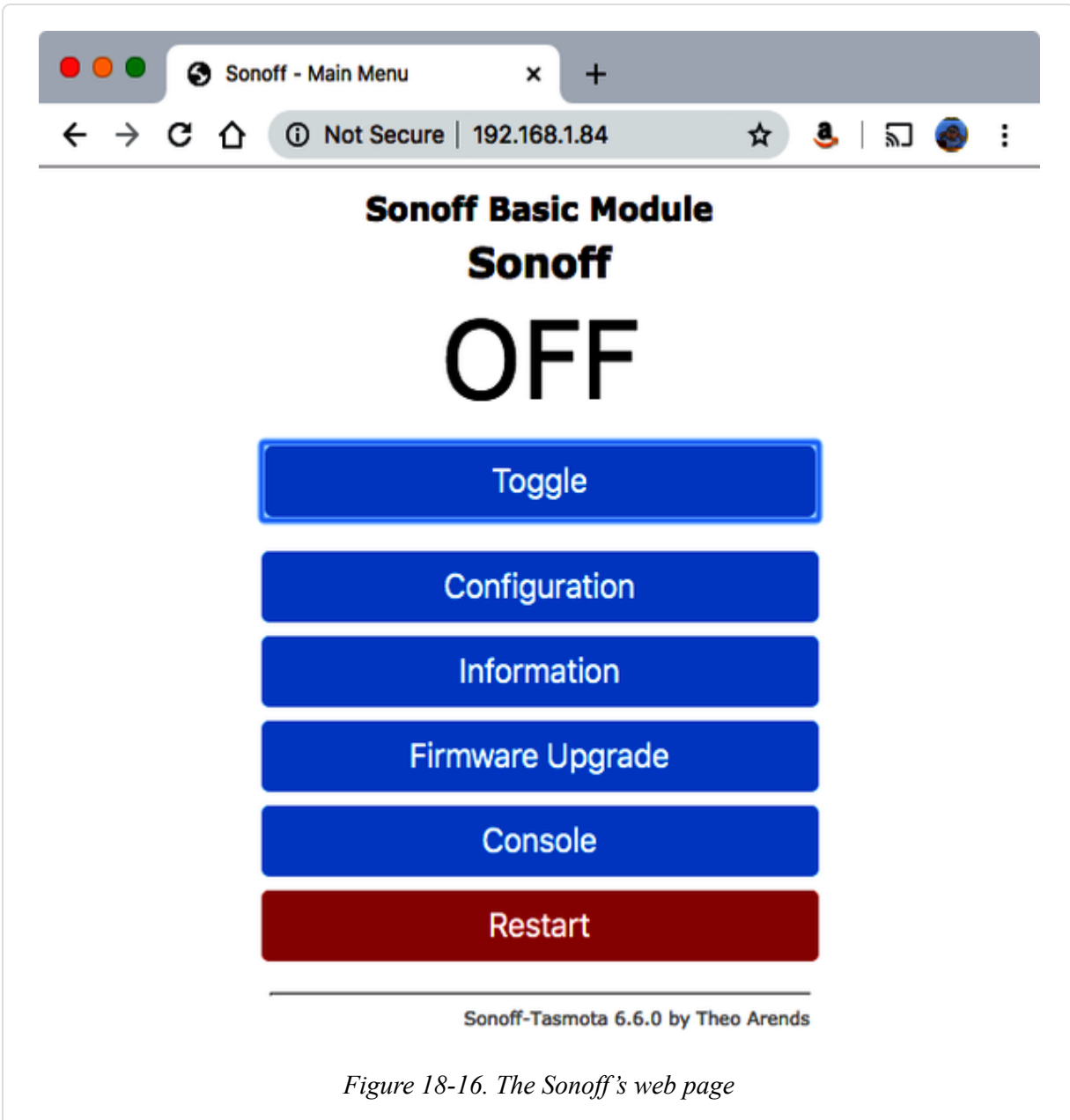


Figure 18-15. Finding the Sonoff's IP address

## Discussion

Now that the Sonoff is connected to your network, it will change mode, and rather than run an entire access point, it will instead run a web server on your network from which you can manage the device. To connect to this web page, enter the IP address of the Sonoff into a browser on any machine connected to the network. You should see something like [Figure 18-16](#).



*Figure 18-16. The Sonoff's web page*

Click the Toggle button to turn the Sonoff's LED on and off. If the Sonoff is actually wired into your house, rather than being powered from your Raspberry Pi, it will turn whatever it was connected to on and off.

## See Also

To see how you can configure these switches to work with MQTT and Node-RED, see [Recipe 18.5](#).

# 18.5 Using Sonoff Web Switches with MQTT

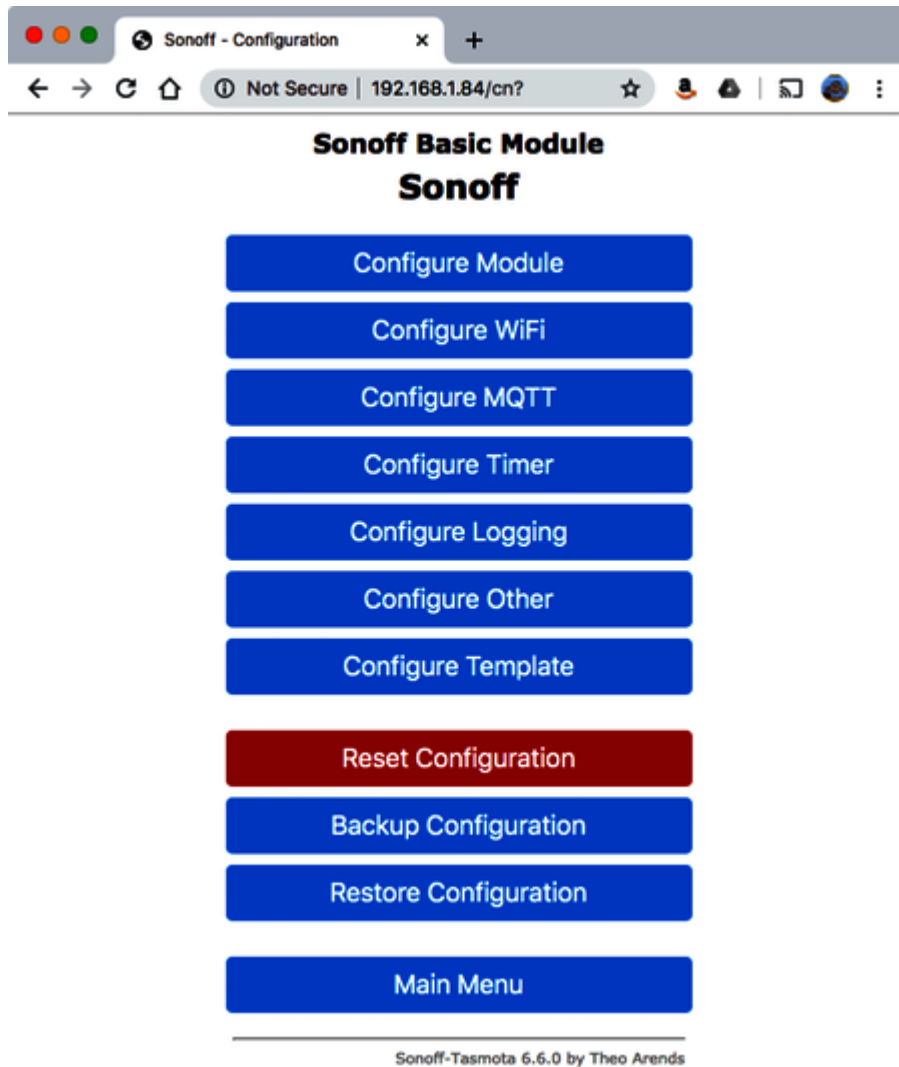
## Problem

You want to be able to control your newly flashed Sonoff web switch using MQTT.

## Solution

First make sure that you have followed [Recipes 18.3](#) and [18.4](#) to flash new firmware onto your Sonoff device and configure it to connect to your WiFi network.

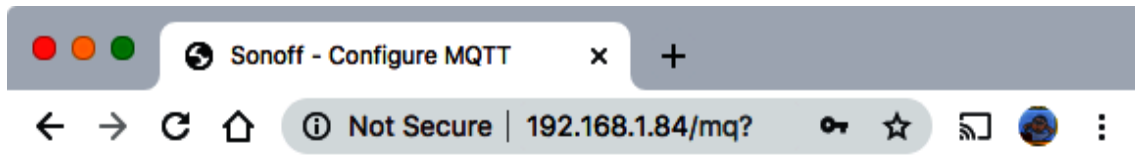
To control the Sonoff switch using MQTT, you need to configure the Sonoff using its web interface. Enter the IP address of your Sonoff (see [Recipe 18.4](#)) into your browser and click the Configuration button. This opens the menu shown in [Figure 18-17](#).



*Figure 18-17. The Sonoff Tasmota configuration menu*

Click the **Configure MQTT** option to open the MQTT configuration page, shown in **Figure 18-18**.





## Sonoff Basic Module Sonoff

**MQTT parameters**

**Host ( )**  
192.168.1.77

**Port (1883)**  
1883

**Client (DVES\_3B69C4)**  
sonoff\_1

**User (DVES\_USER)**  
sonoff

**Password**   
....

**Topic = %topic% (sonoff)**  
sonoff\_1

**Full Topic (%prefix%/topic%/)**  
%prefix%/topic%/

**Save**

**Configuration**

Sonoff-Tasmota 6.6.0 by Theo Arends

Figure 18-18. The Sonoff Tasmota MQTT configuration menu

This is where we configure the Sonoff as a client to an MQTT server (see [Recipe 18.1](#)) and specify how it will subscribe, so that when we publish a command (say, to turn on), it understands the command.

To do this, you need to change some of the fields in the configuration form:

- Change the Host field to the IP address of your Raspberry Pi running the MQTT server.
- Change the Client field to “sonoff\_1”. We’ve added a “\_1” in case we end up with multiple Sonoff devices that we need to distinguish. You can also use a more meaningful name here if you like—perhaps “bedroom\_1\_sonoff” if that’s where the Sonoff is going to be installed.
- The User and Password fields are not used because our MQTT server doesn’t have any security configured. This is not as reckless as it sounds, as no one can do anything unless they are already inside your network. So it doesn’t matter what you put in these fields.
- Change the Topic to “sonoff\_1” again because you might end up with multiple Sonoff switches.
- Leave the Full Topic field unchanged.

Click Save, and the Sonoff will reboot itself for the changes to take effect.

## Discussion

You can test this MQTT interface from a Terminal. Enter the following command, and the Sonoff’s LED should light:

```
$ mosquitto_pub -t cmdnd/sonoff_1/power -m 1
```

Enter this command to turn the switch off again:

```
$ mosquitto_pub -t cmdnd/sonoff_1/power -m 0
```

If this doesn't work, add the `-d` option to the commands to check that the Mosquitto client commands are connecting to the MQTT server.

## See Also

We build on the work of this recipe in [Recipe 18.6](#) to control the switch using Node-RED.

# 18.6 Using Flashed Sonoff Switches with Node-RED

## Problem

You want to use a flashed Sonoff web switch with Node-RED.

## Solution

Follow [Recipe 18.5](#) to get your Sonoff working with MQTT, and then use a Node-RED MQTT node in a flow like that shown in [Figure 18-19](#).

If you want, you can [import the flow](#) rather than build it up from scratch. Follow the instructions in the Discussion section of [Recipe 18.2](#) to import the flow.

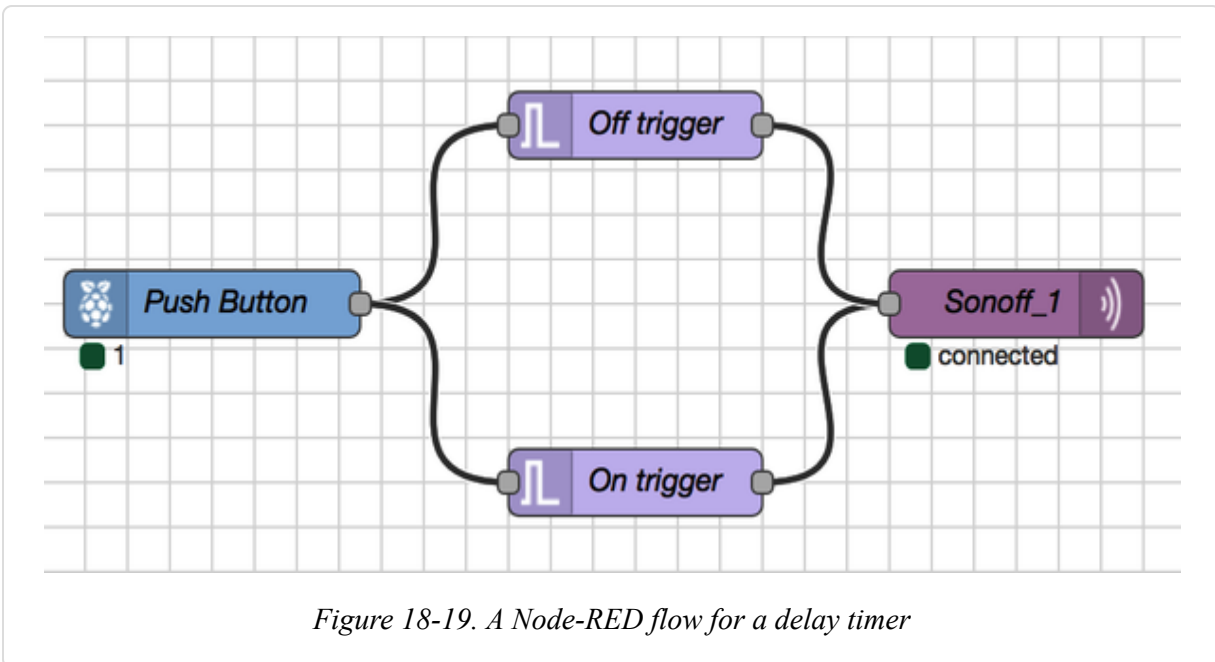


Figure 18-19. A Node-RED flow for a delay timer

This flow assumes that a push button is attached to the Raspberry Pi’s GPIO 25, and that, when it’s pressed, the Sonoff will be switched on for 10 seconds before being switched off again.

The push button is set up the same way as the button we used in [Recipe 17.3](#) and needs a hardware switch connected to GPIO 25 (see [Recipe 13.1](#)).

You will find the Trigger nodes in the Function section of Node-RED. We need two of these, so drag them out to the flow. [Figure 18-20](#) shows the settings for the “On trigger” node.

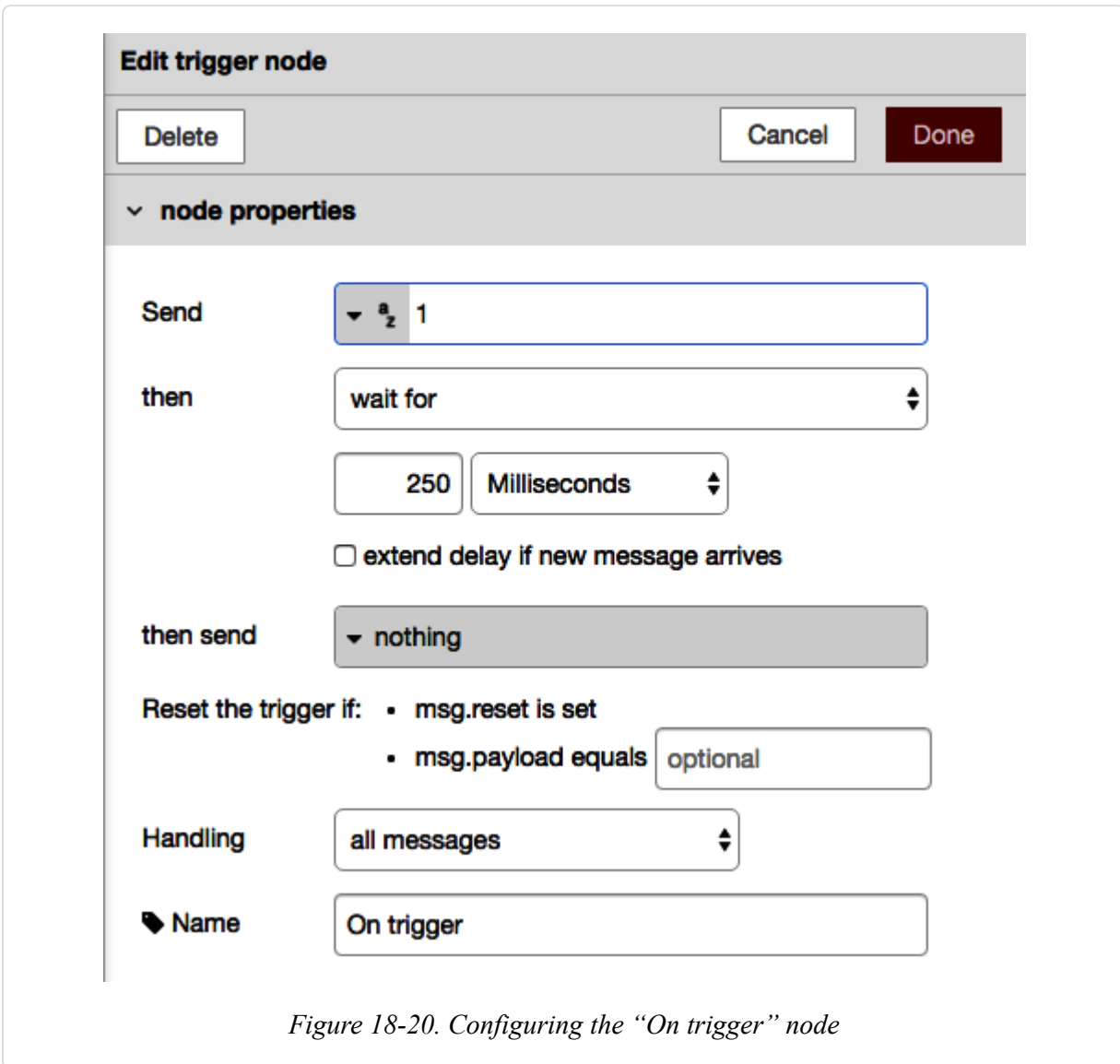


Figure 18-20. Configuring the “On trigger” node

This Trigger node is configured to send a 1 when triggered, wait for a quarter of a second (for debouncing), and then do nothing further. It’s given the name “On trigger.”

The “Off trigger” is different from the “On trigger” because we need it to delay for 10 seconds before sending a 0 to the Sonoff. [Figure 18-21](#) shows the settings for this.

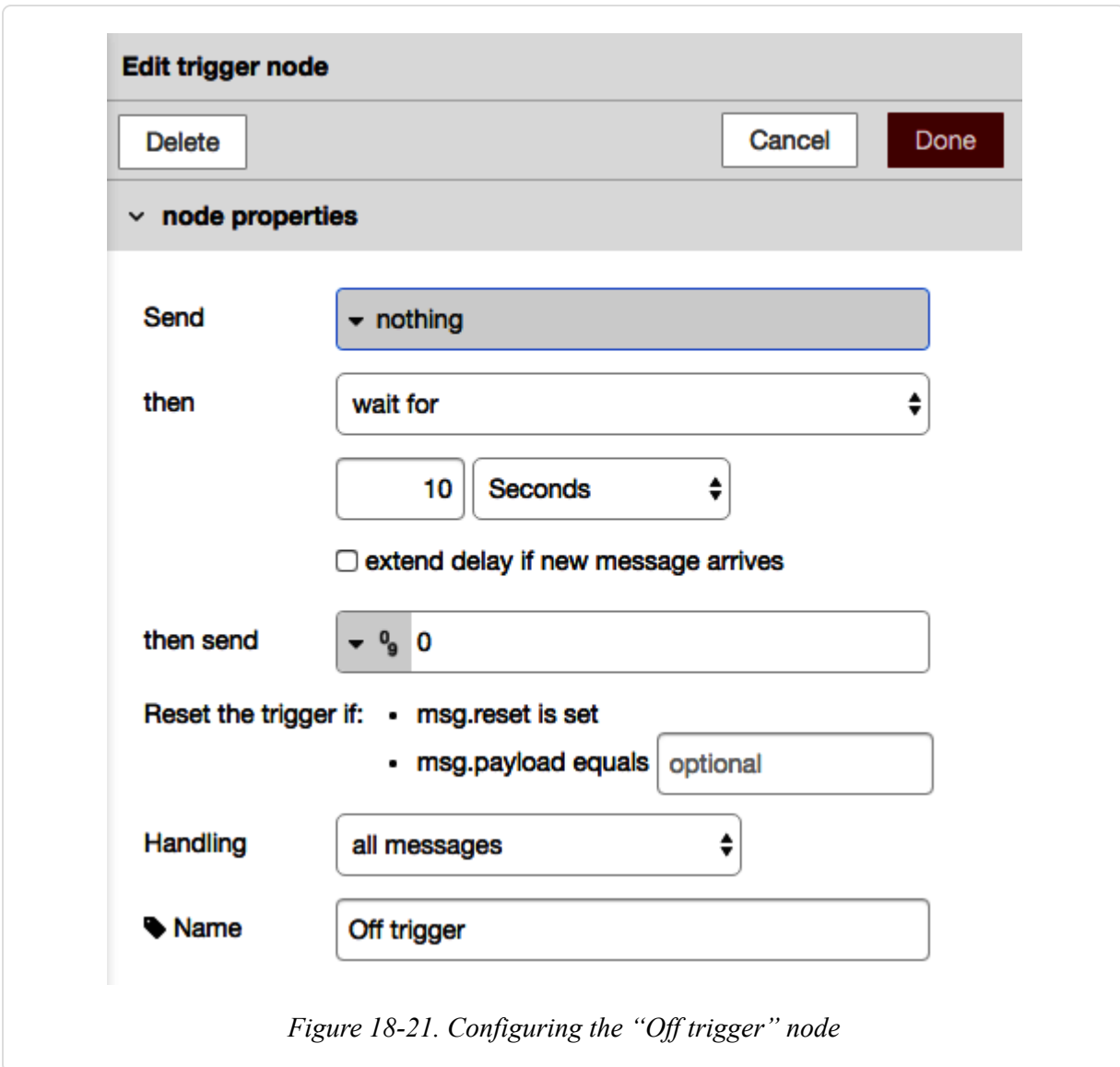


Figure 18-21. Configuring the “Off trigger” node

Finally, add an “mqtt” node from the Output section. Open this to configure it (Figure 18-22).

When clicked on, the Server field will prompt you to add a new MQTT server and enter its details, including its name (I used MyHomeAutomation), IP address (localhost), and port (1880).

Change the Topic and Name fields, as shown in Figure 18-22. You can now connect everything as shown in Figure 18-19 and deploy the flow.

When you press the button, the Sonoff should turn on and then turn itself off again after 10 seconds.

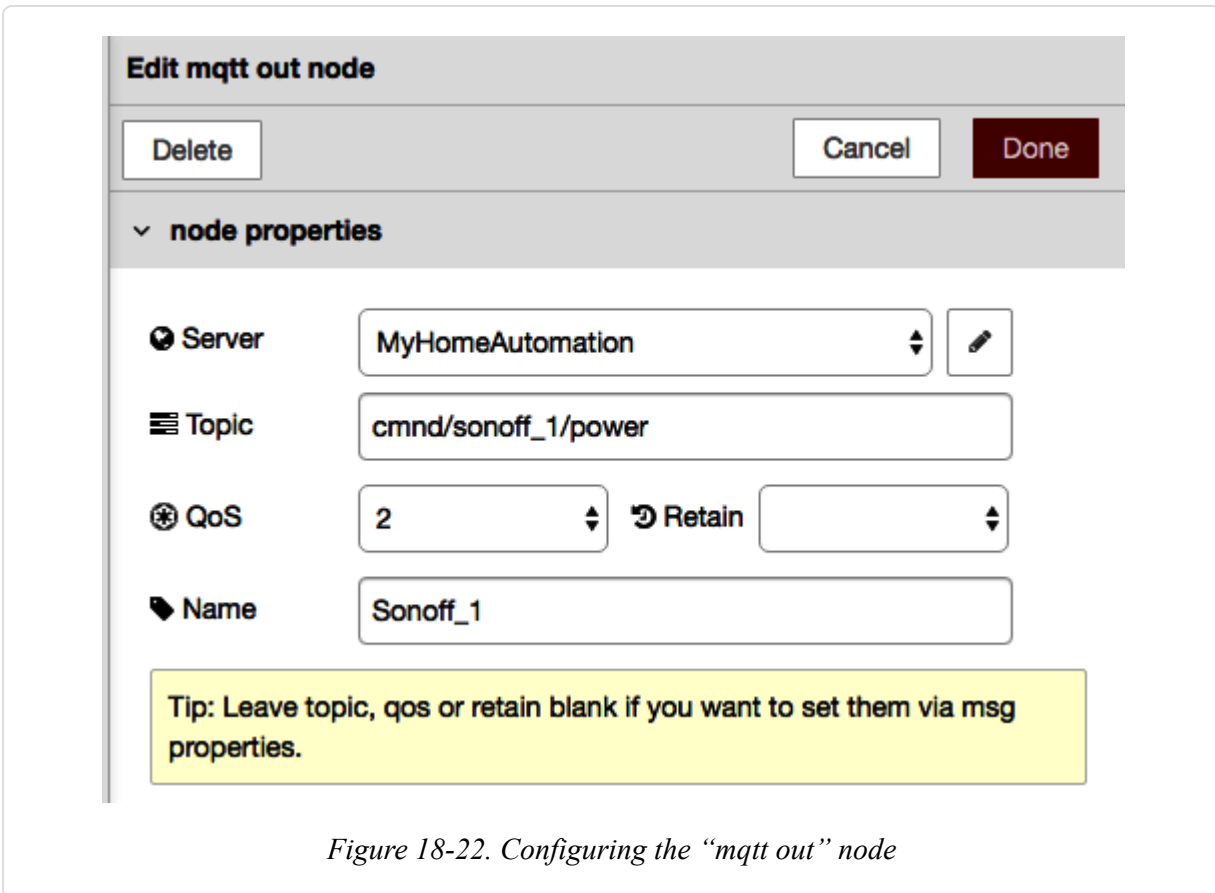


Figure 18-22. Configuring the “mqtt out” node

## Discussion

This recipe shows just how far you can go with Node-RED without having to write any actual code. By thinking of the automation as a flow of messages, Node-RED provides a really nice way of programming.

To use a motion sensor to switch the lights on for a predetermined period of time, you could replace the switch with a passive infrared (PIR) motion sensor (see [Recipe 13.9](#)).

## See Also

More information is available in the full [Node-RED documentation](#).

## 18.7 Turning Things On and Off Using the Node-RED Dashboard

### Problem

You want to be able to turn lighting and other appliances on and off from your smartphone.

### Solution

Install the Node-RED Dashboard extension, add some user interface (UI) controls to a flow, and then visit the Dashboard from your phone's browser.

To install the Node-RED Dashboard, run the following commands:

```
$ sudo systemctl stop nodered.service
$ apt update
$ cd ~/.node-red
$ sudo apt install npm
$ npm install node-red-dashboard
$ sudo systemctl start nodered.service
```

When you've finished installing the Dashboard, you will end up with a new section of control nodes in Node-RED ([Figure 18-23](#)).



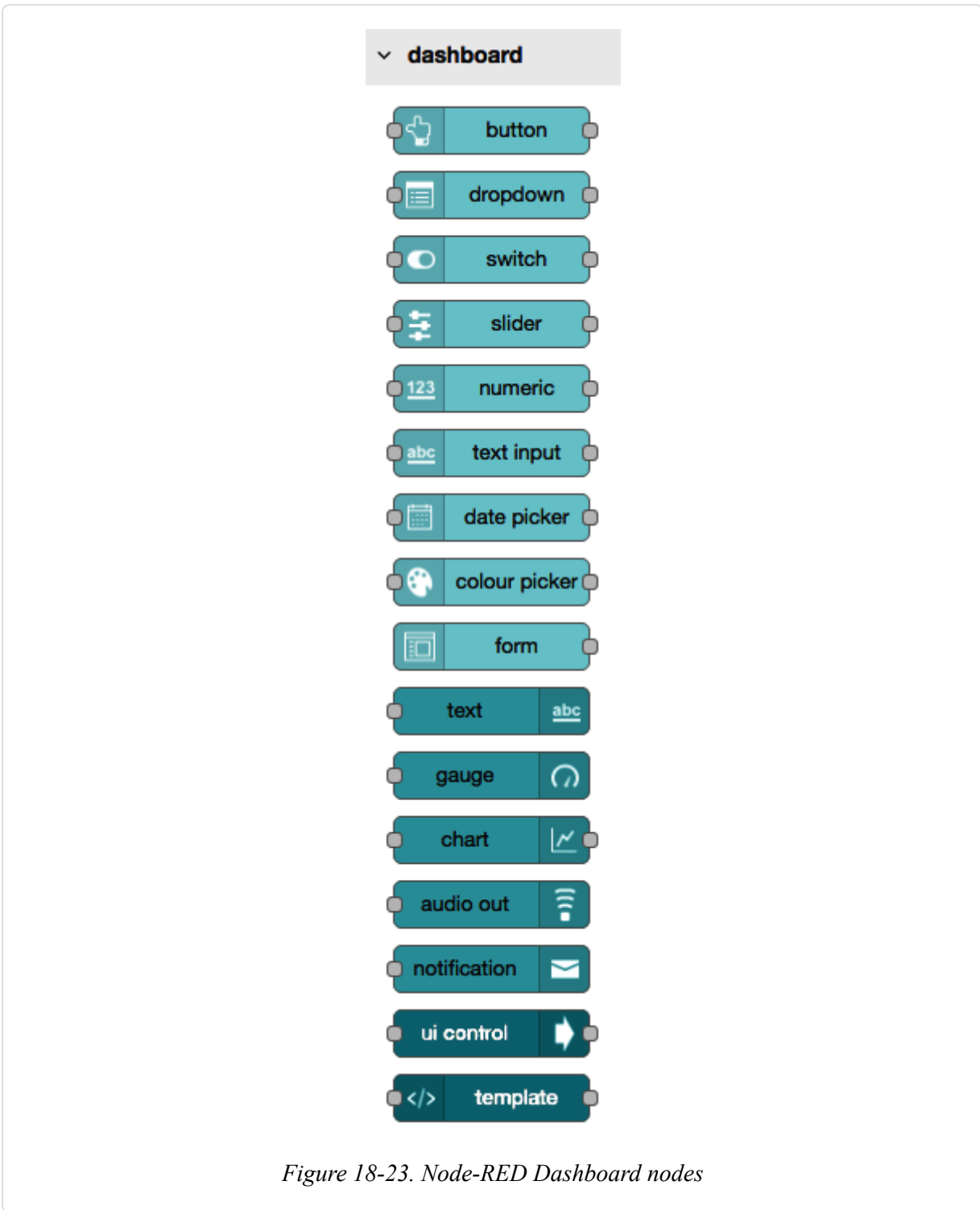


Figure 18-23. Node-RED Dashboard nodes

We can use the button node to replace the physical button that we used in [Recipe 18.6](#) with a button on a web page. The flow for this is shown in [Figure 18-24](#). If you want, you can **import the flow** rather than build it up

from scratch. Follow the instructions in the Discussion section of [Recipe 18.2](#) to import the flow.

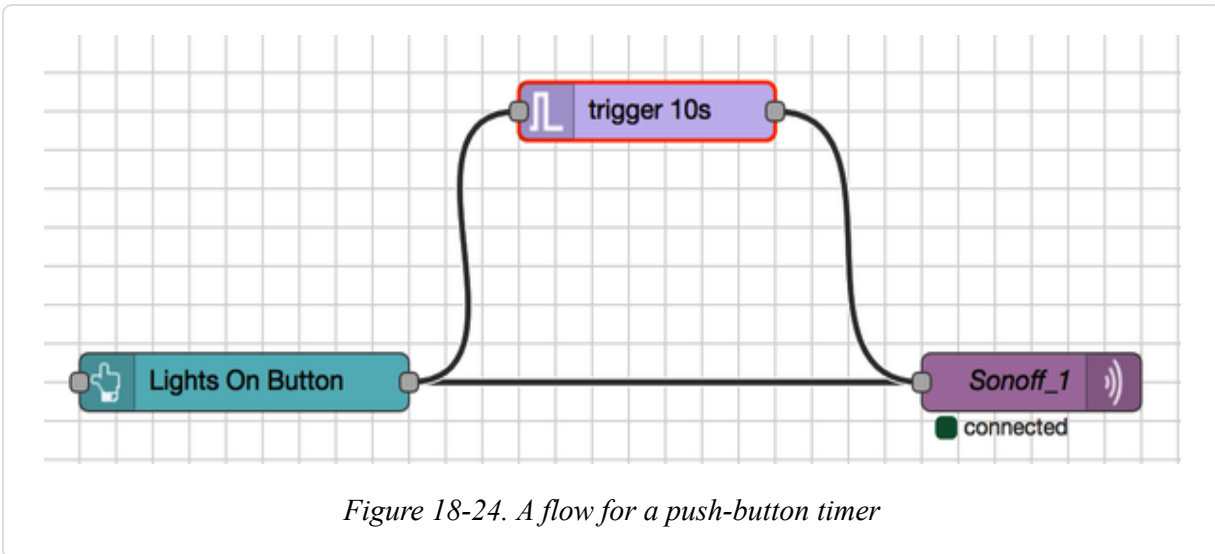


Figure 18-24. A flow for a push-button timer

The trigger button is the same as the “Off trigger” in [Recipe 18.6](#), and the “Sonoff\_1” node is the same as the same node in [Recipe 18.6](#). However, the Raspberry Pi GPIO node is replaced by a Dashboard button node.

Because you might need quite a few controls to remotely control your home automation system, Dashboard controls are collected into a *group*, and groups are themselves collected into *tabs*. You can define your own groups and tabs when you add a new node from the Dashboard category. This happens when you edit the node ([Figure 18-25](#)).










In [Figure 18-25](#) you can see that the Group is set to Lights in the tab [Home]. To get to this point, I had to click on “Add new UI group.” This in turn asked me to “Add a new tab.” Once you have created a tab and group, these will become defaults. You don’t have to create them every time.

Note that in [Figure 18-25](#) the Payload is set to 1 to turn the light on, and it’s connected directly to the “mqtt” node.

**Edit button node**

Delete
Cancel
Done

▼ node properties

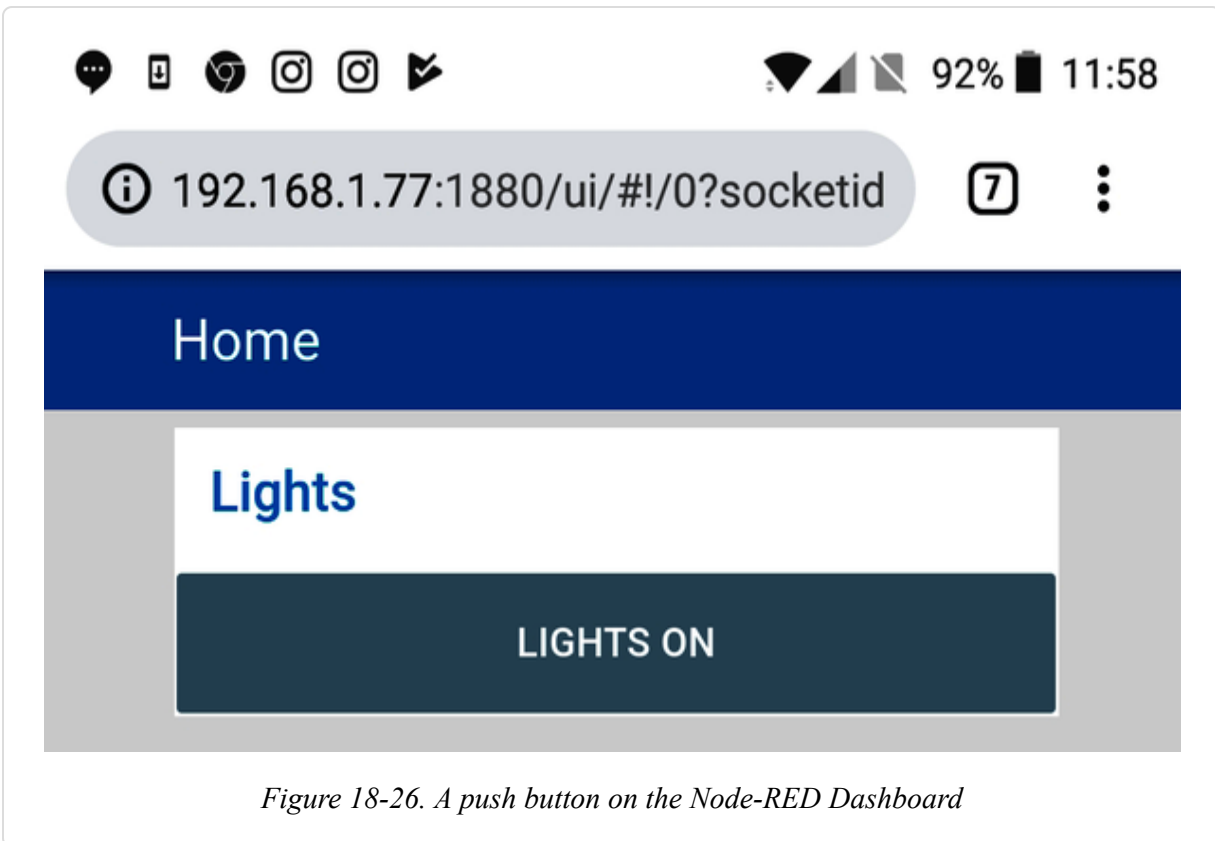
 Group	[Home] Lights	↕	
 Size	auto		
 Icon	optional icon		
 Label	Lights		
 Tooltip	optional tooltip		
 Colour	optional text/icon color		
 Background	optional background color		
<input checked="" type="checkbox"/> When clicked, send:			
Payload	▼ a <sub>2</sub> 1		
Topic			
→ If <span style="border: 1px solid #ccc; padding: 2px;">msg</span> arrives on input, emulate a button click: <input type="checkbox"/>			
 Name	Lights On Button		

*Figure 18-25. Configuring the Lights On Button node*

You can now deploy the workflow.

To try out the new flow, open a browser on your phone (or any computer on your network) and enter your Raspberry Pi's IP address with `:1880/ui` on

the end. For my Raspberry Pi, the full URL is *http://192.168.1.77:1880/ui*. The screen should look something like **Figure 18-26**.



*Figure 18-26. A push button on the Node-RED Dashboard*

When you click the button on your phone, the Sonoff should turn on for 10 seconds.

## **Discussion**

Even though it's quite useful to be able to turn the light on for a preset timed period, it would also be useful to have an override on/off switch. In **Figure 18-27**, a switch has been added. This flow is also available on [GitHub](#).

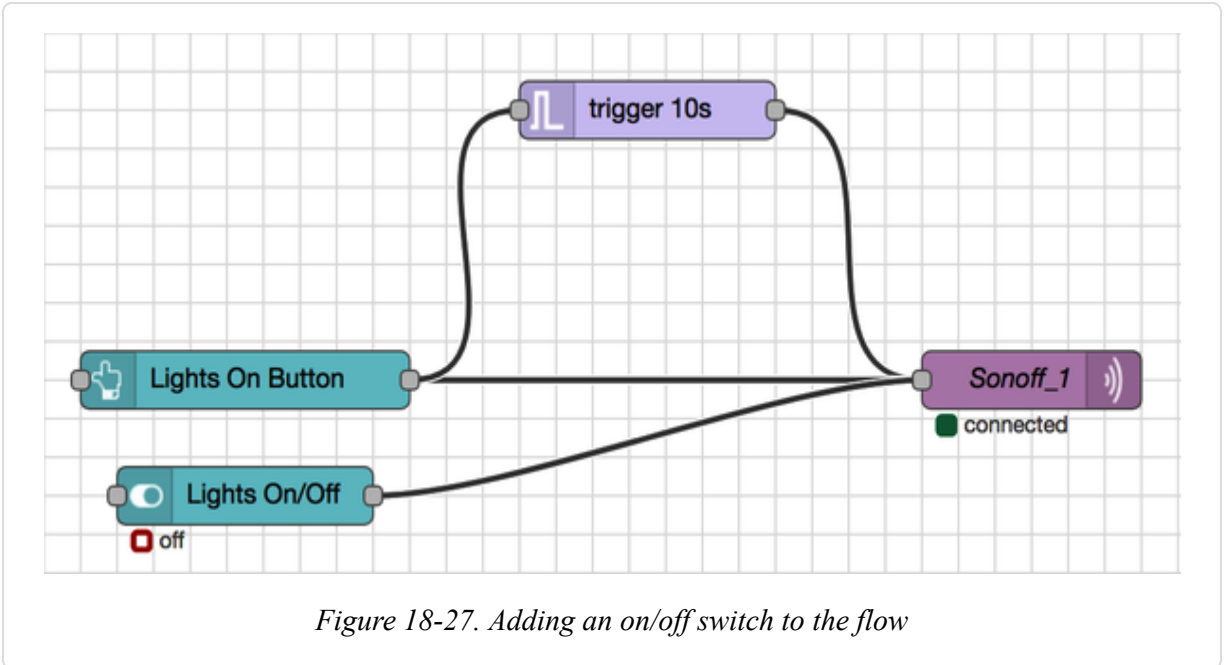


Figure 18-27. Adding an on/off switch to the flow

You also can connect this to the “Sonoff\_1” MQTT node so that it, as well as the push button, can be used to turn the light on and off. **Figure 18-28** shows the settings for the Lights On/Off switch.

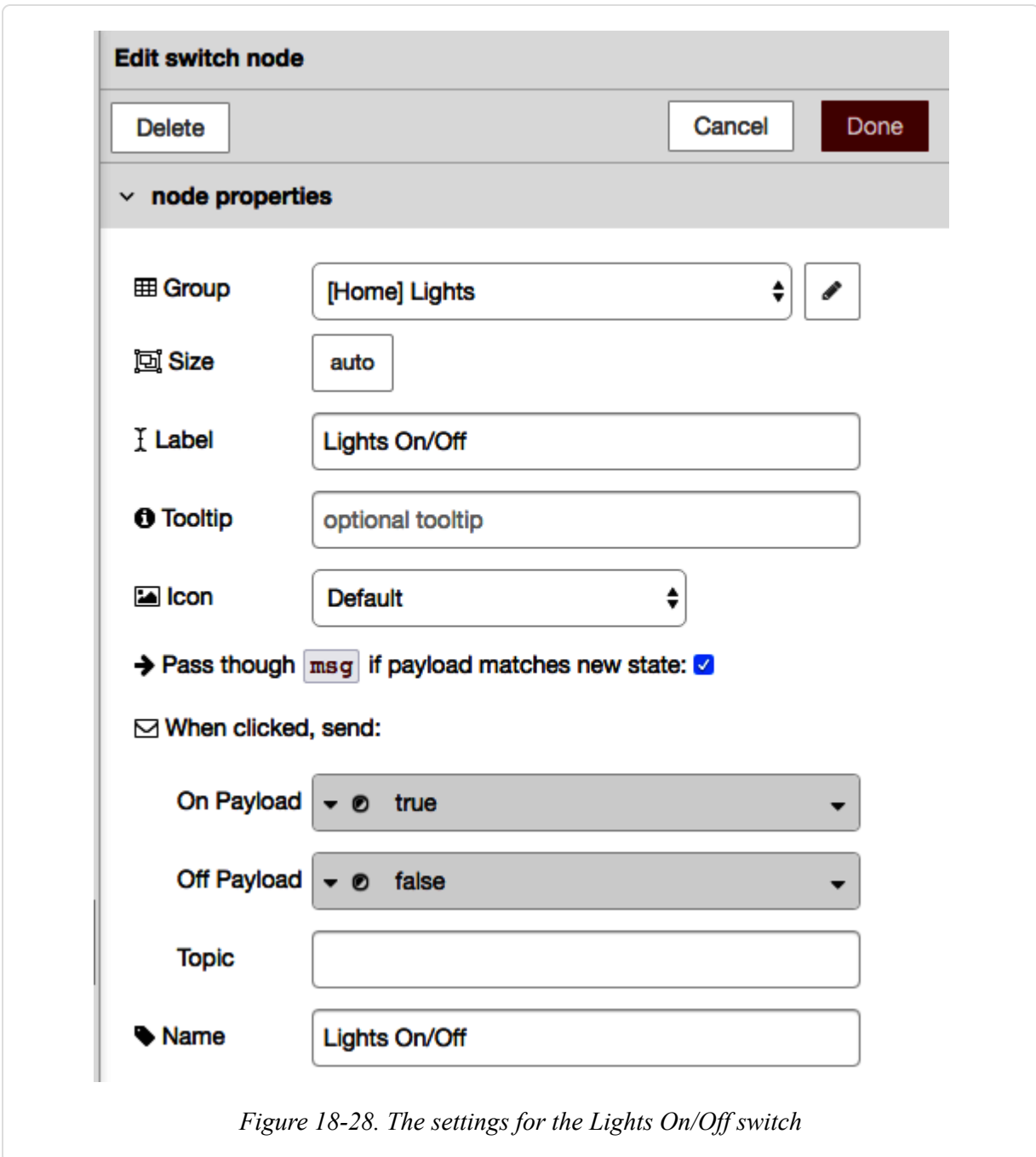


Figure 18-28. The settings for the Lights On/Off switch

When the flow is deployed, the UI will display the extra switch (Figure 18-29) automatically when you go back to the *ui* web page on your phone (something like `http://192.168.1.77:1880/ui`, but with your Node-RED server's IP address).

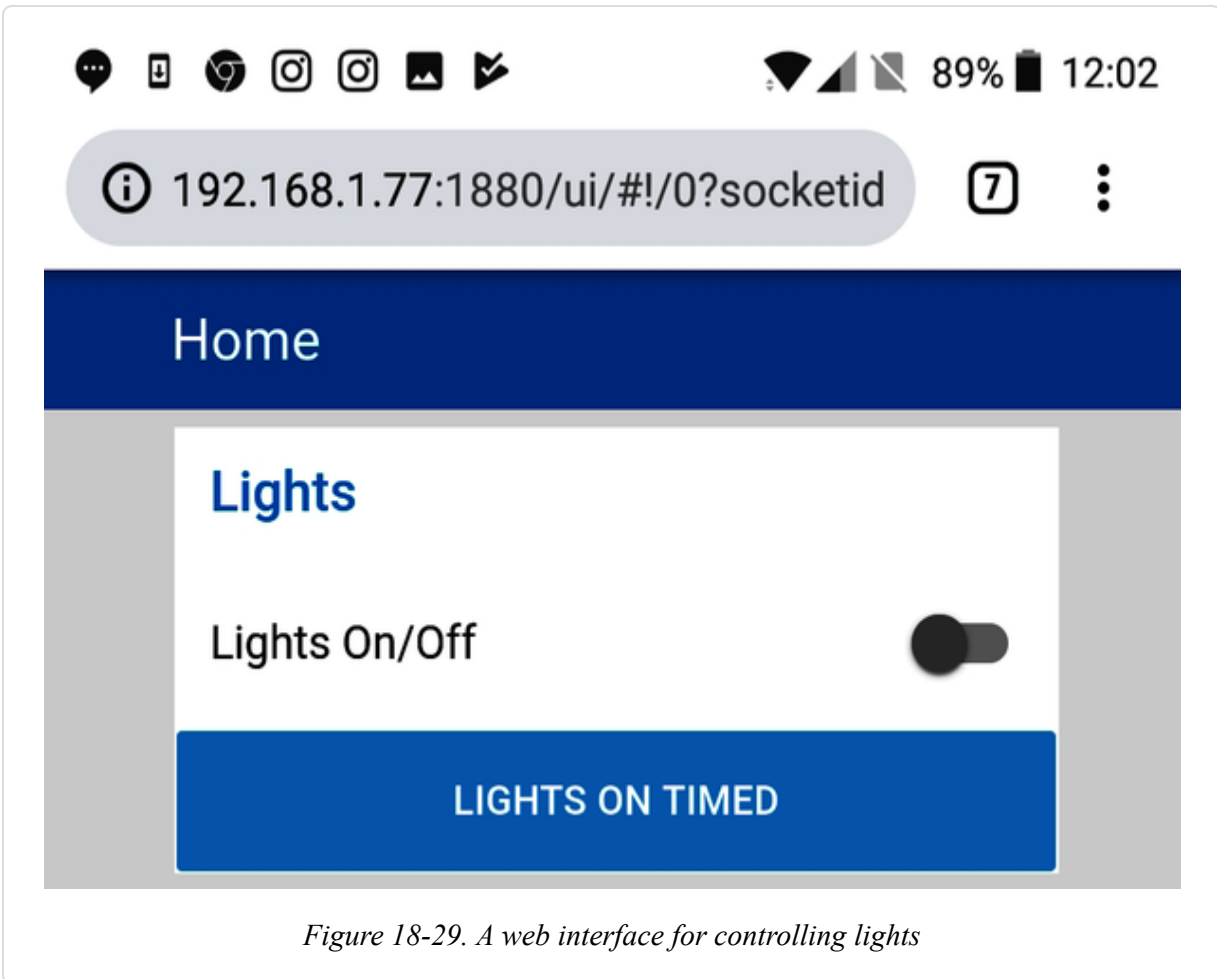


Figure 18-29. A web interface for controlling lights

## See Also

More information is available in the full [Node-RED documentation](#).

## 18.8 Scheduling Events with Node-RED

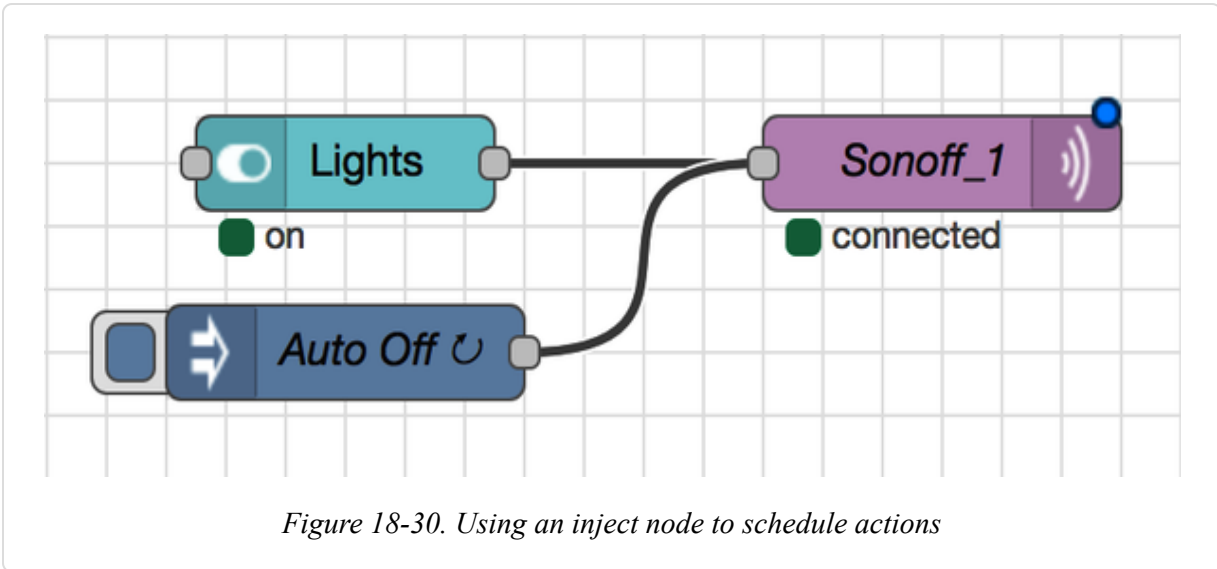
### Problem

You want to use Node-RED to do something at a certain time—for example, to turn out all the lights at 1 a.m. every night.

### Solution

Use the Node-RED *inject* node.

The flow shown in [Figure 18-30](#) is based on the flows of [Recipe 18.6](#). If you want, you can **import the flow** rather than build it up from scratch. Follow the instructions in the Discussion section of [Recipe 18.2](#) to import the flow.



A dashboard *switch* is used to turn the Sonoff (assumed to be switching a light) on and off but, in addition, there is an inject node (Auto Off) that is configured to inject a message of 0 to the “Sonoff\_1” MQTT node.

[Figure 18-31](#) shows the configuration for the inject node.



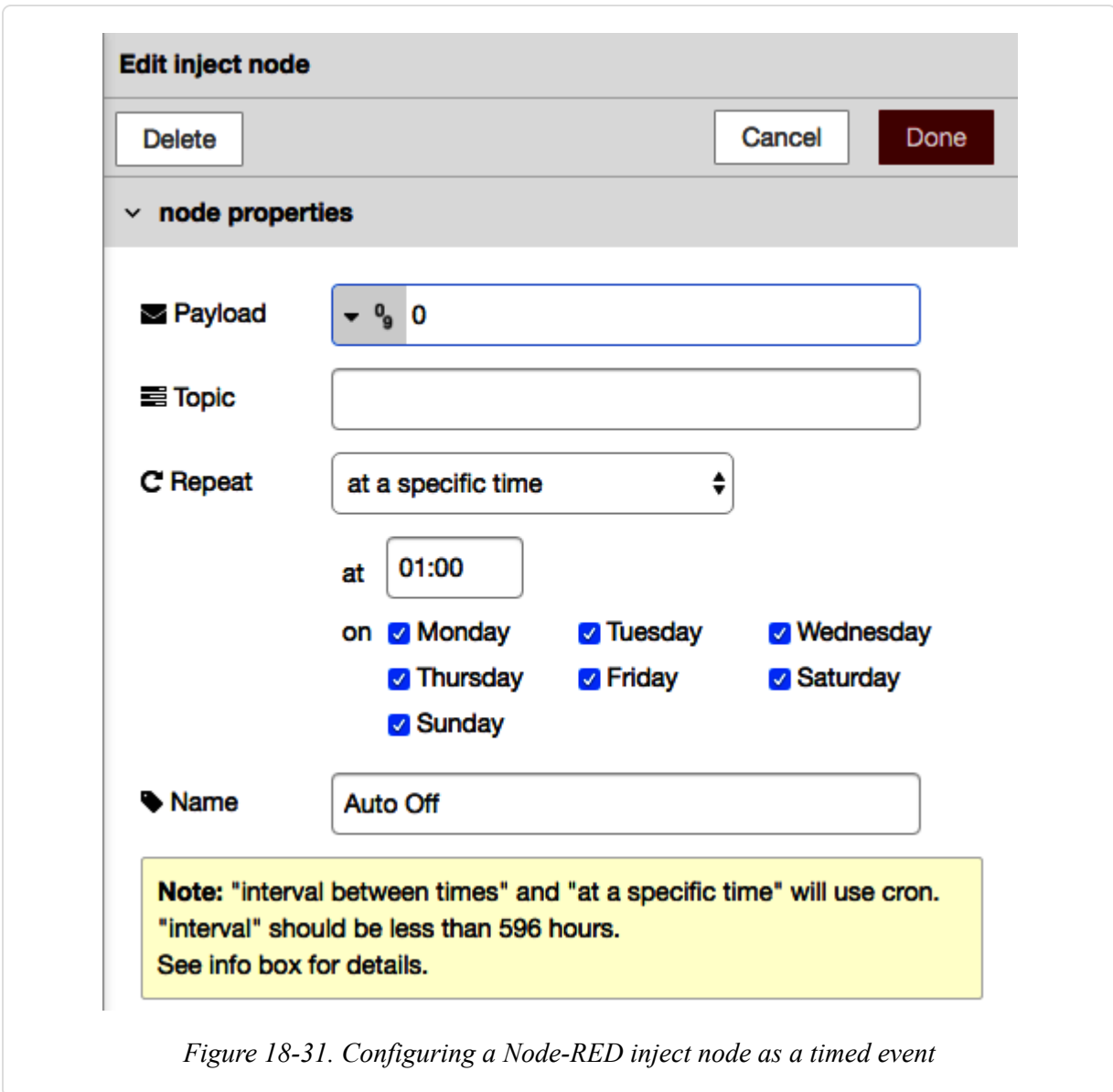


Figure 18-31. Configuring a Node-RED inject node as a timed event

## Discussion

If you had a number of Sonoff switches connected to appliances all around your house, one inject node could turn them all off by fanning out the message, as shown in [Figure 18-32](#).

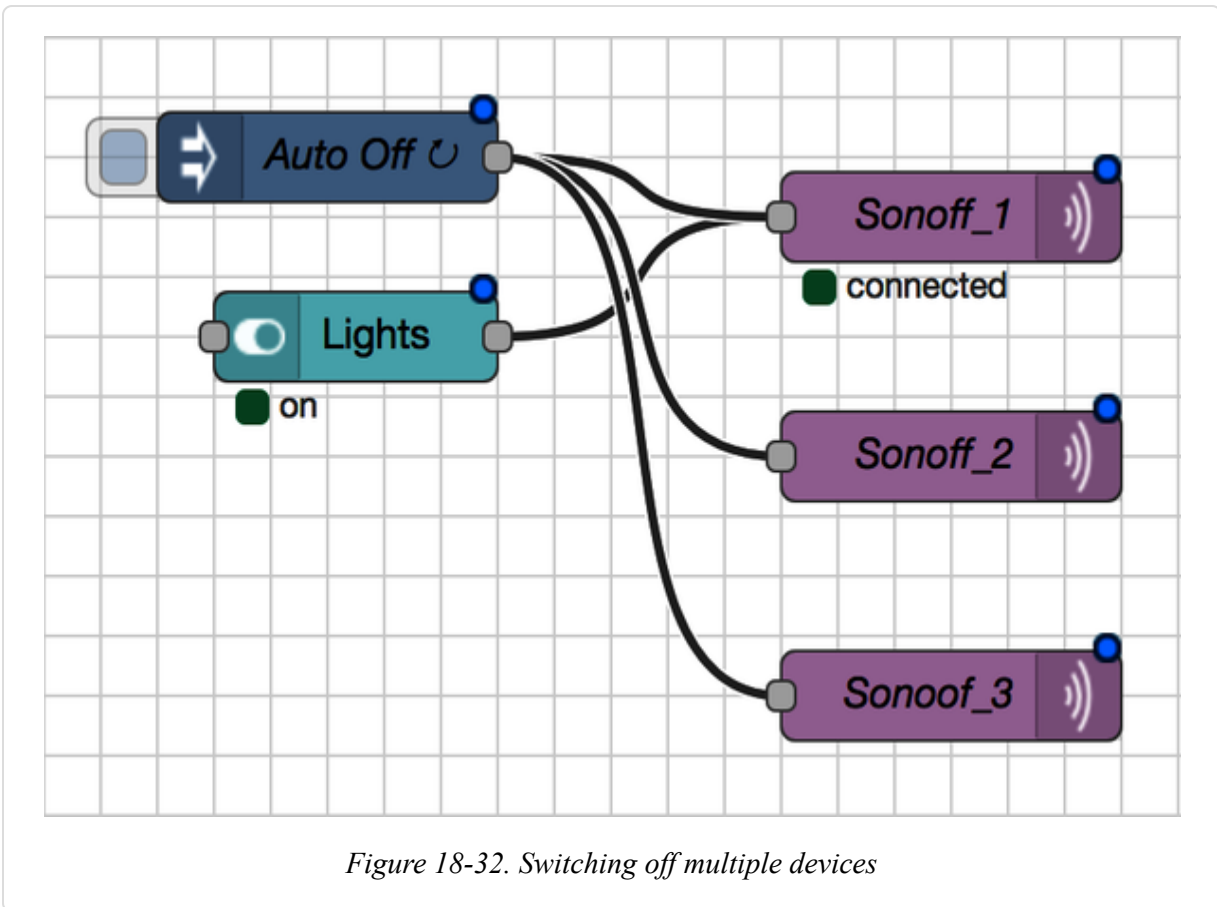


Figure 18-32. Switching off multiple devices

## See Also

More information is available in the full [Node-RED documentation](#).

## 18.9 Publishing MQTT Messages from a Wemos D1

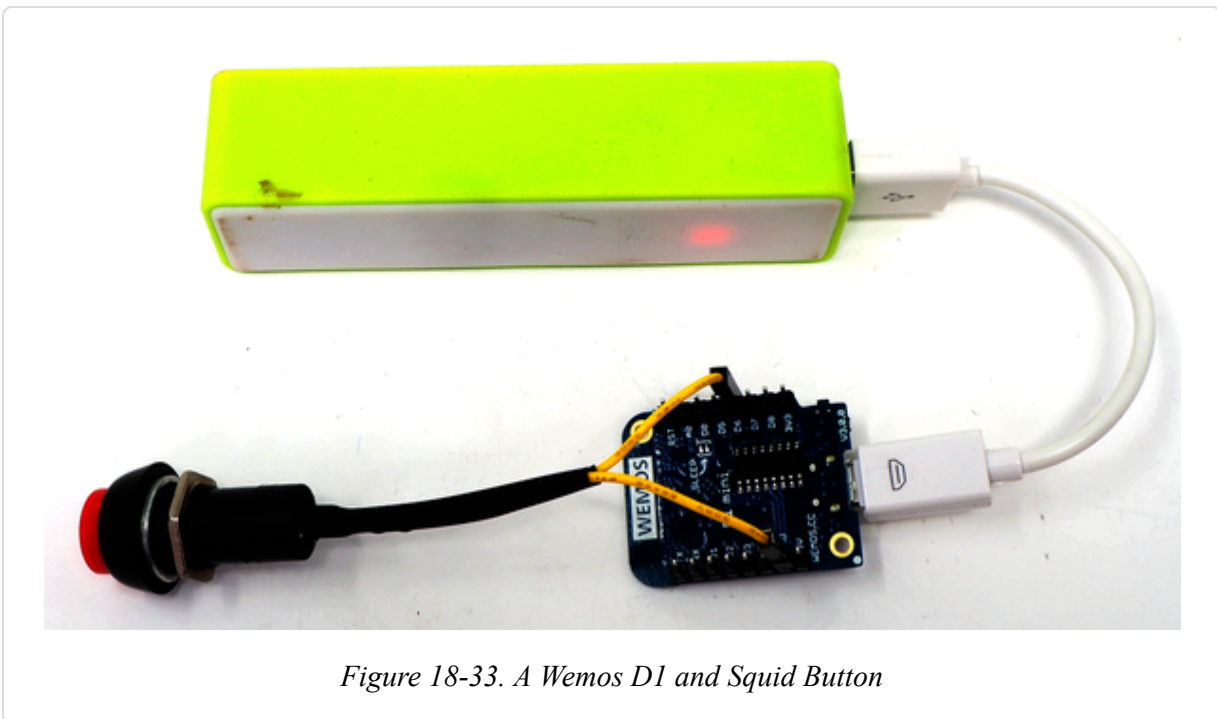
### Problem

You want to be able to publish MQTT messages using a low-cost programmable WiFi board, perhaps in response to a button being pressed.

### Solution

Use a low-cost ESP8266-based board like the Wemos D1 with customized software. **Figure 18-33** shows a Wemos D1 with a Squid Button attached to one of its GPIO pins, powered by a USB power bank.

When the button is pressed, a message will be published to an MQTT server.



*Figure 18-33. A Wemos D1 and Squid Button*

To make this recipe, you will need the following:

- A Wemos D1 mini (see “**Modules**”)
- A Raspberry Squid Button (see “**Modules**”)
- A USB power bank or other means of powering the Wemos D1

To be able to program the Wemos D1 from your Raspberry Pi, you need to first install the **Arduino integrated development environment (IDE)**. Then you’ll need to **add support for the ESP8266**.

Connect the Squid Button or other switch between the Wemos pin named D6 and GND.

Before you can use the *sketch* (the name for Arduino programs), you need to install an MQTT library into the Arduino IDE, so download the library as

a ZIP file using the following commands:

```
$ cd ~  
$ wget  
https://github.com/knolleary/pubsubclient/archive/master.zip
```

Next, open the Arduino IDE. From the Sketch menu, select Include Library, and then select Add ZIP Library. Navigate to the file *master.zip* that you just downloaded, and the library will be installed.

The Arduino program for this is available as part of the downloads for the book (see [Recipe 3.22](#)). You will find it in the folder called *ch\_17\_web\_switch*, inside a folder at the same level as the Python folder called *arduino*.

Open the sketch in the Arduino IDE by clicking *ch\_17\_web\_switch.ino*. Set the board type to Wemos D1 and the serial port to `/dev/ttyUSB0`.

Before uploading the sketch to the Arduino, you need to make a few changes to the code. Look for these lines near the top of the sketch:

```
const char* ssid = "your wifi access point name";  
const char* password = "your wifi password";  
const char* mqtt_server = "your MQTT IP address";
```

Replace the placeholder text for `ssid`, `password`, and `mqtt_server` with the appropriate values for your setup.

Then click the upload button in the Arduino IDE.

After you've programmed it, the Wemos doesn't need to be connected to your Raspberry Pi so, if you want, you can power it by some other means, such as a USB power bank. However, the sketch prints out useful debug information, so it's worth staying tethered to your Raspberry Pi until you know everything is working. To see this information, open the Arduino IDE's serial monitor, which should look something like [Figure 18-34](#).

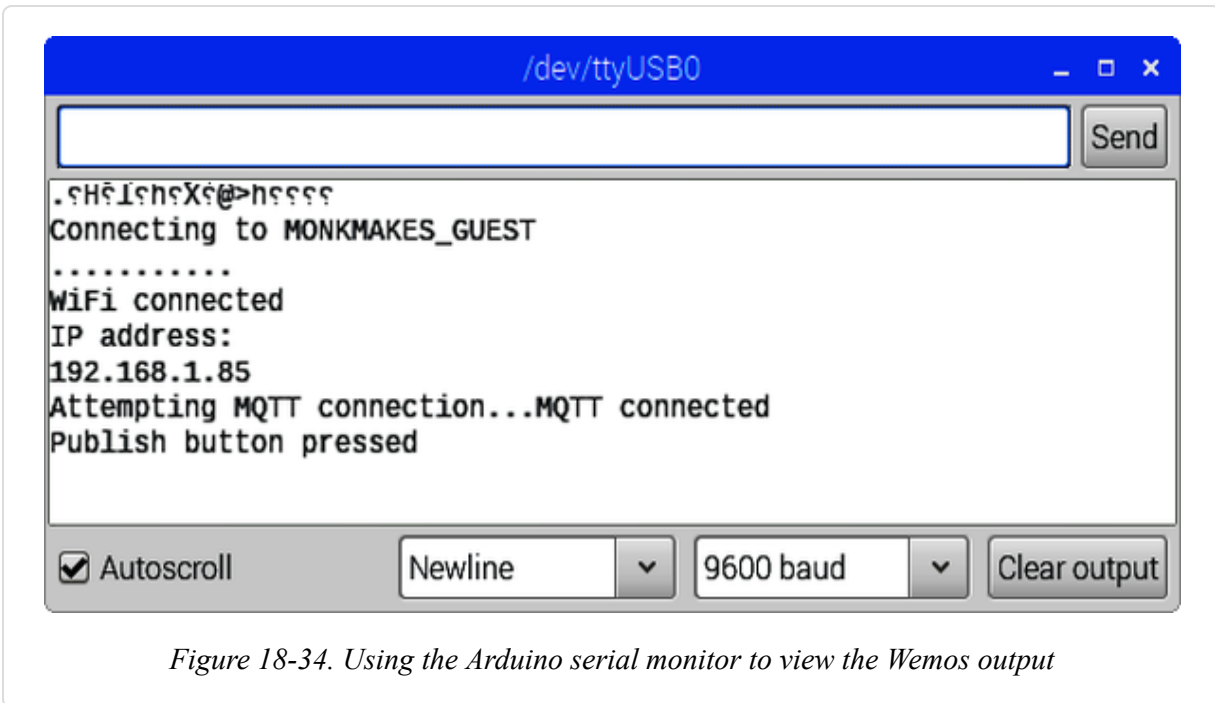


Figure 18-34. Using the Arduino serial monitor to view the Wemos output

To test the recipe, start a Terminal session on your Raspberry Pi and run the following command to subscribe to button presses:

```

$ mosquitto_sub -d -t button_1
Client mosqsub/5007-raspberryp sending CONNECT
Client mosqsub/5007-raspberryp received CONNACK
Client mosqsub/5007-raspberryp sending SUBSCRIBE (Mid: 1, Topic:
button_1,
                                                    QoS: 0)
Client mosqsub/5007-raspberryp received SUBACK
Subscribed (mid: 1): 0
Client mosqsub/5007-raspberryp received PUBLISH (d0, q0, r0, m0,
'button_1', ...
                                                    (16 bytes))

Button 1 pressed
  
```

Now every time you press the button, the message “Button 1 pressed” should appear in the Terminal.

## Discussion

This is a book about Raspberry Pi, not Arduino, so we won’t go through the Arduino C code in any detail.

The code is based on the example sketch called `mqtt_basic` in the library `PubSubClient`. As well as the constants at the top of the file for your WiFi credentials, these lines:

```
const char* topic = "button_1";  
const char* message = "Button 1 pressed";
```

might also be of interest. They determine the MQTT topic to be used and the message to accompany published events. You could program an entire series of Wemos buttons with different topics and messages.

## See Also

To use your newly configured Wemos with Node-RED, see [Recipe 18.10](#). You will learn much more about devices like the Wemos in [Chapter 19](#).

## 18.10 Using a Wemos D1 with Node-RED

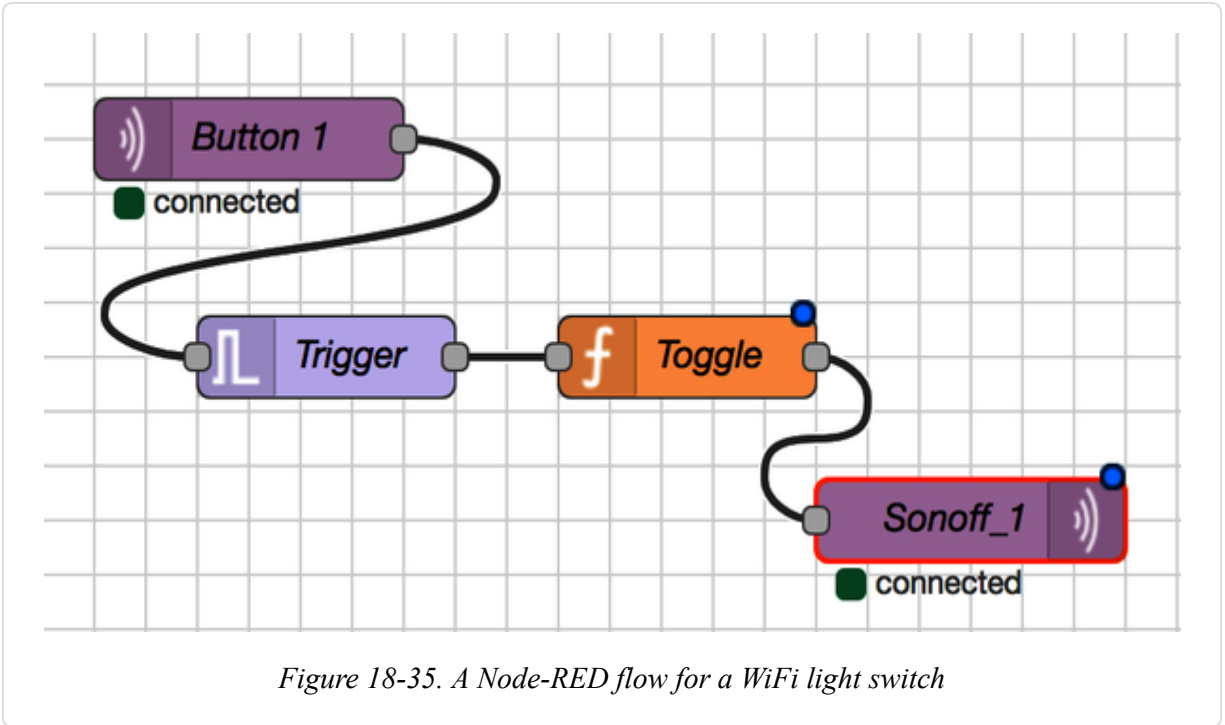
### Problem

You want to include a Wemos D1, with a button attached, in a Node-RED flow.

### Solution

As an example, we can use the Wemos WiFi button from [Recipe 18.9](#) in a Node-RED flow to toggle a Sonoff web switch ([Recipe 18.4](#)) on and off.

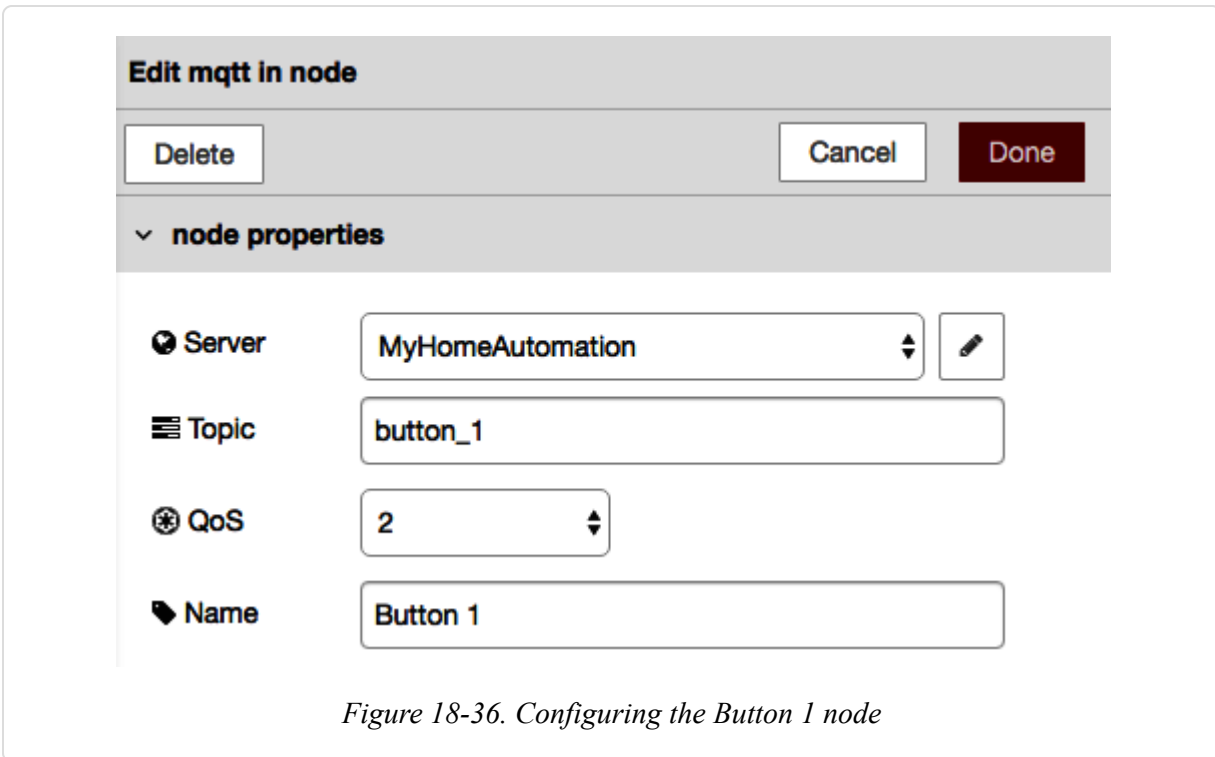
[Figure 18-35](#) shows the Node-RED flow for this. If you want, you can [import the flow](#) rather than build it up from scratch. Follow the instructions in the Discussion section of [Recipe 18.2](#) to import the flow.



*Figure 18-35. A Node-RED flow for a WiFi light switch*

To make this recipe, you will benefit from having already tried all the previous recipes in this chapter.

The Button 1 MQTT node subscribes to messages from the Wemos D1. [Figure 18-36](#) shows the settings for this node.



The important thing here is that the Topic is set to “button\_1.” The Trigger node works in the same way as in [Recipe 18.6](#). The Trigger node is more interesting because this node remembers a value, which can be a 1 or a 0, and flips this value each time a message passes through it. This has to be done as a general function with a little bit of JavaScript code that remembers the state and toggles it. [Figure 18-37](#) shows the node’s configuration including the code. This is a useful function that you will probably find yourself reusing in other projects.



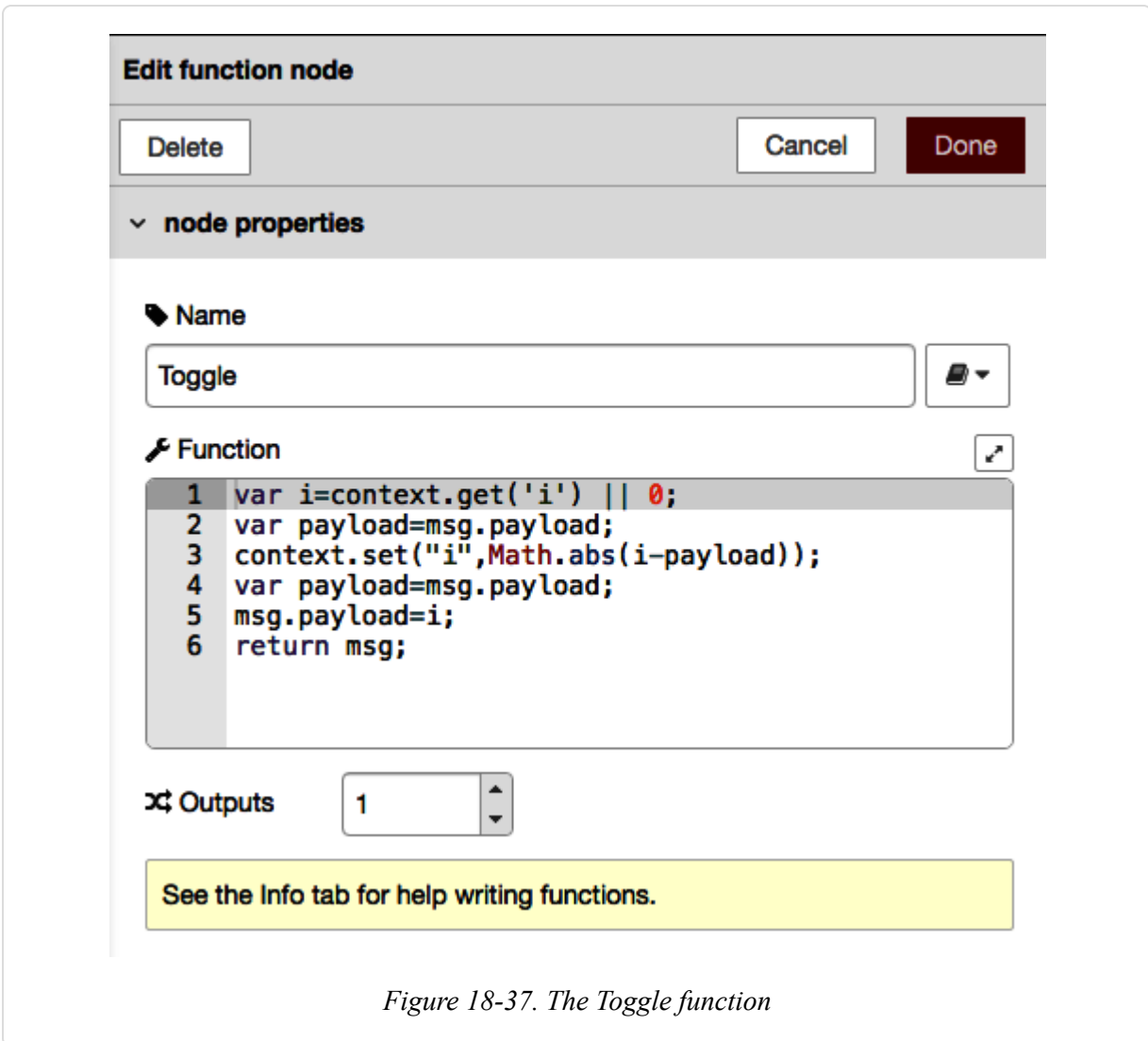


Figure 18-37. The Toggle function

## Discussion

Node-RED is a powerful way to quickly put together a home automation system. If you are used to a conventional programming language, this way of doing things takes a little getting used to; however, when you've mastered it, you won't want to go back to writing reams of code.

Node-RED's palette has lots of other interesting nodes, so take some time to explore them. Hovering over a node will display details about what it does, and you can follow this up by dragging the node onto a flow and trying it out.

## See Also

More information is available in the full [Node-RED documentation](#).

# Chapter 19. Raspberry Pi Pico and Pico W

---

## 19.0 Introduction

Although a regular Raspberry Pi is ideal for projects that need a network connection or a graphical user interface (GUI), its power consumption and lack of any analog inputs puts it at a disadvantage to a simpler microcontroller board such as the [Arduino](#) or Raspberry Pi Pico.

The Raspberry Pi Pico is very different from any other Raspberry Pi in several ways:

- It doesn't have any interface to keyboard, mouse, or screen.
- It has a relatively small 2M of flash memory for storing programs (no microSD slot) and 264k of RAM.
- Its processor runs at just 133 MHz, compared with the Raspberry Pi 4's 1.2 GHz.
- It doesn't have an operating system. In effect, MicroPython is its operating system, and you just get what you see when you're on the Python command line.

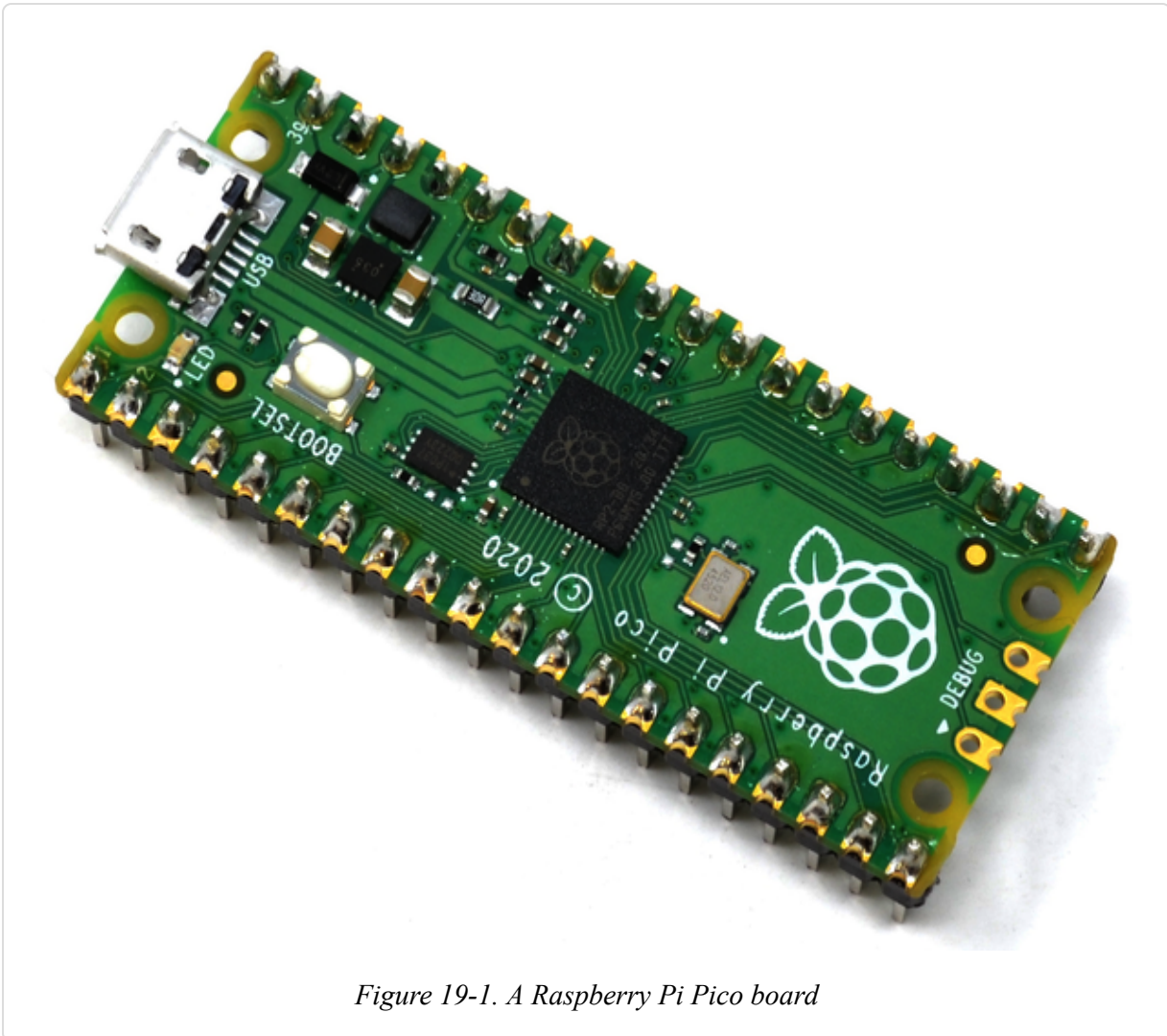
This might lead you to wonder why you would use such an apparently feeble board rather than, say, a Raspberry Pi Zero.

The answer is that the Pico ([Figure 19-1](#)) is even lower cost than a Pi Zero and better than a Raspberry Pi 4 at interfacing with external electronics in several ways. For example:

- Three analog inputs makes connecting analog sensors much easier than with a regular Raspberry Pi.
- Six pulse-width modulation (PWM) outputs. These outputs are hardware-timed and produce a much more accurate PWM signal than

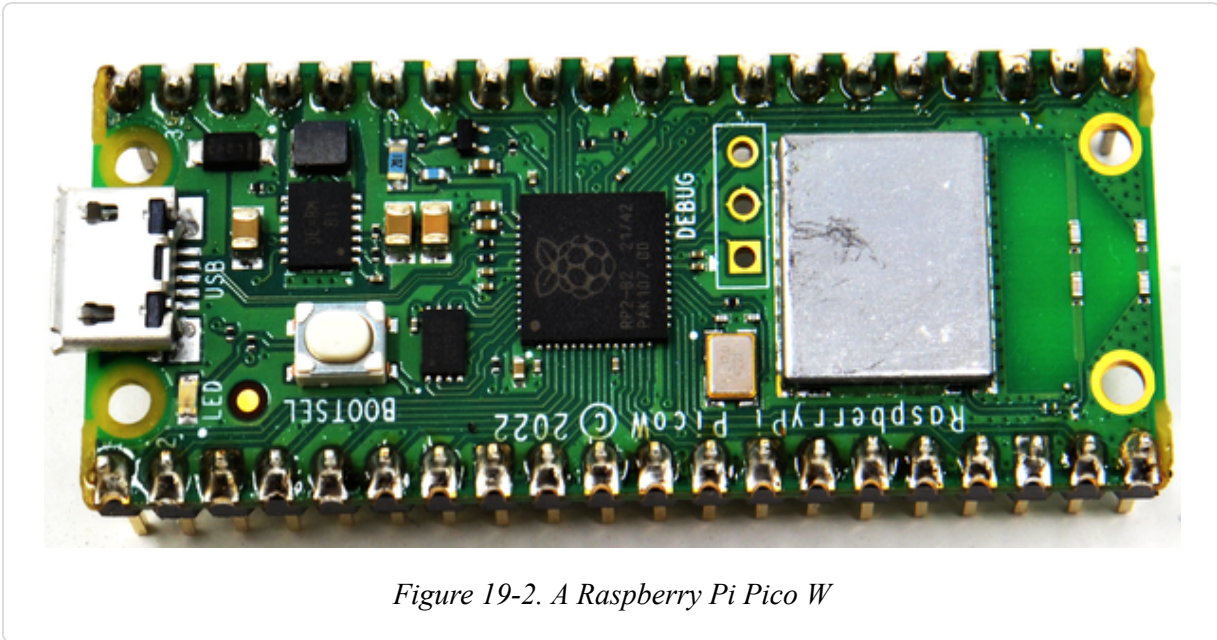
can be achieved with the Raspberry Pi, making them a lot better for controlling servomotors.

- Built-in power supply that allows the Pico to be powered from 1.8 to 5.5V, and low power consumption so you can run it from a pair of AA batteries for hours.



*Figure 19-1. A Raspberry Pi Pico board*

The Raspberry Pi Pico W (Figure 19-2) shares the same pinout as the Pico, but the metal rectangle that you see on the right of the board is a WiFi and Bluetooth module. This makes the Pico W suitable for connected projects. The Pico W is still a good value, but considerably more expensive than the Pico.



*Figure 19-2. A Raspberry Pi Pico W*

In the remainder of this chapter, I will refer to the Raspberry Pi Pico and Pico W as just “Pico” (except when the difference matters) and refer to a regular Raspberry Pi 4 or 400, etc. as just a “Raspberry Pi.”

## 19.1 Connecting a Pico or Pico W to a Computer

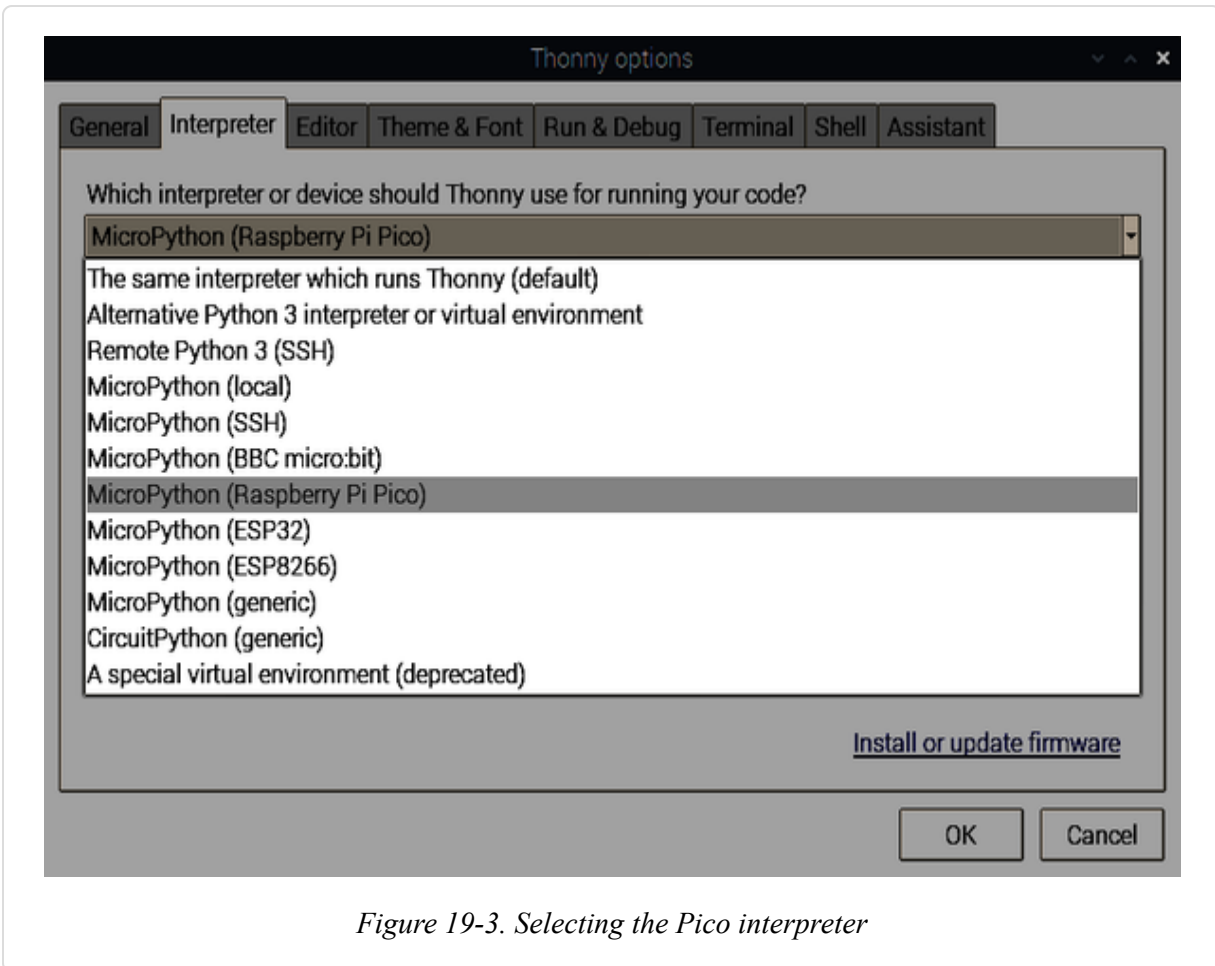
### Problem

You want to connect your Pico or Pico W to your Raspberry Pi or a Mac or Windows PC so that you can program it.

### Solution

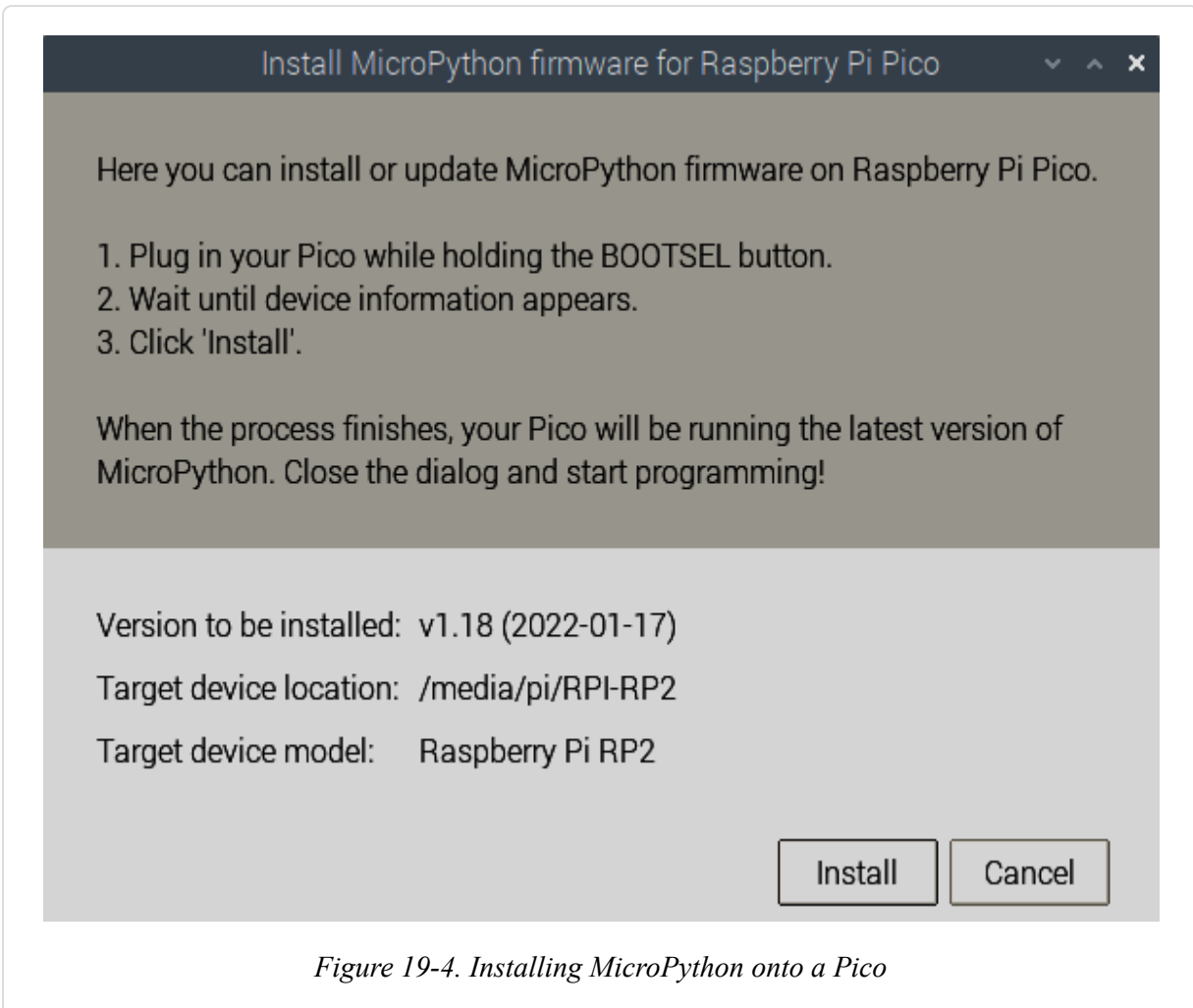
The Thonny Python editor has support for the Pico and Pico W and is pre-installed with Raspberry Pi OS. If you have not used Thonny, open it from the Programming section of the Raspberry Menu.

To configure Thonny for Pico use, you need to set it into Normal mode ([Recipe 5.3](#)), so that you have access to the menu, and then select Options from the Tools menu and go to the Interpreter tab ([Figure 19-3](#)).



*Figure 19-3. Selecting the Pico interpreter*

On the drop-down list, select MicroPython (Raspberry Pi Pico) as the interpreter to use, and then click OK. This will take you to a screen offering to install MicroPython onto your Pico (**Figure 19-4**). To do this, you need to use a micro-USB lead to connect the Pico to one of your Raspberry Pi's USB ports. Hold down the BOOTSEL button on the top of the Pico while you plug it in. This will put the Pico into a boot mode that will allow Thonny to install MicroPython onto it.



Click on Install and MicroPython will be flashed onto the Pico. You shouldn't need to perform this step again for that Pico.

## INSTALLING FIRMWARE BY DOWNLOAD

Occasionally, you might find that your Pico gets itself in a zombie state and appears to be broken. Often this can be fixed by manually replacing the firmware. Think of this as a factory reset.

If you have a Pico W, you'll need to download a specific version of the firmware for it.

To manually install new firmware onto your Pico or Pico W, start by unplugging it from your Raspberry Pi or other computer.

If you have a Pico W, go to <https://rpf.io/pico-w-firmware>. This will immediately start the download of a *.uf2* firmware file for the Pico W.

If, on the other hand, you have a regular Pico, **download the file**.

Next, hold down the BOOTSEL button (Figure 19-5), and with the button held down, connect the Pico with your USB lead.

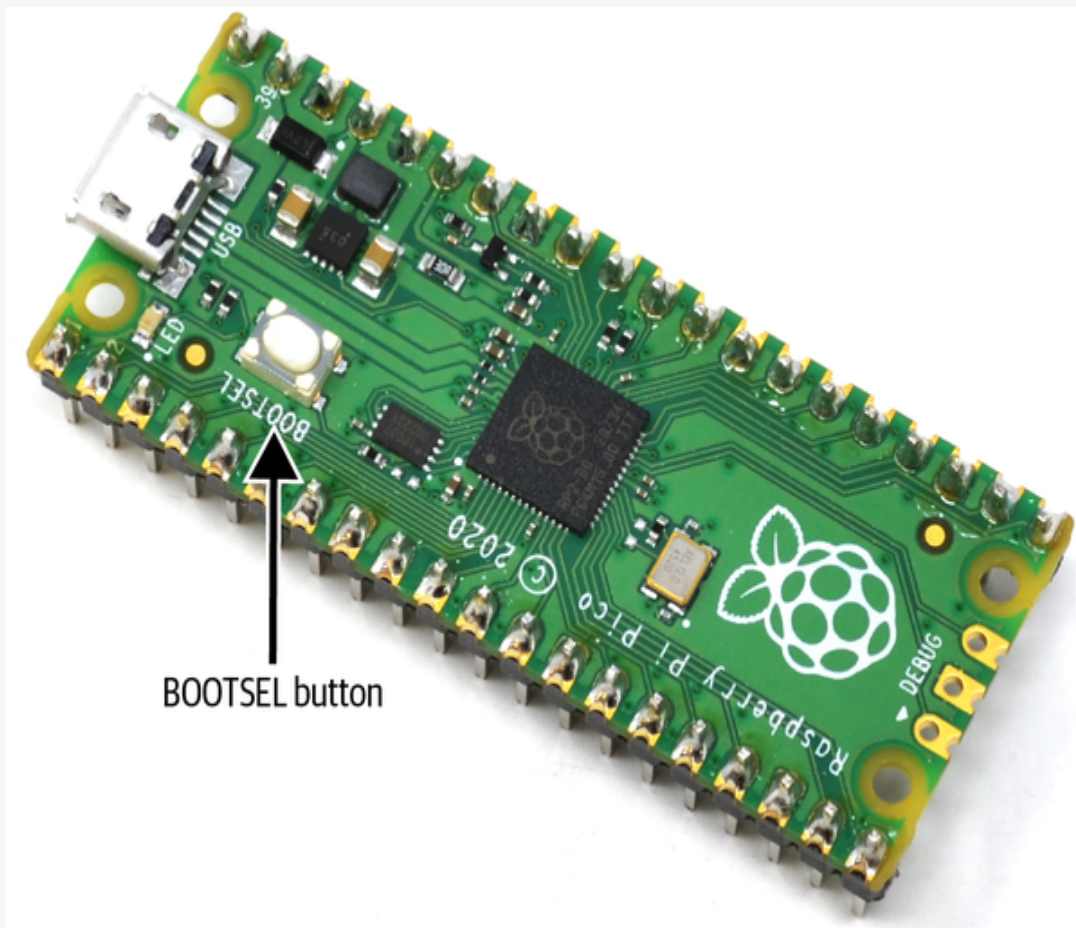


Figure 19-5. The BOOTSEL button



Release the BOOTSEL button and your Pico or Pico W will mount itself as USB flash storage on your Raspberry Pi (Figure 19-6, right).

Drag or copy the `.uf2` file that you downloaded earlier onto the Pico or Pico W's filesystem.

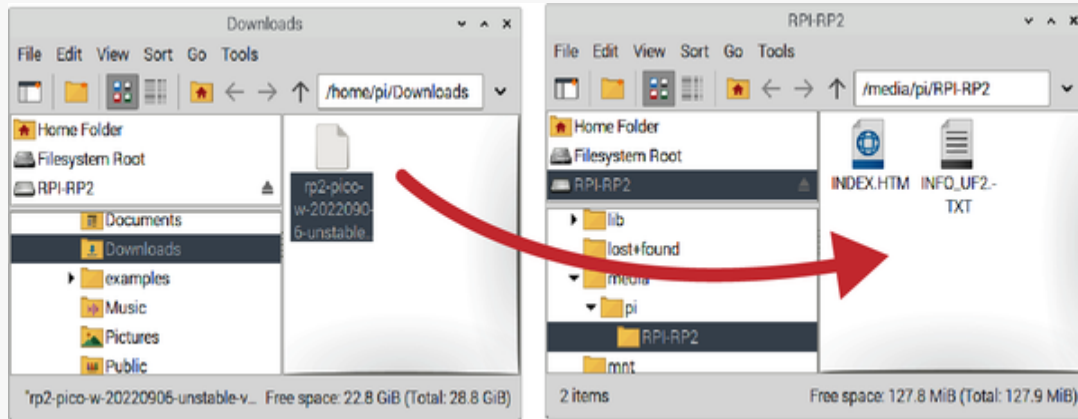


Figure 19-6. Dragging new firmware onto a Pico or Pico W

Once the file is copied, the Pico or Pico W will unmount itself and be ready to receive your MicroPython programs via Thonny.

If you find that installing the firmware here doesn't work on a Raspberry Pi, you can also install Thonny on a Mac or Windows PC and then install the firmware onto the Pico.

## Discussion

When you choose to install MicroPython onto your Pico or Pico W, what happens is that a cut-down version of Python 3 called *MicroPython* is installed onto the Pico's flash memory. As you will see in [Recipe 19.2](#) this allows you to interact with a Python Shell on the Pico and also copy Python programs to be run onto the Pico's flash memory.

Although MicroPython is a cut-down version of Python 3, you will find that most of the material covered in [Chapters 5, 6, and 7](#) will also work in MicroPython. In addition, other hardware-specific libraries are available to use the Pico's GPIO pins (among other things).

## See Also

For more information on Thonny, see [Recipe 5.3](#).

## 19.2 Using the Python Shell on a Pico

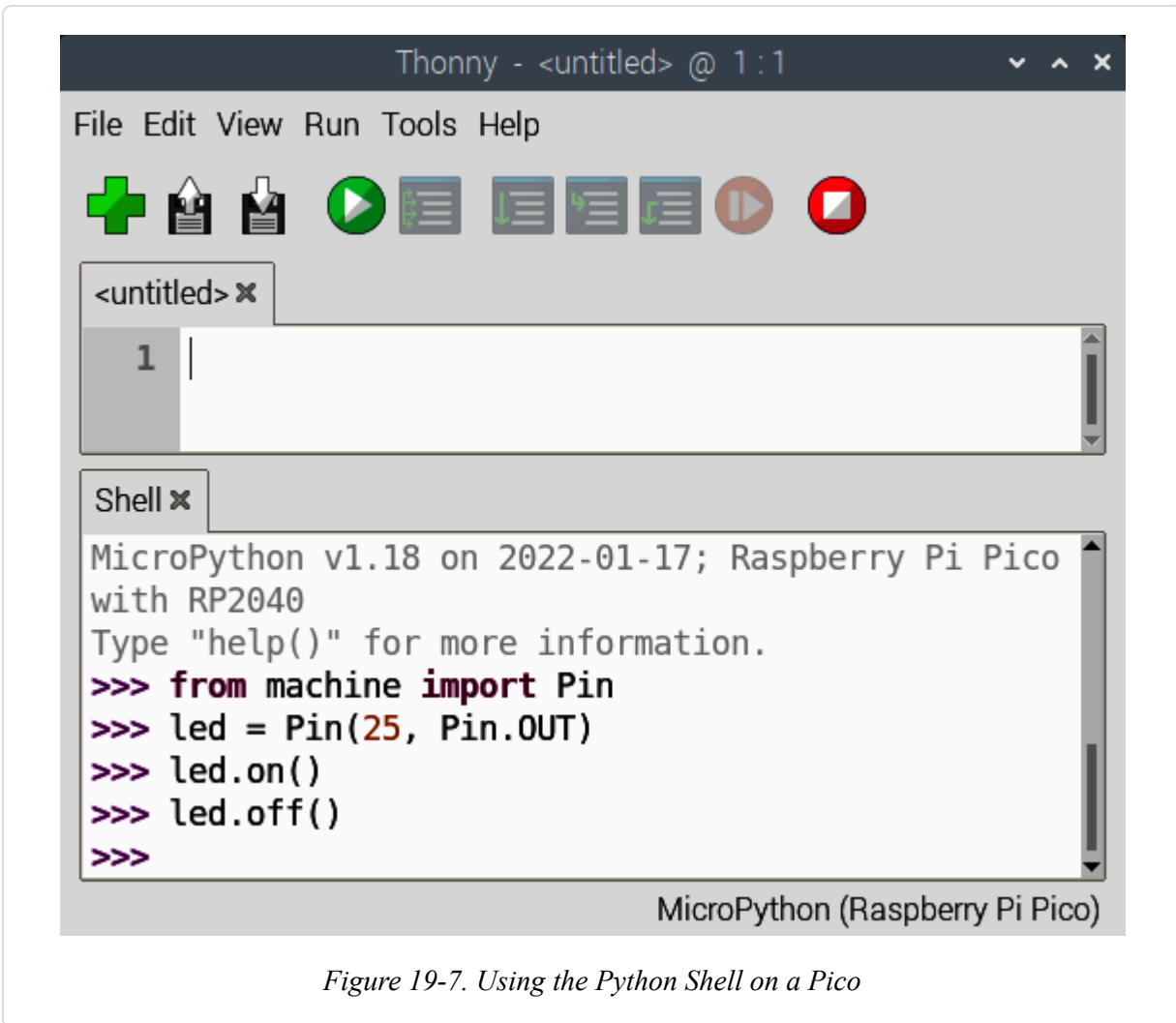
### Problem

Having set up Thonny for the Pico and installed MicroPython onto your Pico or Pico W, you want to interact with it using the Python Shell.

### Solution

As soon as Thonny has uploaded MicroPython onto your Pico, you should see the shell area appear on the bottom half of the Thonny window ([Figure 19-7](#)).

As the `>>>` prompt implies, this is a Python command line, where you can type any Python you like and it will run. The important point is that it will run on the Pico, not on your regular Raspberry Pi.



*Figure 19-7. Using the Python Shell on a Pico*

The Pico has a built-in LED connected to GPIO pin 25. We can turn this on and off using the Python commands shown in [Figure 19-7](#). First we must import the `Pin` class from the `machine` module. We can then create a new instance of `Pin` assigned to pin 25 and specify that it's an output (`Pin.OUT`). Finally, we can use the methods `on` and `off` to control the LED.

## Discussion

The `machine` module is a MicroPython module that contains the interface to the actual hardware of the Pico. This looks a little different from `gpiozero`, the module use to control the pins of a regular Raspberry Pi.

For comparison, [Table 19-1](#) shows the code we have just written for the Pico, next to the equivalent in `gpiozero` taken from [Recipe 11.1](#).

*Table 19-1. MicroPython on Pico vs. gpiozero on Raspberry Pi*

MicroPython	GPIOZero
<code>from machine import Pin</code>	<code>from gpiozero import LED</code>
<code>led = Pin(25, Pin.OUT)</code>	<code>led = LED(18)</code>
<code>led.on()</code>	<code>led.on()</code>
<code>led.off()</code>	<code>led.off()</code>

As you can see, the approach is very similar. The main difference is that `gpiozero` uses `LED` to represent the pin and the direction of the pin (output) is inferred rather than having to be explicitly stated when the pin is declared, which you have to do for the MicroPython code for the Pico.

Since the code for this chapter is MicroPython rather than the normal Python that we have used everywhere else in this book, you will find it in a separate directory called *pico* in the book's code download (see [Recipe 3.22](#)). This is where you'll find MicroPython programs to open in Thonny and run.

## See Also

More information is available in the [MicroPython documentation](#).

## 19.3 Using a Pico with a Breadboard

### Problem

You want to use your Pico with a solderless breadboard.

### Solution

Buy a breadboard kit such as the [MonkMakes Electronics Kit 1 for Pico](#) or the [Kitronik Discovery Kit](#). These kits both include a solderless breadboard, jumper wires, and some components to get you started. The breadboard included in the MonkMakes kit is labeled with the Pico's pin names (see [Figure 19-8](#)), making it easier to connect things.

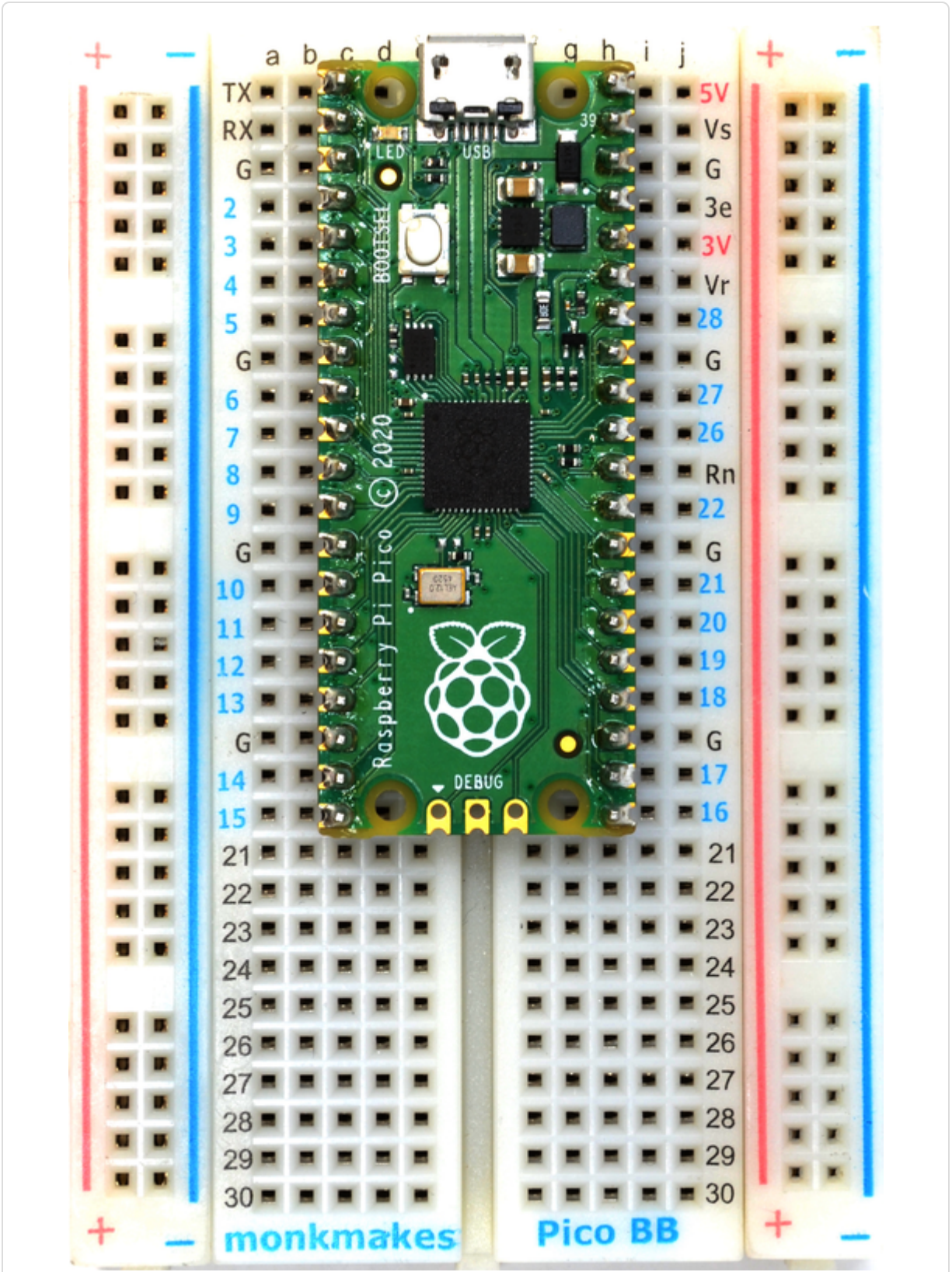


Figure 19-8. Custom breadboard with the Pico pins labeled

## Discussion

Buying a breadboard kit will save you time and probably money over buying separate components. Buying a breadboard with the Pico's pin names written on it is particularly helpful.

**Figure 19-9** shows the pinout of the Raspberry Pi Pico. This is the same for the Pico W.

The GPIO pins are labeled “GP” followed by a number. Many of the pins have secondary functions, such as interfacing to serial connections of various types (UART, I2C, and SPI). In addition, some of the pins are also capable of being used as analog inputs (**Recipe 19.7**). So, when connecting simple electronics like LEDs and switches, I tend to start with the otherwise unused pins GPs 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 21, and 22.

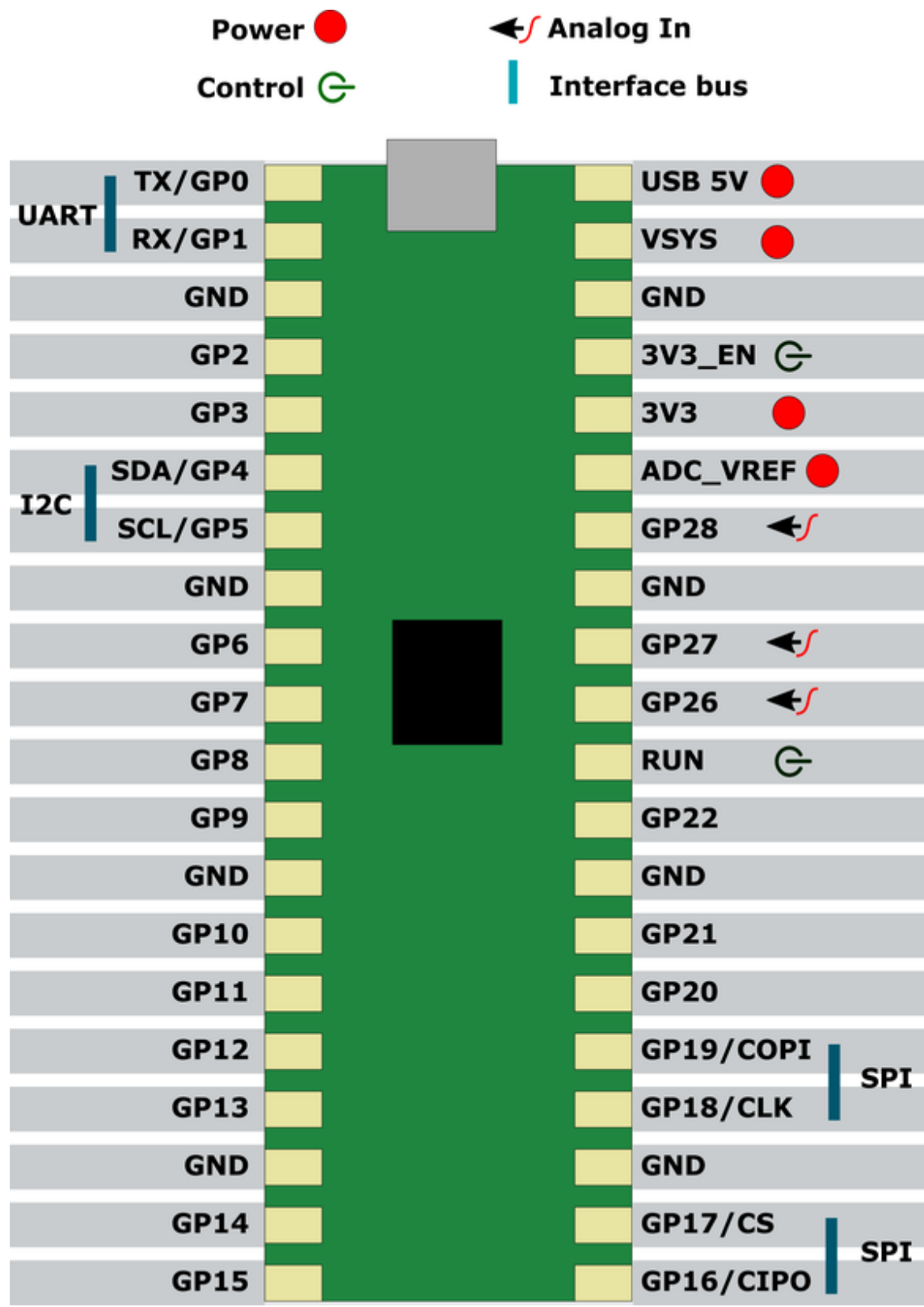


Figure 19-9. The Raspberry Pi Pico and Pico W pinout



## See Also

For using a breadboard with a regular Raspberry Pi, see [Recipe 10.9](#).

Many of the recipes from Chapters 9 to 14 are also possible using a Pico, so you may find it useful to refer back to these chapters.

## 19.4 Using Digital Outputs on a Pico

### Problem

You want to use a digital output, perhaps to drive an LED, on the Raspberry Pi Pico or Pico W.

### Solution

Use the `Pin` class in the `machine` module of MicroPython and don't draw more current than the total budget of 50mA for all pins. 1mA (milliamp) is one-thousandth of an ampere, the unit of current.

Referring back to [Figure 19-9](#), all of the GPIO pins can be used as digital outputs, although pins 12, 13, 14, 15, and 16 have the advantage of being near the end of the breadboard with more free rows nearby if the Pico is positioned at the top of the breadboard.

We are going to attach an LED to pin 16 of the Pico (see [Figure 19-10](#)). The G (ground or GND) pin of the Pico is connected to the negative supply rail of the breadboard, which is also connected to the negative shorter lead of the LED. The positive end of the LED is connected to a 470 $\Omega$  resistor that limits the current back to pin 16.

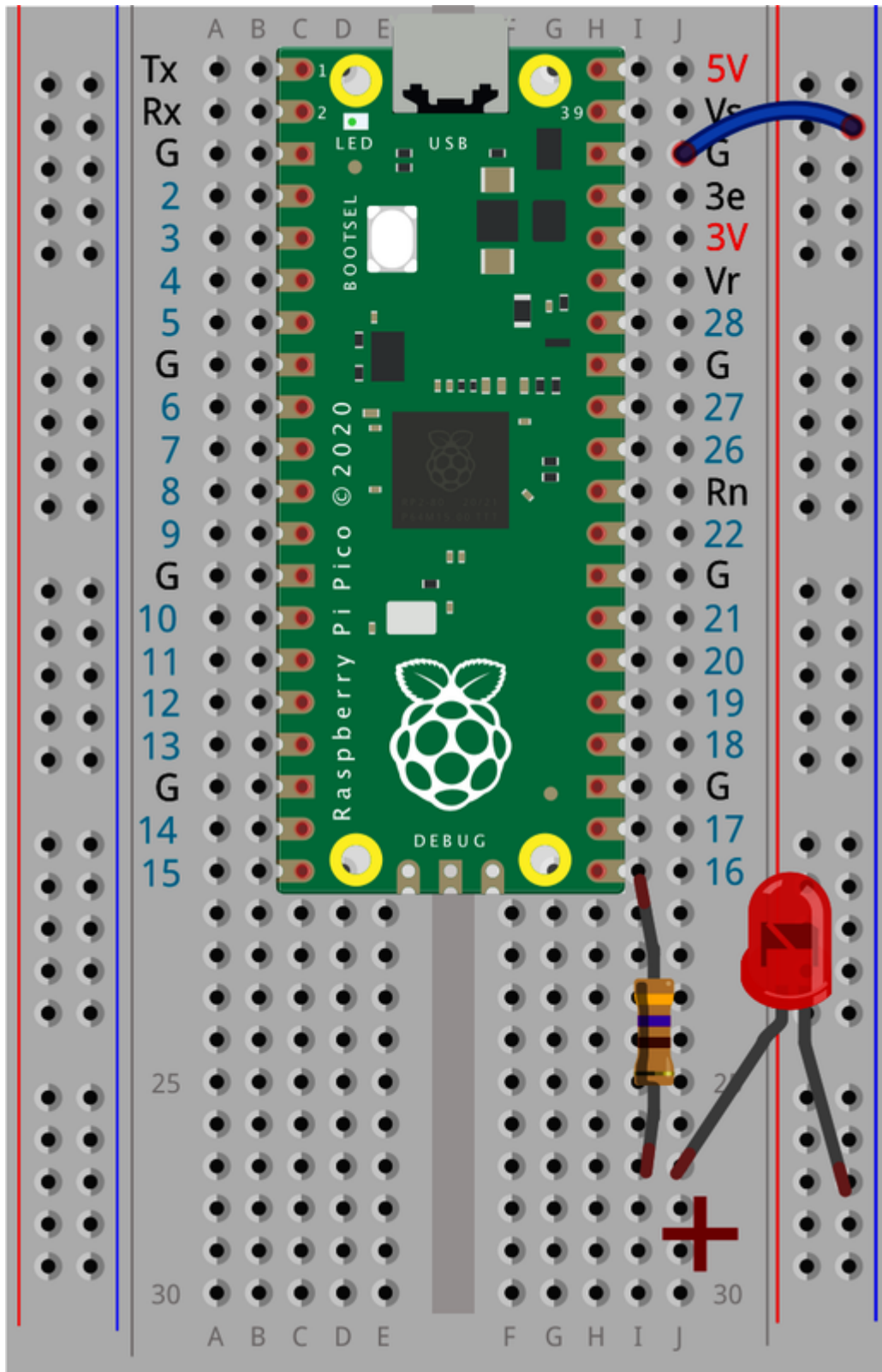


Figure 19-10. Connecting an LED to a Pico with a series resistor

As you saw in [Recipe 19.2](#), the `Pin` class contains the methods that you need both to set a pin to be a digital output and to switch it on and off. So the following code that you will find as `ch_19_blink.py` in the `pico` folder should be familiar:

```
from machine import Pin
from utime import sleep
led = Pin(16, Pin.OUT)
while True:
    led.on()
    sleep(0.5) # pause
    led.off()
    sleep(0.5)
```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

In MicroPython, the `sleep` function is in the `utime` module rather than the `time` module used in regular Python. However, it does the same job here of pausing for half a second.

## Discussion

The total current budget for all GPIO pins is 50mA. That is, when you add up all the currents used by things connected to the GPIO pins, it should not exceed 50mA. LEDs should not be connected to a GPIO pin without a series resistor that sets the current that the LEDs will draw. Without such a resistor, an LED might draw too much current and destroy the Pico, or at least that GPIO pin. Most indicator LEDs are designed to shine at close to their maximum brightness at around 15mA. So you could connect three LEDs drawing 15mA (for a total of 45mA) and still have a little current budget to spare. However, modern high-brightness LEDs work just fine at 5mA and even passably well at 1mA.

The current  $I$ , drawn by an LED connected with a series resistor as shown in [Figure 19-10](#), is determined by the formula:

$$I = \frac{3.3 - V_f}{R}$$

Where  $V_f$  is the forward voltage of the LED. This depends mostly on the color of the LED, with a red LED typically having a  $V_f$  of about 2V.  $R$  is the value of the resistor in  $\Omega$  (ohms).

For example, if you use a red LED with a  $V_f$  of 2V and a resistor of 470 $\Omega$ , the current will be:

$$(3.3 - 2) / 470 = 0.00276 \text{ A} = 2.76\text{mA}$$

Resistors come in standard values, with common ones being 100 $\Omega$ , 270 $\Omega$ , 470 $\Omega$ , and 1k $\Omega$  (1,000 $\Omega$ ). To complicate things further,  $V_f$  is usually the  $V_f$  at 20mA and this actually decreases a little at lower currents. [Table 19-2](#) will provide you with a useful reference for choosing series resistors without having to do the math.

*Table 19-2. LED series resistor selector*

LED color	Series resistor value	Approximate current
Red ( $V_f = 1.8$ )	100 $\Omega$	15mA
Red ( $V_f = 1.8$ )	270 $\Omega$	6mA
	470 $\Omega$	3mA
Red, Orange ( $V_f = 2.0$ )	100 $\Omega$	13mA
	270 $\Omega$	5mA
	470 $\Omega$	3mA
Yellow, Green ( $V_f = 2.2$ )	100 $\Omega$	11mA
	270 $\Omega$	4mA
Blue, White ( $V_f = 3.5$ ) <sup>a</sup>	100 $\Omega$	4mA <sup>a</sup>

<sup>a</sup> Although theoretically a lower supply voltage than forward voltage means that you could use the LED without a resistor, this could still draw too much current, so say a 100 $\Omega$  resistor is a good idea even with blue and white LEDs.

## See Also

For more information on connecting an LED to GPIO pins, see [Recipe 11.1](#).

Many online calculators are available for working our values of series resistor. [This](#) is a good one.

Resistors like the ones used here with the breadboard have [colored stripes](#) that indicate their resistance value.

## 19.5 Using Digital Inputs on a Pico

### Problem

You want to be able to connect a switch, or other digital input, to a Raspberry Pi Pico or Pico W.

### Solution

Connect your switch (or other source as a digital input) to your Pico and use the `Pin` class to set the pin to act as an input and to read its value.

Let's start by connecting a switch between one of the Pico's GND pins and pin 16. We can do this on a breadboard, as shown in [Figure 19-11](#).

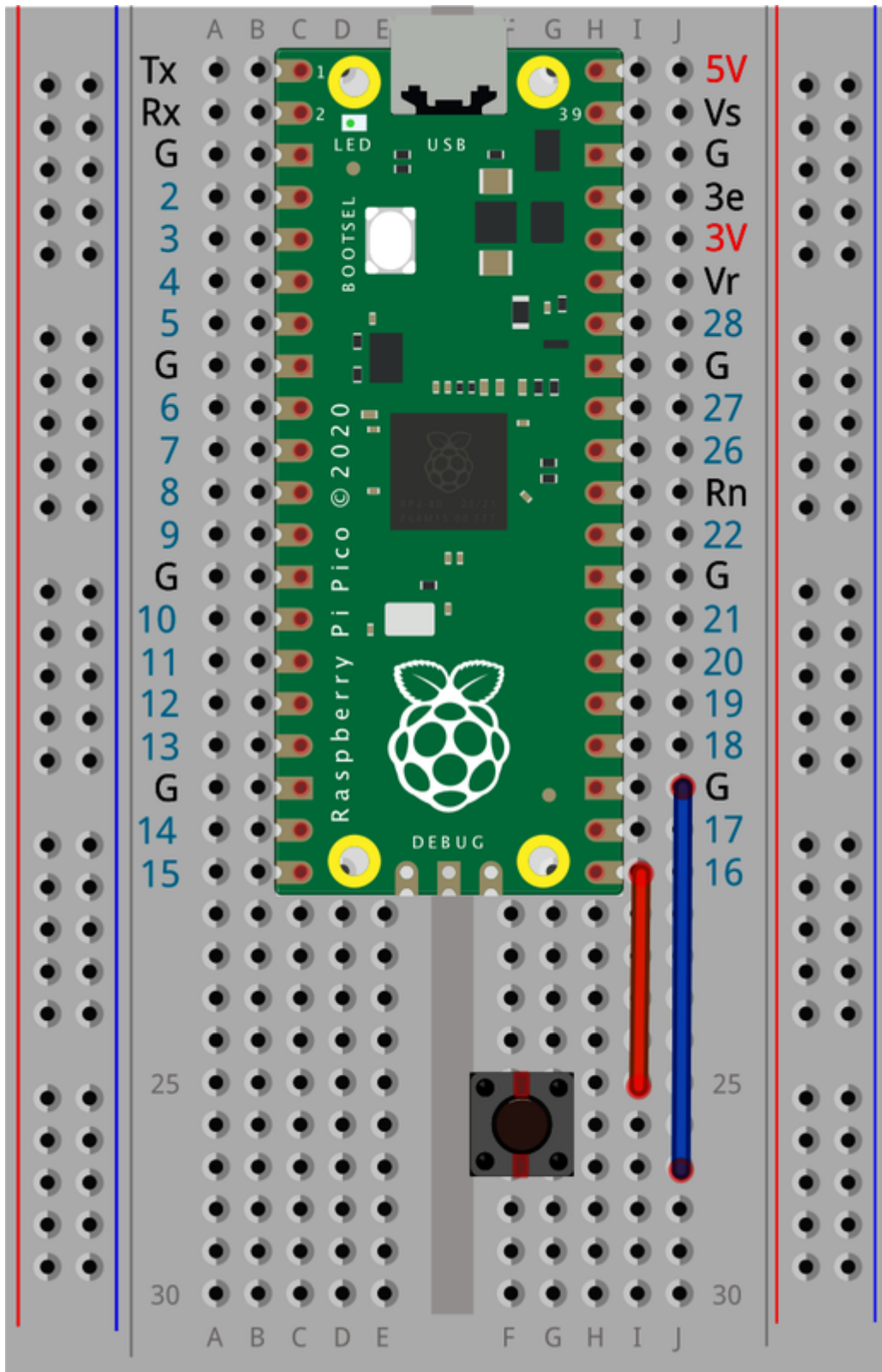


Figure 19-11. Connecting a switch to a Pico

Note that the tactile push switch shown in [Figure 19-11](#) is a two-pin version. If your switch has four pins, then it can go on the same row, but will need to span the two halves of the breadboard.

The program `ch_19_digital_input.py` repeatedly reads the digital input of pin 16, displaying either 1 or 0, depending on whether the switch is pressed or not. Open it in Thonny and try running it:

```
from machine import Pin
from utime import sleep

switch = Pin(16, Pin.IN, Pin.PULL_UP)

while True:
    print(switch.value())
    sleep(0.1)
```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

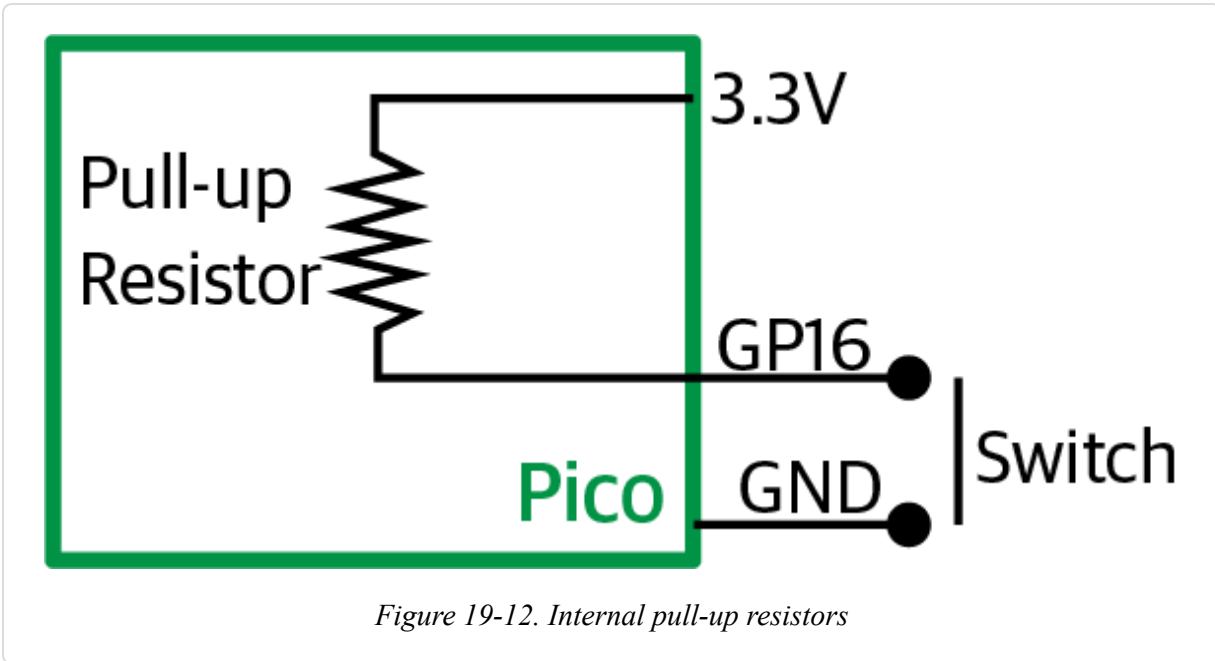
## Discussion

In the preceding example, when the variable `switch` is assigned an instance of `Pin`, the call to create a new instance of `Pin` has three parameters:

- the pin number
- the direction
- `Pin.PULL_UP`

The last parameter does something clever inside the circuitry of the Pico's GPIO pin. It turns on a transistor that connects an internal pull-up resistor ([Figure 19-12](#)). This resistor, which has a value of about 20kΩ, pulls the GPIO pin up to about 3.3V. Without this resistor, the GPIO pin would be said to be *floating*. This means that the pin will act like an antenna picking up electrical noise, and will flip between on and off more or less at random, which is not behavior you want in a switch. There is nothing special about GP16—all of the Pico's GPIO pins have this feature.

When the switch is pressed, GP16 is connected to GND (0V). This massively overrides the weak pull-up resistor, taking GP16 from high to low. That is why program `ch_19_digital_input.py`, rather counterintuitively, displays a 0 when the switch button is pressed and a 1 when it is released.



## See Also

To connect a switch to the GPIO pin of a regular Raspberry Pi, see [Recipe 13.1](#).

## 19.6 Using Analog (PWM) Outputs on a Pico

### Problem

You want to vary the brightness of an LED from a Python program on your Pico or Pico W.

### Solution

Use the PWM (pulse-width modulation) capability of the Pico to control the average power supplied to the LED.



To try this out, you can connect an external LED to the Pico as described in [Recipe 19.4](#) or, if you prefer, use the Pico's built-in LED. If you decide to use the built-in LED, remember to change the pin from 16 to 25 in the following example program, which you will find in the file `ch_19_pwm.py`:

```
from machine import Pin, PWM
from utime import sleep

led = PWM(Pin(16))

while True:
    brightness_str = input("brightness (0-65534):")
    brightness = int(brightness_str)
    led.duty_u16(brightness)
```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

Run the program in Thonny. In the Python console, you will be prompted to enter a value of brightness between 0 (off) and 65534 (maximum brightness).

## Discussion

PWM provides pulses of varying duration to the LED to control its apparent brightness. You will find a proper explanation of this in [Recipe 11.3](#).

To make an output pin into a PWM output pin, you just wrap it in the `PWM` class with `PWM(Pin(16))`. You can then change the duty cycle of the output by calling the rather unfriendly sounding `duty_u16` method. The `u16` part means an unsigned 16-bit number. The maximum value of an unsigned 16-bit number is 65535, not 65534, but the Pico's official documentation states that the maximum duty value should be the latter, so that's what we use here.

If you have used PWM on a regular Raspberry Pi, you may have noticed that the LED brightness might change a little. This unreliability is because PWM on a regular Raspberry Pi is implemented in software and, every so often, Raspberry Pi OS will stop generating pulses for PWM and go off and

do other operating system things. The great advantage of PWM on a Pico is that the Pico implements PWM in hardware dedicated just to generating the pulses and is therefore rock-solid.

## See Also

PWM on a regular Raspberry Pi is described in [Recipe 11.3](#).

## 19.7 Using Analog Inputs on a Pico

### Problem

You want to read an analog voltage using MicroPython on your Pico or Pico W.

### Solution

Connect a voltage source of between 0 and the 3.3V supply to one of the analog-capable GPIO pins (26, 27, and 28) and use the Pico's ADC (analog-to-digital converter) to measure the voltage.

To illustrate this recipe, we'll use a trimpot (a potentiometer or variable resistor) to provide a voltage between 0 and 3.3V to pin 26, depending on the position of its knob. [Figure 19-13](#) shows how this can be wired-up on a breadboard. Any value of pot between 1k $\Omega$  and 100k $\Omega$  will work just fine. A small trimpot, such as the one supplied in the MonkMakes Electronics Kit 1 for Pico, will fit nicely on the breadboard.

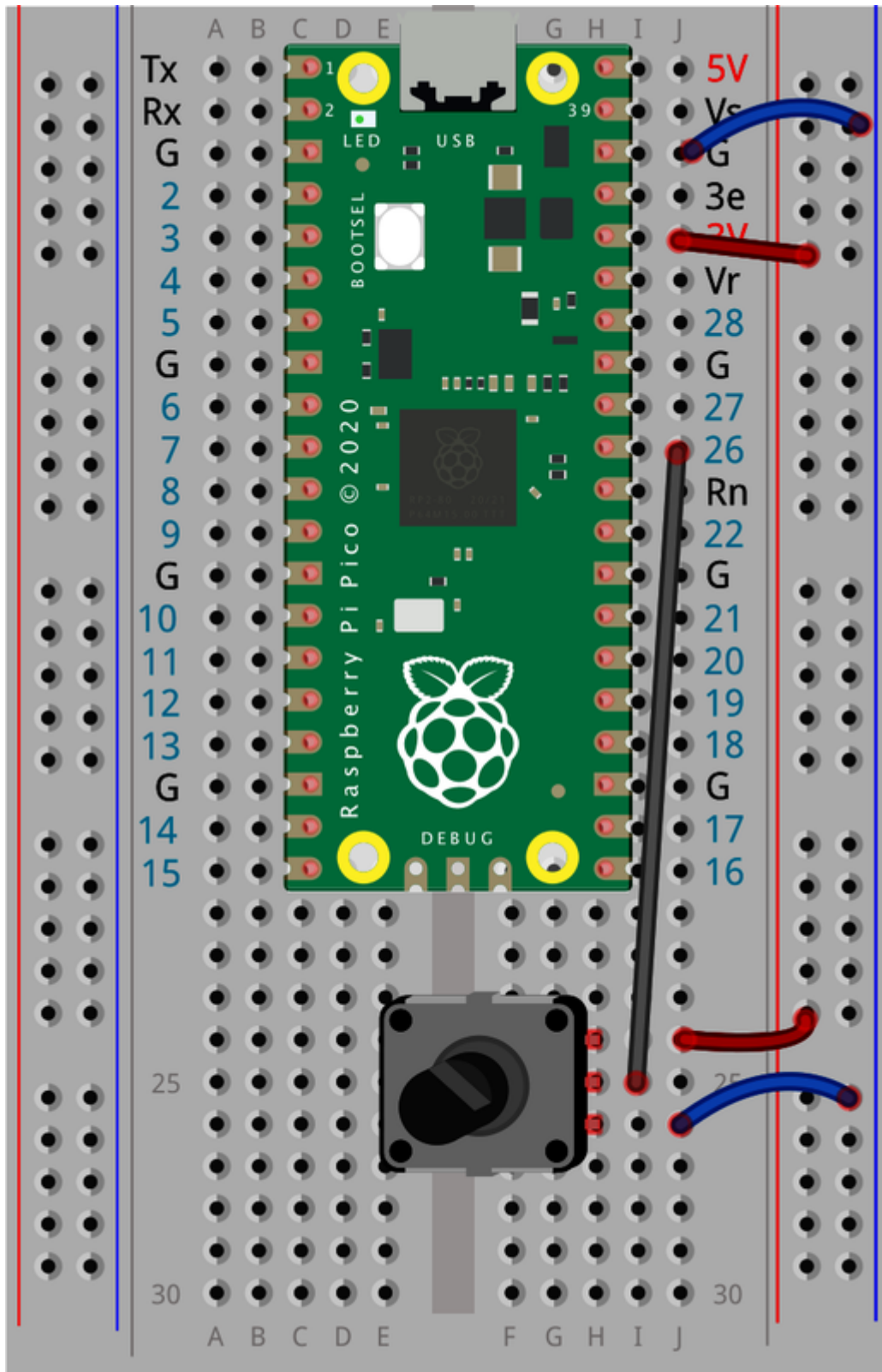


Figure 19-13. Measuring voltage with a Pico

Load the program `ch_19_voltmeter.py` into Thonny and run it. You should see a series of voltage readings. Try rotating the knob of the pot and you should see the voltage change:

```
from machine import ADC, Pin
from utime import sleep

analog = ADC(26)

def volts_from_reading(reading):
    min_reading = 336
    max_reading = 65534
    reading_span = max_reading - min_reading
    volts_per_reading = 3.3 / reading_span
    volts = (reading - min_reading) * volts_per_reading
    return volts

while True:
    reading = analog.read_u16()
    print(volts_from_reading(reading))
    sleep(0.5)
```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

## Discussion

To read analog inputs rather than the digital inputs of [Recipe 19.5](#), you have to wrap the pin in the ADC class. You can then use `read_u16` to get an analog value between 0 and 65534 for the pin in question.

Even though the value returned is an unsigned 16-bit number, the resolution of the ADC is only 12 bits, providing 4096 different possible values. The function `volts_from_reading` converts the raw analog reading into a voltage. The ADC does not quite measure from 0 all the way to 3.3V. There are dead zones. So, even if the analog input is connected to GND (0V), the ADC will read a value of roughly 336. In the function `volts_from_reading`, this is taken to be the `min_reading`, and the `max_reading` is taken to be 65534 (as discussed earlier). The span of readings is therefore `max_reading - min_reading`, and so the

number of `volts_per_reading` can be calculated. From this, the voltage at the input can be calculated.

You can't use all of the GPIO pins for analog inputs; this feature is available only on pins 26, 27, and 28. However, analog channel 4 is connected internally on the RP2040 chip to an analog temperature sensor, which will tell you the temperature of the chip after applying a bit of math to the analog reading. You can try this out with the program *ch\_19\_thermometer.py*:

```
from machine import Pin, ADC
from utime import sleep

temp_sensor = ADC(4)
points_per_volt = 3.3 / 65535

def read_temp_c():
    reading = temp_sensor.read_u16() * points_per_volt
    temp_c = 27 - (reading - 0.706)/0.001721
    return temp_c

while True:
    temp_c = read_temp_c()
    print(temp_c)
    sleep(0.5)
```

Note that this is analog channel 4—not pin 4—so you can still use pin 4 as normal.

The function `read_temp_c` takes an analog reading using `read_u16` from the temperature sensor and then applies a bit of math to it, resulting in the temperature in degrees C. The numeric constants are taken from the datasheet for the Pico's RP2040 processor.

This sensor reports the temperature of the processor rather than the environment, but, even so, if you place your finger on the processor chip it will warm it, and the readings should rise.

The presence of analog inputs opens up the possibility of attaching all sorts of analog sensors to your Pico, including sensors for light, temperature,

mechanical stress, and even gas sensors. Many of the recipes in [Chapter 14](#) can be easily adapted to work with the Pico.

## See Also

To add analog inputs to a Raspberry Pi, see [Recipe 14.7](#).

Take a look at the [datasheet for the RP2040](#).

## 19.8 Controlling a Servomotor from a Pico

### Problem

You want to control servomotors using your Pico or Pico W.

### Solution

Connect power to the servomotor from the Pico's 5V supply and connect one of the GPIO pins to the servomotor's control pin. Then use the PWM class to generate pulses to set the servomotor's angle.

Connect the servomotor using male-to-male jumper wires as shown in [Figure 19-14](#). Servomotors have different color schemes used to identify the leads. In [Figure 19-14](#), black is ground, red is 5V, and yellow is the control pin. Another typical color scheme is brown for ground, red for 5V, and orange for the control wire. Check the datasheet for your servomotor to find the right pins.

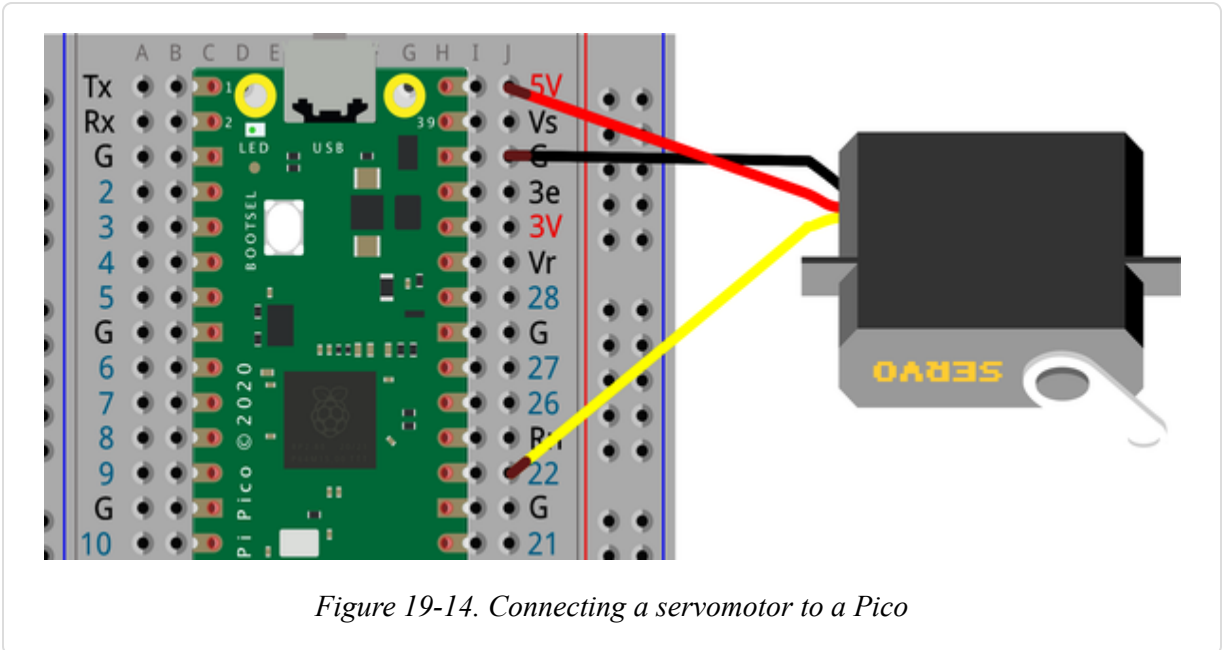


Figure 19-14. Connecting a servomotor to a Pico

Open `ch_19_servo.py` in Thonny and run it. The shell will prompt you to enter an angle between 0 and 180 (Figure 19-15). When you press Enter, the servomotor's arm should move to the new position.

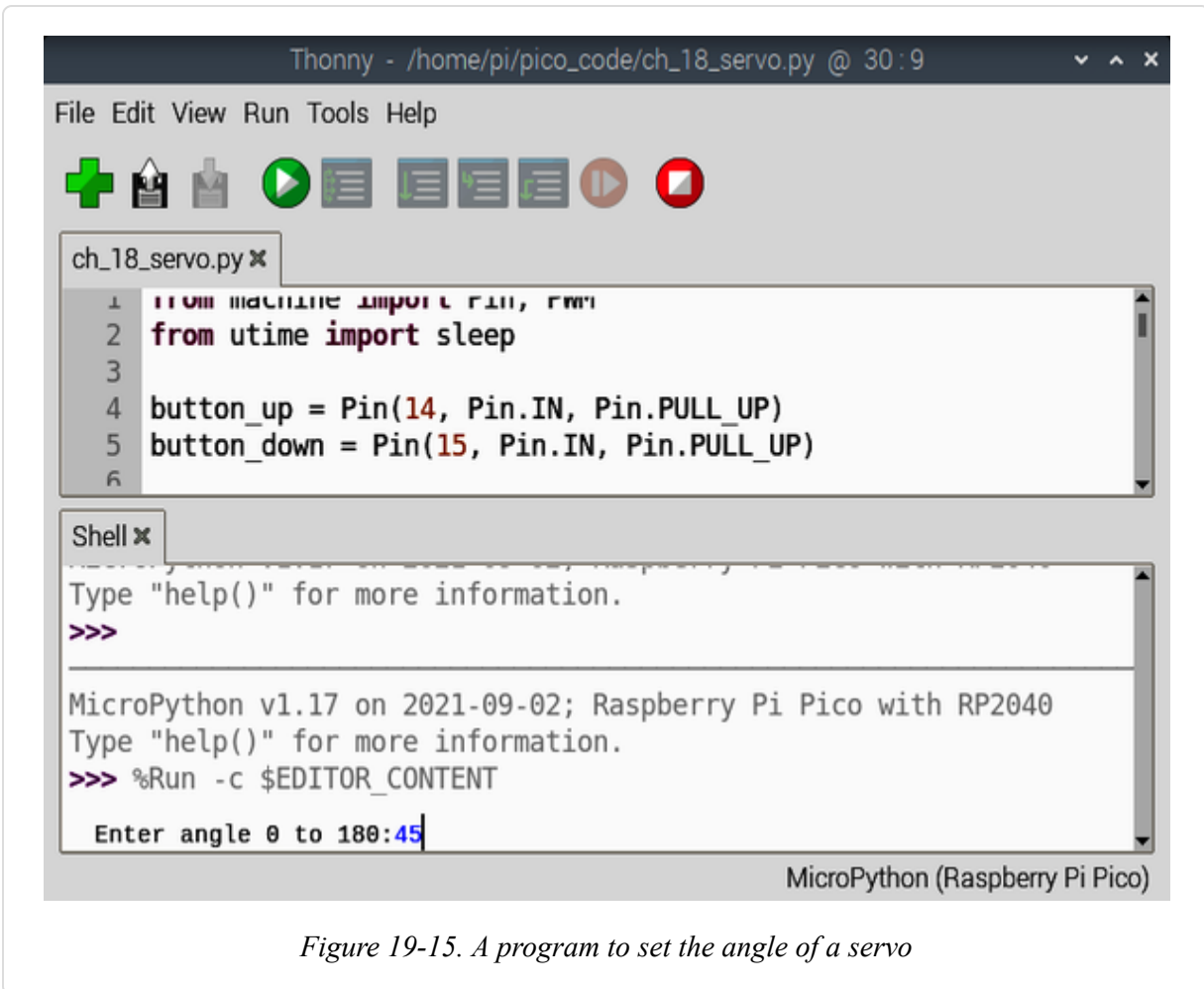


Figure 19-15. A program to set the angle of a servo

## Discussion

The code listing for `ch_19_servo.py` is:

```
from machine import Pin, PWM
from utime import sleep

servo = PWM(Pin(16))
servo.freq(50) # pulse every 20ms

def set_angle(angle, min_pulse_us=500, max_pulse_us=2500):
    us_per_degree = (max_pulse_us - min_pulse_us) / 180
    pulse_us = us_per_degree * angle + min_pulse_us
    # duty 0 to 1023. At 50Hz, each duty_point is 20000/65535 =
    0.305
    us/duty_point
    duty = int(pulse_us / 0.305)
```



```

    # print("angle=" + str(angle) + " pulse_us=" + str(pulse_us)
+ " duty="
    + str(duty))
    # print(angle)
    servo.duty_u16(duty)

angle = 90
set_angle(90)
min_angle = 10
max_angle = 160

while True:
    angle_str = input("Enter angle 0 to 180:")
    angle = int(angle_str)
    if (angle >= 0 and angle <=180):
        set_angle(angle)

```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

Servomotors are controlled by changing the duration of pulses on a control connection of the servo. You might need to read through the Discussion of [Recipe 12.1](#), which explains how servomotors work.

The default PWM frequency is too high for servomotors, which expect a new pulse every 20 milliseconds (50 times a second), so after specifying PWM on pin 16, the PWM frequency is then set to 50 Hz using `servo.freq(50)`.

All the code for calculating the PWM duty is contained in the function `set_angle`. As you would expect, this takes a parameter of the angle itself, but also two optional parameters, `min_pulse_us` and `max_pulse_us`, that set the minimum and maximum pulse lengths in microseconds (hence `_us`). These can be tweaked to match your servomotor to either give it maximum angular range or to prevent judder at either end of its travel. This kind of juddering can occur if the pulses are too short or too long.

The `set_angle` function first finds the number of microseconds per degree from the range of pulse lengths. It then calculates the length of the

pulse needed in microseconds and finally converts this into a value of `duty`.

Using a Pico to control a servo works a lot better than using a regular Raspberry Pi. The precise timing of the Pico's PWM output means that you shouldn't see any twitchiness.

## See Also

For controlling servomotors on a regular Raspberry Pi, and more about servomotors in general, see [Recipe 12.1](#).

# 19.9 Using the Pico and Pico W's Filesystem

## Problem

You want to store and retrieve data from the Pico's filesystem.

## Solution

Use the MicroPython's file commands to read, write, and create files that are stored on the Pico and can be copied over to your Raspberry Pi.

As an example, the code in `ch_19_temp_logger.py` will read the RP2040's temperature in degrees C and log it to a file every 10 seconds.

When you run the program, a new temperature reading appears in the Shell. When you have recorded enough, press the Stop button or press Ctrl-C to finish the logging, close the file, and exit the program.

To see the file that has been written (`temp_reading.txt`), you need to go to the View menu of Thonny and select Files ([Figure 19-16](#)). This will open a column on the left of the Thonny window listing any files on the Pico. You can double-click on `temp_readings.txt` to open it in the Thonny editor, where you could, if you wanted, edit the file and then save it back to the Pico. You can also transfer the file to your Raspberry Pi by right-clicking on

the filename on the left and selecting the pop-up menu option “Download to /home/pi.”

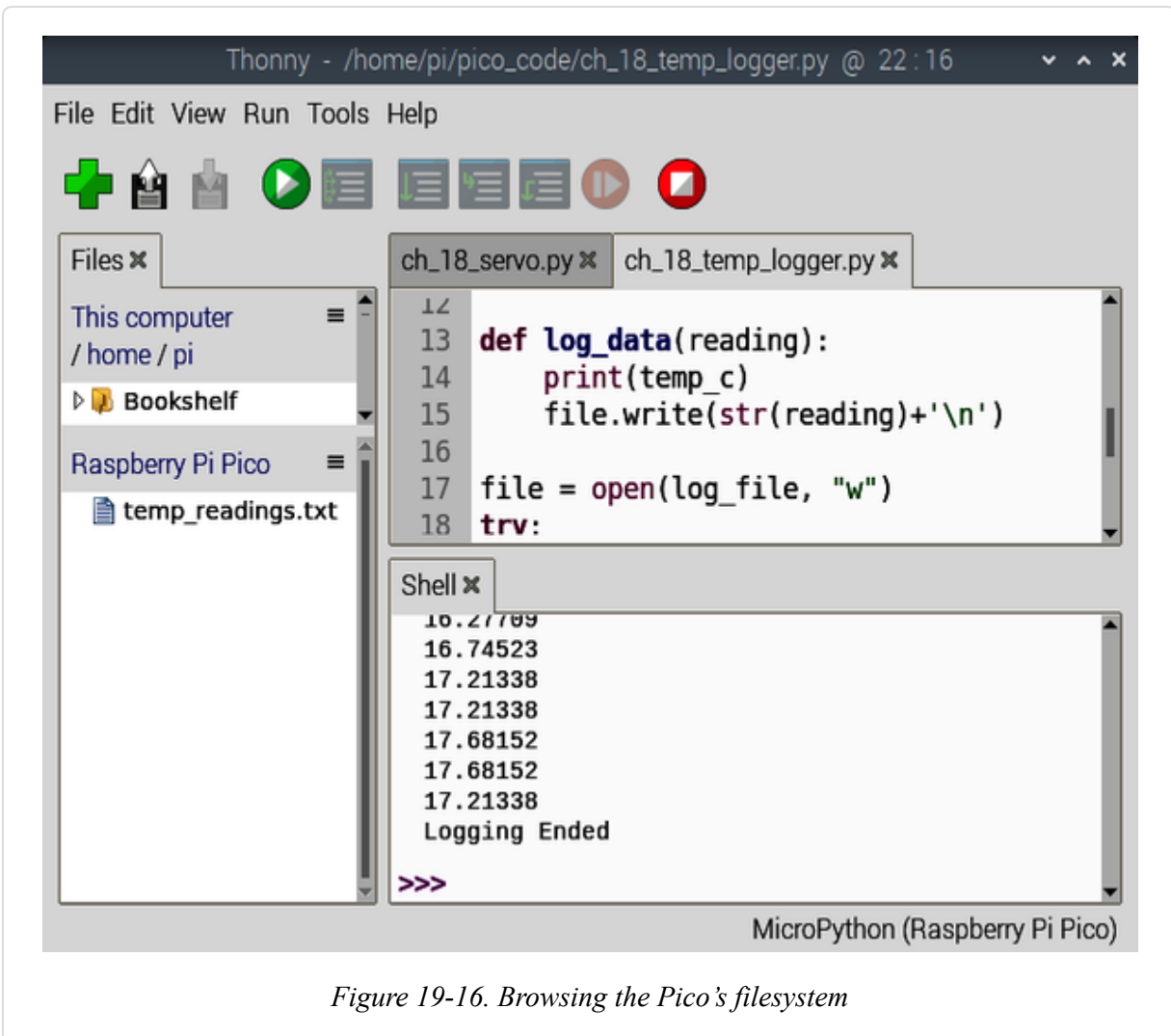


Figure 19-16. Browsing the Pico's filesystem

## Discussion

The code for the data logger program (*ch\_19\_temp\_logger.py*) is:

```
from machine import Pin, ADC
from utime import sleep

log_file = 'temp_readings.txt'
temp_sensor = ADC(4)
points_per_volt = 3.3 / 65535
```

```

def read_temp_c():
    reading = temp_sensor.read_u16() * points_per_volt
    temp_c = 27 - (reading - 0.706)/0.001721
    return temp_c

def log_data(reading):
    print(temp_c)
    file.write(str(reading)+'\n')

file = open(log_file, "w")
try:
    while True:
        temp_c = read_temp_c()
        log_data(temp_c)
        sleep(10)
except:
    print('Logging Ended')
    file.close()

```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)). This example is built around the code used in the Discussion of [Recipe 19.7](#), with added code to write the temperature readings to a file.

You don't have to import any modules to use the filesystem; it's part of MicroPython and is essentially a cut-down version of the Python 3 filesystem described in [Chapter 7](#). The file is opened for writing using the command `open`, which takes the filename as its first parameter and the mode (`w` for write) as its second. This will replace any existing file with the same name.

The `log_data` function prints out the supplied reading and then appends it to the file after converting it to a string and adding the newline (`\n`) character to the end.

The main `while` loop is surrounded by a `try/ except` so that when you do `Ctrl-C` in the Shell, this will be caught and the file closed.

When it comes to reading a file, you need to open the file in read (`r`) mode, and then you can read the contents of the file as a string using the `read` function. You can find an example of this in the program

`ch_19_file_read.py`. This program will just print out the contents of a file in the Shell:

```
f = open("temp_readings.txt", "r")
print(f.read())
f.close()
```

The capacity of the filesystem of the Pico is limited to just 1.6 MB, so you are not going to be storing video on it, or even any large sound samples. The other aspect of the filesystem to be aware of is that if you have to reinstall your Pico's firmware as you did in [Recipe 19.1](#), you will lose the contents of the filesystem.

## See Also

For writing to a file with a regular Raspberry Pi, see [Recipe 7.7](#), and for reading from a file, see [Recipe 7.8](#).

## 19.10 Making Use of the Second Core

### Problem

You want to use the Pico or Pico W's second core (processor) to do more than one thing at a time.

### Solution

Use the MicroPython *thread* mechanism to run code in parallel with the main thread of execution. You will find an example of this in the program `ch_19_multicore.py`:

```
from utime import sleep
import _thread
from random import randint

def core0():
```

```

while True:
    print("core 0 says hello")
    sleep(randint(1, 3))

def core1():
    while True:
        print("core 1 says hello")
        sleep(randint(1, 3))

_thread.start_new_thread(core1, ( ))
core0()

```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

When you run this program, you'll see output something like this in the Shell:

```

core 1 says hello
core 1 says hello
core 0 says hello
core 1 says hello
core 0 says hello
core 1 says hello
core 0 says hello

```

Each of the two threads displays a message and then waits for a random period between one and three seconds. Every so often, you might see the messages appear on the same line. This happens when both are competing for access to the USB port.

## Discussion

A good way to structure your code to handle two threads is to put the code for each thread in a function of its own. In the preceding example, I have called these functions `core0` and `core1`. Both functions contain a `while True` loop: `core1` is allocated to one physical processor core by calling `start_new_thread`, and then `core0` is started on the other processor core just by calling the `core0` function.

In regular Python 3 on a Raspberry Pi, you can create as many threads of execution as you like but, on the Pico, you are limited to just two threads.

## See Also

For information on running multiple threads in regular Python 3, see [Recipe 7.19](#).

## 19.11 Running a WiFi Web Server on the Pico W

### Problem

You want to make use of the Pico's WiFi capabilities, and what better way to demonstrate that than by running a web server on the Pico W?

### Solution

Use the `microdot` web server module and the `mm_wlan` WiFi connection modules.

For convenience, these modules are included with the test programs in the Pico section of the downloads for the book in a folder called *ch\_19\_webserver*.

Copy the files *pmon.py*, *microdot.py*, and *mm\_wlan.py* onto your Pico, by using the SaveAs menu option in Thonny and then selecting Raspberry Pi Pico for the destination. Give each file the same filename as the original file. You can see the files listed as on the filesystem in [Figure 19-17](#).

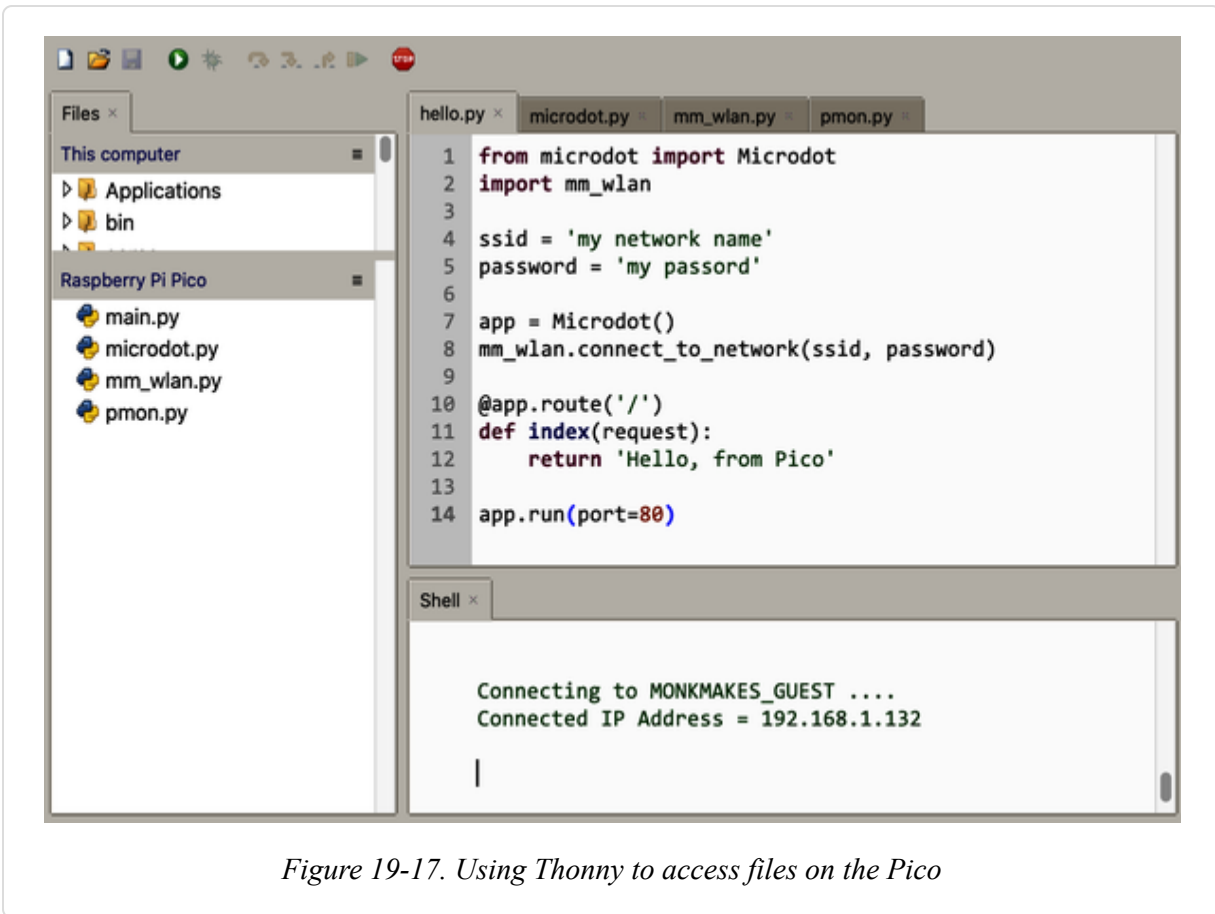
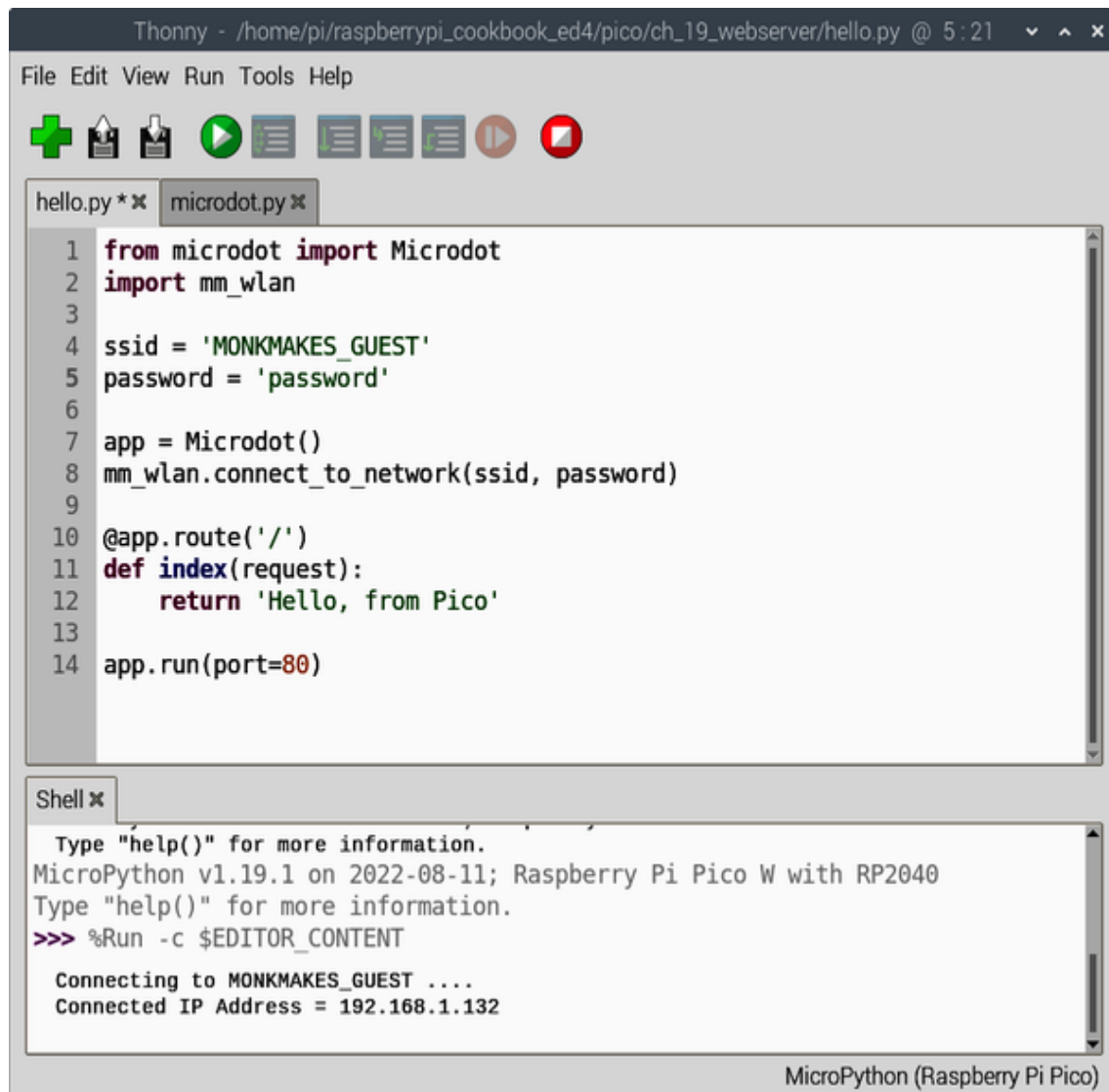


Figure 19-17. Using Thonny to access files on the Pico

Open the file *hello.py* and change `ssid` and `password` to match the name of your WiFi network and the password, then run *hello.py* from Thonny (see [Figure 19-18](#)).





The screenshot shows the Thonny IDE interface. The top menu bar includes File, Edit, View, Run, Tools, and Help. Below the menu is a toolbar with icons for file operations and execution. Two tabs are open: 'hello.py \*x' and 'microdot.py x'. The main editor area displays the following Python code:

```
1 from microdot import Microdot
2 import mm_wlan
3
4 ssid = 'MONKMAKES_GUEST'
5 password = 'password'
6
7 app = Microdot()
8 mm_wlan.connect_to_network(ssid, password)
9
10 @app.route('/')
11 def index(request):
12     return 'Hello, from Pico'
13
14 app.run(port=80)
```

The Shell area at the bottom shows the execution output:

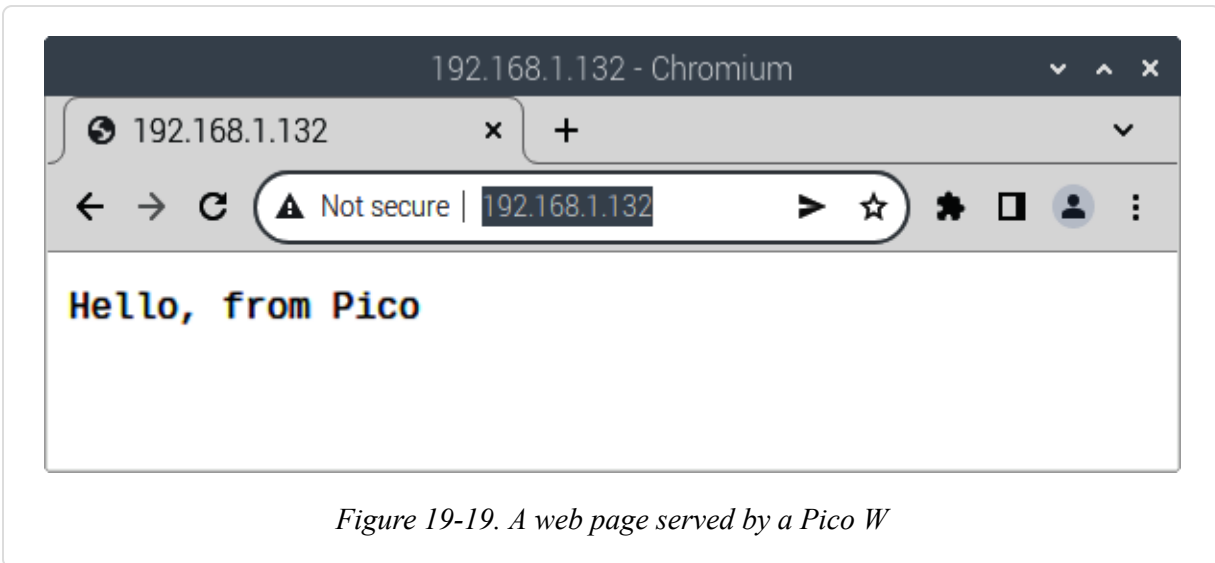
```
Type "help()" for more information.
MicroPython v1.19.1 on 2022-08-11; Raspberry Pi Pico W with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

Connecting to MONKMAKES_GUEST ....
Connected IP Address = 192.168.1.132
```

The status bar at the bottom right indicates 'MicroPython (Raspberry Pi Pico)'.

Figure 19-18. Thonny showing the program hello.py

In the Shell area of [Figure 19-18](#) you can see that the Pico W has successfully connected to WiFi and has the IP address of 192.168.1.132. If you now navigate to this address in a browser window, either on your Raspberry Pi or any other machine connected to the network, you will see something like [Figure 19-19](#).



*Figure 19-19. A web page served by a Pico W*

## Discussion

The entire program for *hello.py* is shown in [Figure 19-18](#), and, as you can see, there is not much to it.

The `@app.route('/')` specifies the handler for the root page that the Pico W is serving, and all this function does is to return the text “Hello, from Pico” to be shown on the web page.

The command `app.run(port=80)` starts the web server running on port 80, the standard web serving port.

We could expand this example to report readings from a remote sensor attached to a Pico W. As an example of this, we can use the Plant Monitor from [Recipe 14.6](#), but attach it to a Pico W rather than a regular Raspberry Pi.

Connect the Pico to the Plant Monitor as follows:

- GND to GND
- 3V to 3V
- TX on the Plant Monitor to RX on the Pico W
- RX on the Plant Monitor to TX on the Pico W

You can either connect the two devices directly using female-to-female headers, or, as shown in [Figure 19-20](#) using a breadboard and male-to-

female jumper wires.

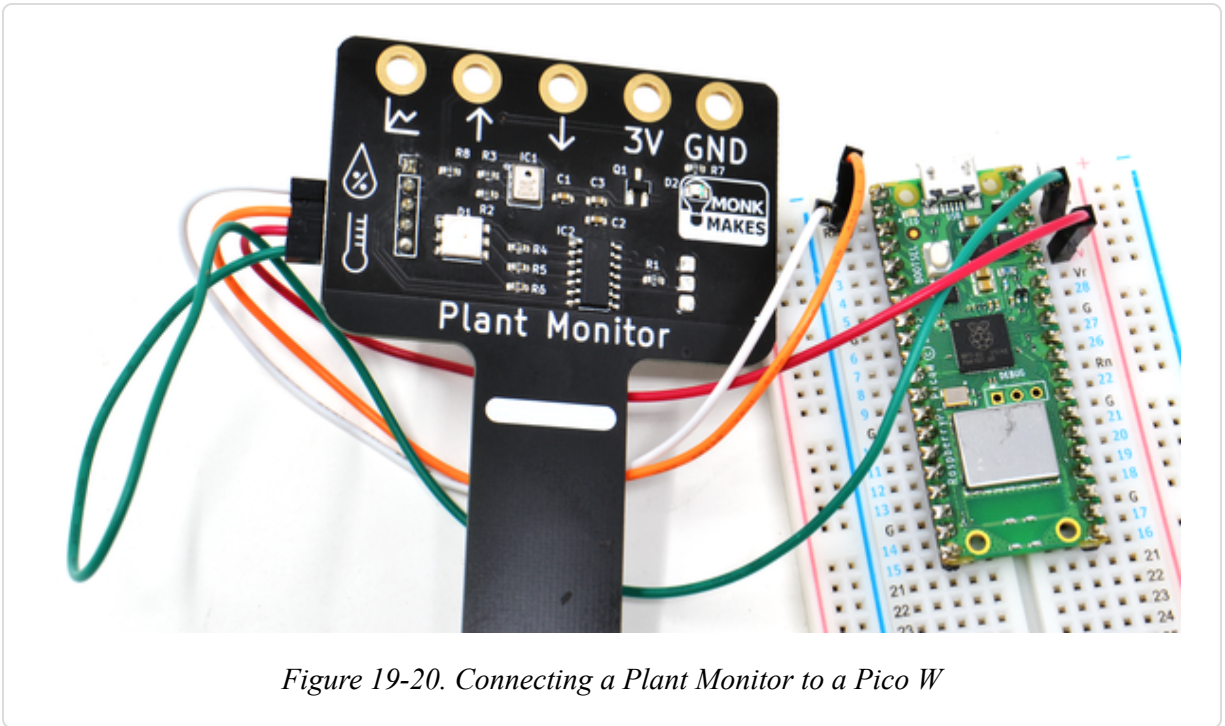
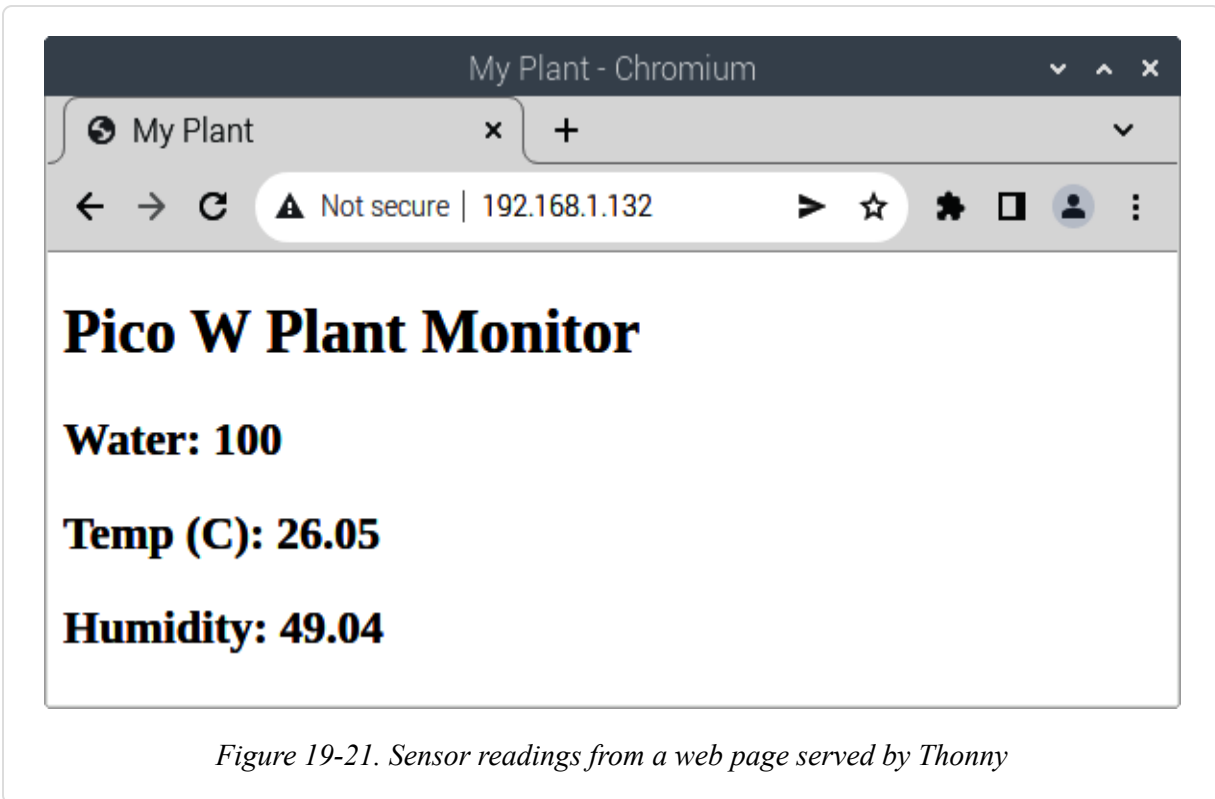


Figure 19-20. Connecting a Plant Monitor to a Pico W

Run the example `pico_w_server.py` and then refresh the browser window. You should see something that looks like [Figure 19-21](#).



*Figure 19-21. Sensor readings from a web page served by Thonny*

## See Also

More information is available on the `microdot module`, and on `mm_wlan`.

If the syntax of the `microdot module` looks familiar, then that might be because it is much the same as for the `Bottle library` used in [Recipe 7.17](#).

Find out about the Plant Monitor at <https://oreil.ly/2OBHW>.

## 19.12 Using Pico-Compatible Boards

### Problem

You'd like to make use of one of the many Pico-compatible boards.

### Solution

One purpose of the Raspberry Pi Pico is to show off the features of the RP2040 processor that it uses. Many other boards use the RP2040 and can be programmed in MicroPython using Thonny in just the same way.

**Table 19-3** lists a selection of interesting RP2040 boards available at the time of writing. An internet search will, I'm sure, reveal other RP2040 boards.

*Table 19-3. RP2040 boards*

<b>Name</b>	<b>Manufacturer</b>	<b>Notes</b>
QTPy RP2040	Adafruit	A tiny board, with reduced number of I/O pins and a USB-C connector.
Feather RP2040	Adafruit	In Adafruit's Feather format, a Pico with added LiPo battery charger and USB-C connector. Also available in pink.
PicoLiPo	Pimoroni	Very similar spec to the Feather RP2040.
Badger	Pimoroni	Minimal pins, but 5 push buttons and a large E Ink display.

## Discussion

The Pimoroni Badger is one of the most interesting RP2040 boards. It can be used as an ID badge, displaying graphics and text on its E Ink display, which will still display even when the power goes off. But it also has the full processing power of the Pico available for you to use in MicroPython.

The Badger comes with a MicroPython image installed and running an example program. To interact with it or put your own programs on it, connect it to your Raspberry Pi using a USB-C lead (not the micro USB of the Pico), and you can then interact with it as if it were a Pico.

To make it easy to access the E Ink display, Pimoroni has included a Python module to draw graphics and text on the E Ink display.

The example program in `ch_19_badger.py`, and shown in **Figure 19-22**, uses the RP2040's temperature sensor to display a rudimentary thermometer.

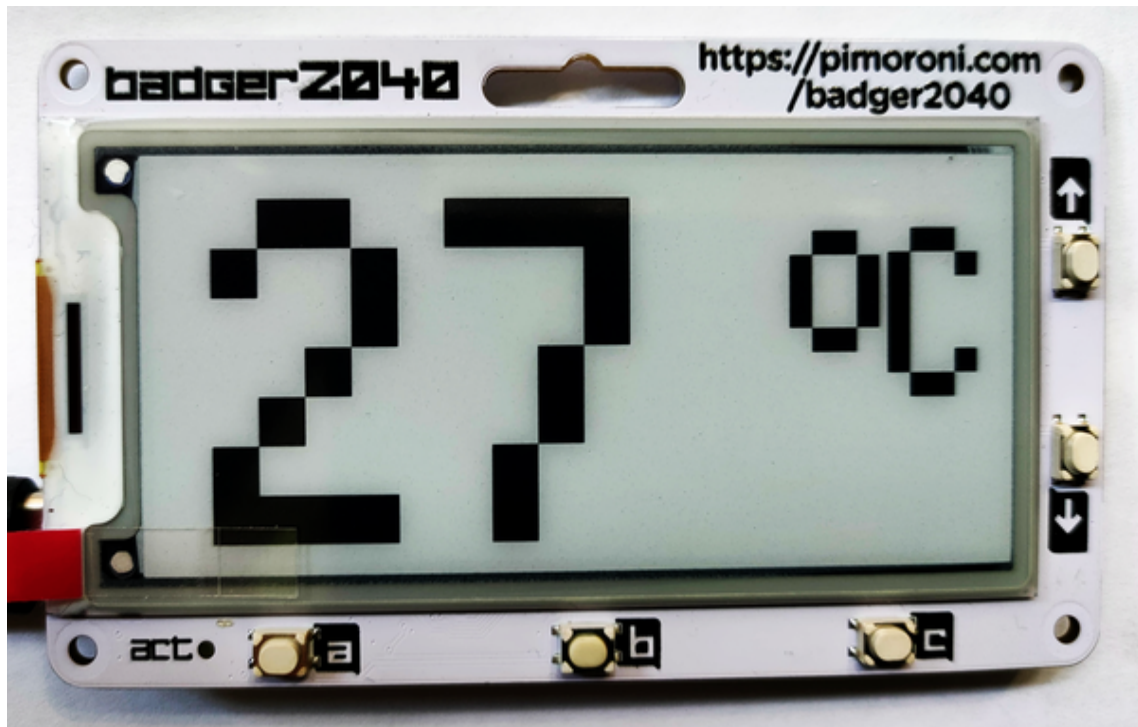


Figure 19-22. The Pimoroni Badger RP2040 board

If you prefer your temperatures in degrees Fahrenheit, use the program `ch_19_badger_f.py`. The temperature is likely to always be shown a few degrees higher than the room temperature, as this is the temperature of the chip, not the environment:

```
import badger2040
from machine import ADC
from utime import sleep

badger = badger2040.Badger2040()
temp_sensor = ADC(4)
points_per_volt = 3.3 / 65535

def read_temp_c():
    reading = temp_sensor.read_u16() * points_per_volt
    temp_c = 27 - (reading - 0.706) / 0.001721
    return temp_c

badger.font("bitmap8")

old_t = 0
```

```

while True:
    t = round(read_temp_c())
    if t != old_t:
        old_t = t
        badger.pen(255)
        badger.clear()
        badger.pen(0)
        badger.text(str(t), 20, 10, scale=16)
        badger.text("°", 220, 5, scale=8)
        badger.text("C", 255, 20, scale=8)
        badger.update()
        sleep(5)

```

As with all the program examples in this book, you can also download this code (see [Recipe 3.22](#)).

The program is based on *ch\_19\_temp\_logger.py*, with the addition of code to display the temperature. The module `badger2040` contains the interface to the E Ink display. Having created an instance of the `Badger2040` class, the default font is set to `bitmap8`, which is quite a blocky font, but looks good as a thermometer.

The main `while` loop reads the temperature, and if it has changed, then it redisplay the new temperature. Unlike other displays, E Ink displays refresh slowly and with a fair amount of blinking. So the display would look quite distracting if it refreshed every time through the `while` loop.

Writing to the display is similar to using an OLED display (see [Recipe 15.4](#)). The things to be displayed are written to a buffer before the display is told to update using the `update` method. In this case, the display is first cleared by setting the ink color to white (255) and then calling `clear()`. The pen color is then set to 0 (black), the temperature is converted to text, and then it is drawn at coordinates `x = 20, y = 10`. Some smaller text is then displayed for the units of degrees. Finally, there is a `sleep` of five seconds to stop the display from constantly refreshing when the measured temperature is at the threshold between two values.

## See Also

For more information on using the Pimoroni Badger, see <https://oreil.ly/EyxoP>.

Check out a reference on the [badger2040 module](#).

## 19.13 Using the Pico on Batteries

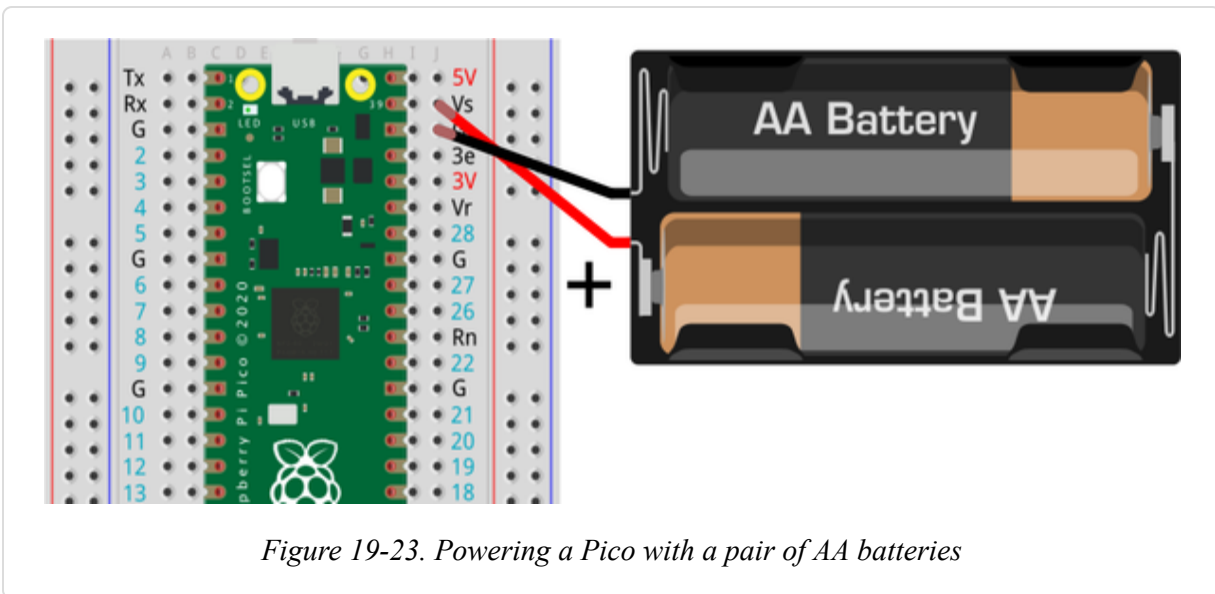
### Problem

You want to run your Pico or Pico W from batteries.

### Solution

A really good balance of long battery life and low cost can be achieved by using a battery pack containing two AA batteries.

This can easily be connected using a Pico on a breadboard as shown in [Figure 19-23](#).



*Figure 19-23. Powering a Pico with a pair of AA batteries*

### Discussion

The Pico is pretty easy to power from batteries by virtue of its buck/boost voltage regulator. This takes an input voltage to pin Vsys (marked Vs on the



MonkMakes Breadboard for Pico) that can be anything from 1.8V to 5.5V and converts that voltage to the 3.3V that the RP2040 needs.

A Raspberry Pi 4 can use several amps of current, making it poorly suited for low-power battery applications. In contrast, the Pico uses only about 25mA. An AA battery can typically hold at least 2,000mAh, which means that a pair of AA batteries could last for 80 hours or more.

Some of the Pico-compatible boards, such as Adafruit's Feather RP2040 and Pimoroni's PicoLiPo, allow a 3.7V rechargeable LiPo battery to be attached. This will charge while the board is connected to USB, and then provide power to the Pico when you unplug it.

## **See Also**

Find out more about the [Feather RP2040](#) and the [PicoLiPo](#).

# Appendix A. Parts and Suppliers

---

## Parts

The following tables will help you find the parts used in this book. Where possible, I have listed product codes for a few suppliers.

Many electronic component manufacturers and suppliers now cater to the maker and electronics hobbyist. Some of the most popular are listed in [Table A-1](#).

*Table A-1. Parts suppliers*

Supplier	Website	Notes
Adafruit	<a href="http://www.adafruit.com">http://www.adafruit.com</a>	Good for modules
Cool Components	<a href="https://coolcomponents.co.uk">https://coolcomponents.co.uk</a>	Wide range of accessories for Raspberry Pi
CPC	<a href="http://cpc.farnell.com">http://cpc.farnell.com</a>	UK-based; wide range of components
Digi-Key	<a href="http://www.digikey.com">http://www.digikey.com</a>	Wide range of components
Farnell	<a href="http://www.farnell.com">http://www.farnell.com</a>	International; wide range of components
MonkMakes	<a href="http://www.monkmakes.com">http://www.monkmakes.com</a>	Electronics kits for Raspberry Pis, etc.
Mouser	<a href="http://www.mouser.com">http://www.mouser.com</a>	Wide range of components
Pimoroni	<a href="https://shop.pimoroni.com">https://shop.pimoroni.com</a>	UK-based retailer and manufacturer of interesting HATs
Pololu	<a href="https://www.pololu.com">https://www.pololu.com</a>	Great for motor controllers and robots
Seeed Studio	<a href="http://www.seeedstudio.com">http://www.seeedstudio.com</a>	Interesting low-cost modules
SparkFun	<a href="http://www.sparkfun.com">http://www.sparkfun.com</a>	Good for modules

The other great source of components is eBay.

Searching for components can be time consuming and difficult. The [Octopart component search engine](#) can be very helpful in tracking down parts. MonkMakes, Adafruit, and SparkFun all have packs of components to get you started.

## Prototyping Equipment and Kits

Many of the hardware projects in this book use jumper wires of various sorts. Male-to-female leads (to connect the Raspberry Pi GPIO connector to a breadboard) and male-to-male (to make connections on the breadboard) are particularly useful. Female-to-female are occasionally useful for connecting modules directly to GPIO pins. You rarely need leads longer than 3 inches (75 mm). [Table A-2](#) lists some jumper wire and breadboard specifications, along with their suppliers.

A handy way to get started with a breadboard, jumper wires, and some basic components is to buy a starter kit, like the [Project Box 1 kit for Raspberry Pi](#) from [MonkMakes](#). This kit was developed, at least partly, with this book in mind.

*Table A-2. Prototyping equipment and kits*

Description	Suppliers
M-M jumper wires	SparkFun: PRT-08431; Adafruit: 759
M-F jumper wires	SparkFun: PRT-09140; Adafruit: 825
F-F jumper wires	SparkFun: PRT-08430; Adafruit: 794
Half-size breadboard	SparkFun: PRT-09567; Adafruit: 64
Pi Cobbler	Adafruit: 1105
Raspberry Leaf (26-pin)	Adafruit: 1772
Raspberry Leaf (40-pin)	Cool Components: 3408
Electronics Starter Kit for Raspberry Pi	Amazon; <i>MonkMakes</i>
Adafruit Perma-Proto for Pi (half breadboard)	Adafruit: 1148

Description	Suppliers
Adafruit Perma-Proto for Pi (full breadboard)	Adafruit: 1135
Adafruit Perma-Proto HAT	Adafruit: 2314
DC barrel jack-to-screw terminal adapter (female)	Adafruit: 368
Pimoroni Breakout Garden HAT	Pimoroni
Basic soldering kit	Adafruit: 136

## Resistors and Capacitors

Table A-3 shows resistors and capacitors used in this cookbook and some suppliers.

*Table A-3. Resistors and capacitors*

Part	Suppliers
270 $\Omega$ 0.25W resistor	Mouser: 293-270-RC
470 $\Omega$ 0.25W resistor	Mouser: 293-470-RC
1k $\Omega$ 0.25W resistor	Mouser: 293-1K-RC
3.3k $\Omega$ 0.25W resistor	Mouser: 293-3.3K-RC
4.7k $\Omega$ 0.25W resistor	Mouser: 293-4.7K-RC
10k $\Omega$ trimpot	Adafruit: 356; SparkFun: COM-09806; Mouser: 652-3362F-1-103LF
Photoresistor	Adafruit: 161; SparkFun: SEN-09088
330 nF capacitor	Mouser: 80-C330C334K5R
Thermistor T0 of 1k Beta 3800 NTC	Mouser: 871-B57164K102J (Note: Beta is 3730)

## Transistors and Diodes

Table A-4 lists transistors and diodes used in this cookbook and some suppliers.

*Table A-4. Transistors and diodes*

<b>Part</b>	<b>Suppliers</b>
FQP30N06L N-Channel logic level MOSFET transistor	Mouser: 512-FQP30N06L; Sparkfun: COM-10213
2N3904 NPN bipolar transistor	SparkFun: COM-00521; Adafruit: 756
1N4001 diode	Mouser: 512-1N4001; SparkFun: COM-08589; Adafruit: 755
TIP120 Darlington transistor	Adafruit: 976; CPC: SC10999
2N7000 MOSFET transistor	Mouser: 512-2N7000; CPC: SC06951

## Integrated Circuits

Table A-5 lists integrated circuits used in this cookbook and some suppliers.

*Table A-5. Integrated circuits*

<b>Part</b>	<b>Suppliers</b>
L293D motor driver	SparkFun: COM-00315; Adafruit: 807; Mouser: 511-L293D; CPC: SC10241
ULN2803 Darlington driver IC	SparkFun: COM-00312; Adafruit: 970; Mouser: 511-ULN2803A; CPC: SC08607
DS18B20 temperature sensor	SparkFun: SEN-00245; Adafruit: 374; Mouser: 700-DS18B20; CPC: SC10426
MCP3008 eight-channel ADC IC	Adafruit: 856; Mouser: 579-MCP3008-I/P; CPC: SC12789
TMP36 temperature sensor	SparkFun: SEN-10988; Adafruit: 165; Mouser: 584-TMP36GT9Z; CPC: SC10437

## OptoElectronics

Table A-6 lists optoelectronic components used in this cookbook and some suppliers.

*Table A-6. Optoelectronics*

<b>Part</b>	<b>Suppliers</b>
5 mm red LED	SparkFun: COM-09590; Adafruit: 299
RGB common cathode LED	SparkFun: COM-11120; eBay
TSOP38238 IR sensor	SparkFun: SEN-10266; Adafruit: 157

## Modules

Table A-7 lists modules used in this cookbook and some suppliers.

*Table A-7. Modules*

<b>Part</b>	<b>Suppliers</b>
Raspberry Pi Camera Module	Adafruit: 3099; Cool Components: 1932
Level converter, four-way	SparkFun: BOB-12009; Adafruit: 757
Level converter, eight-way	Adafruit: 395
LiPo boost converter/charger	SparkFun: PRT-14411
PowerSwitch Tail	Amazon
16-channel servo controller	Adafruit: 815
Motor driver 1A dual	SparkFun: ROB-14451
PIR motion detector	Adafruit: 189
Ultimate GPS	Adafruit: 746
Methane sensor	SparkFun: SEN-09404
Gas sensor breakout board	SparkFun: BOB-08891
ADXL335 triple-axis accelerometer	Adafruit: 163
4x7-segment LED with I2C backpack	Adafruit: 878

<b>Part</b>	<b>Suppliers</b>
Bicolor LED square-pixel matrix with I2C backpack	Adafruit: 902
RTC module	Adafruit: 3296
16x2 HD44780 compatible LCD module	SparkFun: LCD-00255; Adafruit: 181
Sense HAT	Adafruit: 2738
Adafruit Capacitive Touch HAT	Adafruit: 2340
Stepper Motor HAT	Adafruit: 2348
16-channel PWM HAT	Adafruit: 2327
Pimoroni Explorer HAT Pro	<i>Pimoroni</i> ; Adafruit: 2427
Squid Button	<i>MonkMakes</i> ; Amazon
Raspberry Squid RGB LED	<i>MonkMakes</i> ; Amazon
I2C OLED display 128x64 pixels	eBay—search for: I2C OLED Arduino
MMA8452Q triple-axis accelerometer breakout	SparkFun: SEN-12756
MH-Z14A CO <sub>2</sub> sensor module	eBay—search for: MH-Z14A
RC-522 RFID module	eBay—search for: RC-522
Pimoroni VL53L1X distance sensor breakout	Pimoroni or eBay—search for: VL53L1X
Sonoff Basic WiFi Switch	ITEAD
Raspberry Pi Zero Camera Adapter	Adafruit: 3157
Wemos D1 Mini	eBay—search for: Wemos D1 Mini
Pimoroni Audio DAC Shim (line-out)	Pimoroni
Pimoroni Badger 2040	Pimoroni

## Miscellaneous

Table A-8 lists miscellaneous tools and components used in this cookbook and some suppliers.

*Table A-8. Miscellaneous*

<b>Part</b>	<b>Suppliers</b>
1200mAh LiPo battery	Adafruit: 258

<b>Part</b>	<b>Suppliers</b>
5V relay	SparkFun: COM-00100
5V panel meter	SparkFun: TOL-10285
Standard servomotor	SparkFun: ROB-09065; Adafruit: 1449
9g mini servomotor	Adafruit: 169
5V 2A power supply	Adafruit: 276
Low-power 6V DC motor	Adafruit: 711
0.1-inch header pins	SparkFun: PRT-00116; Adafruit: 392
5V, 5-pin unipolar stepper motor	Adafruit: 858
12V, 4-pin bipolar stepper motor	Adafruit: 324
Tactile push switch	SparkFun: COM-00097; Adafruit: 504
Miniature slide switch	SparkFun: COM-09609; Adafruit: 805
Rotary encoder	Adafruit: 377
4×3 keypad	SparkFun: COM-14662
Piezo buzzer	SparkFun: COM-07950; Adafruit: 160
Reed switch	Adafruit: 375
Console lead	Adafruit: 954



# Appendix B. Raspberry Pi Pinouts

---

## Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero

Figure B-1 shows the pinout for the current 40-pin general-purpose input/output (GPIO) Raspberry Pi.

3.3V	<input type="checkbox"/>	<input type="checkbox"/>	5V
2 SDA	<input type="checkbox"/>	<input type="checkbox"/>	5V
3 SCL	<input type="checkbox"/>	<input type="checkbox"/>	GND
4	<input type="checkbox"/>	<input type="checkbox"/>	14 TXD
GND	<input type="checkbox"/>	<input type="checkbox"/>	15 RXD
17	<input type="checkbox"/>	<input type="checkbox"/>	18
27	<input type="checkbox"/>	<input type="checkbox"/>	GND
22	<input type="checkbox"/>	<input type="checkbox"/>	23
3.3V	<input type="checkbox"/>	<input type="checkbox"/>	24
10 MOSI	<input type="checkbox"/>	<input type="checkbox"/>	GND
9 MISO	<input type="checkbox"/>	<input type="checkbox"/>	25
11 SCKL	<input type="checkbox"/>	<input type="checkbox"/>	8
GND	<input type="checkbox"/>	<input type="checkbox"/>	7
ID_SD	<input type="checkbox"/>	<input type="checkbox"/>	ID_SC
5	<input type="checkbox"/>	<input type="checkbox"/>	GND
6	<input type="checkbox"/>	<input type="checkbox"/>	12
13	<input type="checkbox"/>	<input type="checkbox"/>	GND
19	<input type="checkbox"/>	<input type="checkbox"/>	16
26	<input type="checkbox"/>	<input type="checkbox"/>	20
GND	<input type="checkbox"/>	<input type="checkbox"/>	21

Figure B-1. 40-pin Raspberry Pi GPIO pinout

## Raspberry Pi Model B revision 2, A

If you bought one of the early 26-pin Raspberry Pis, then it is most likely to be a model B revision 2 board, as shown in [Figure B-2](#). (If you have one of these first editions of the Raspberry Pi, hang on to it; it might be valuable one day.)

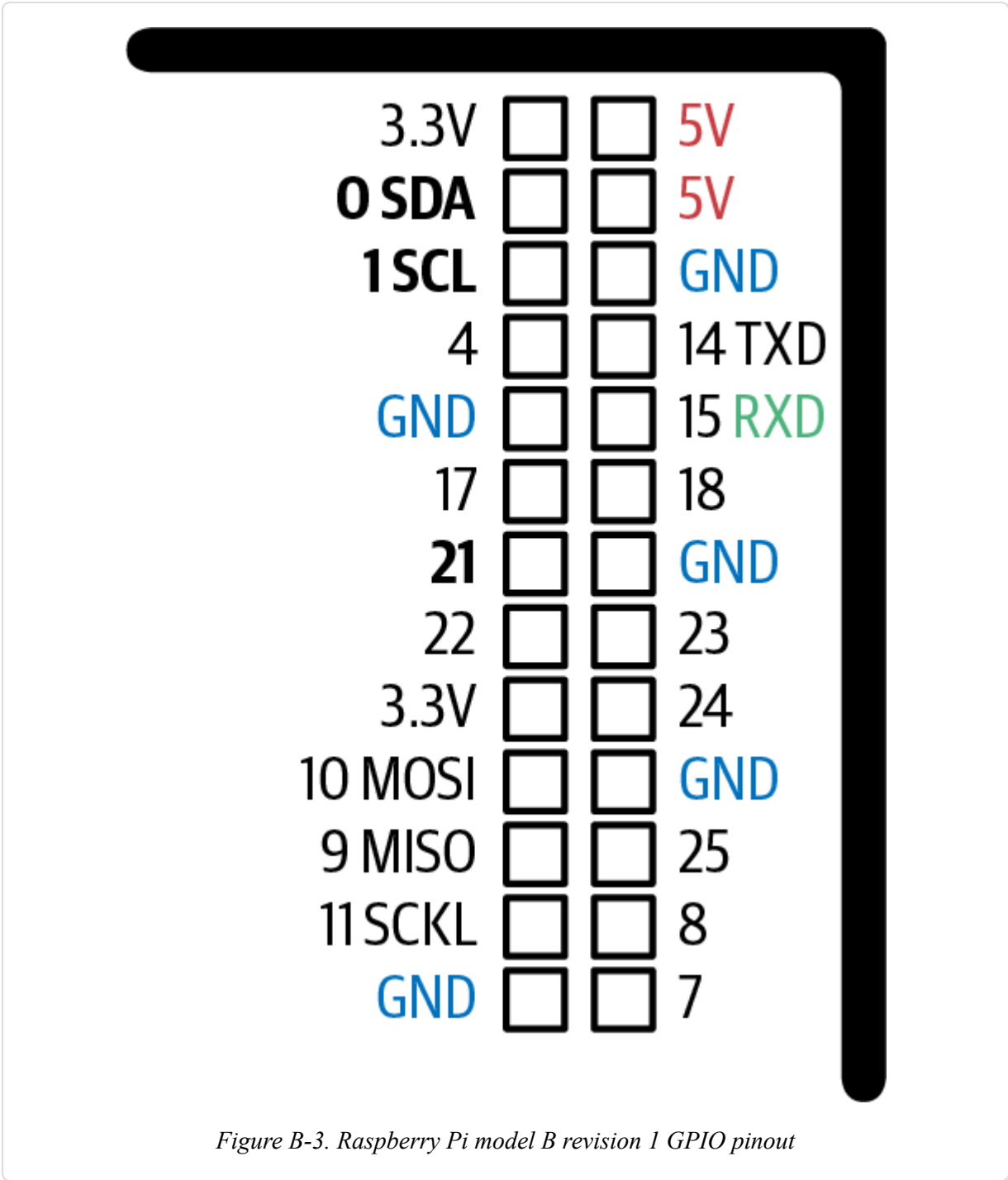
3.3V	<input type="checkbox"/>	<input type="checkbox"/>	5V
2 SDA	<input type="checkbox"/>	<input type="checkbox"/>	5V
3 SCL	<input type="checkbox"/>	<input type="checkbox"/>	GND
4	<input type="checkbox"/>	<input type="checkbox"/>	14 TXD
GND	<input type="checkbox"/>	<input type="checkbox"/>	15 RXD
17	<input type="checkbox"/>	<input type="checkbox"/>	18
27	<input type="checkbox"/>	<input type="checkbox"/>	GND
22	<input type="checkbox"/>	<input type="checkbox"/>	23
3.3V	<input type="checkbox"/>	<input type="checkbox"/>	24
10 MOSI	<input type="checkbox"/>	<input type="checkbox"/>	GND
9 MISO	<input type="checkbox"/>	<input type="checkbox"/>	25
11 SCKL	<input type="checkbox"/>	<input type="checkbox"/>	8
GND	<input type="checkbox"/>	<input type="checkbox"/>	7

*Figure B-2. Raspberry Pi model B revision 2 and model A GPIO pinout*

## Raspberry Pi Model B revision 1

The very first released version of the Raspberry Pi model B (revision 1) has some minor pinout differences to the revision 2 that followed. This is the

only version of the Raspberry Pi that is not compatible with later pinouts. The incompatible pins that changed are highlighted in bold in **Figure B-3**.



## Raspberry Pi Pico

Although the Raspberry Pi Pico also has a 40-pin connector, the connections are entirely different, as shown in [Figure B-4](#).

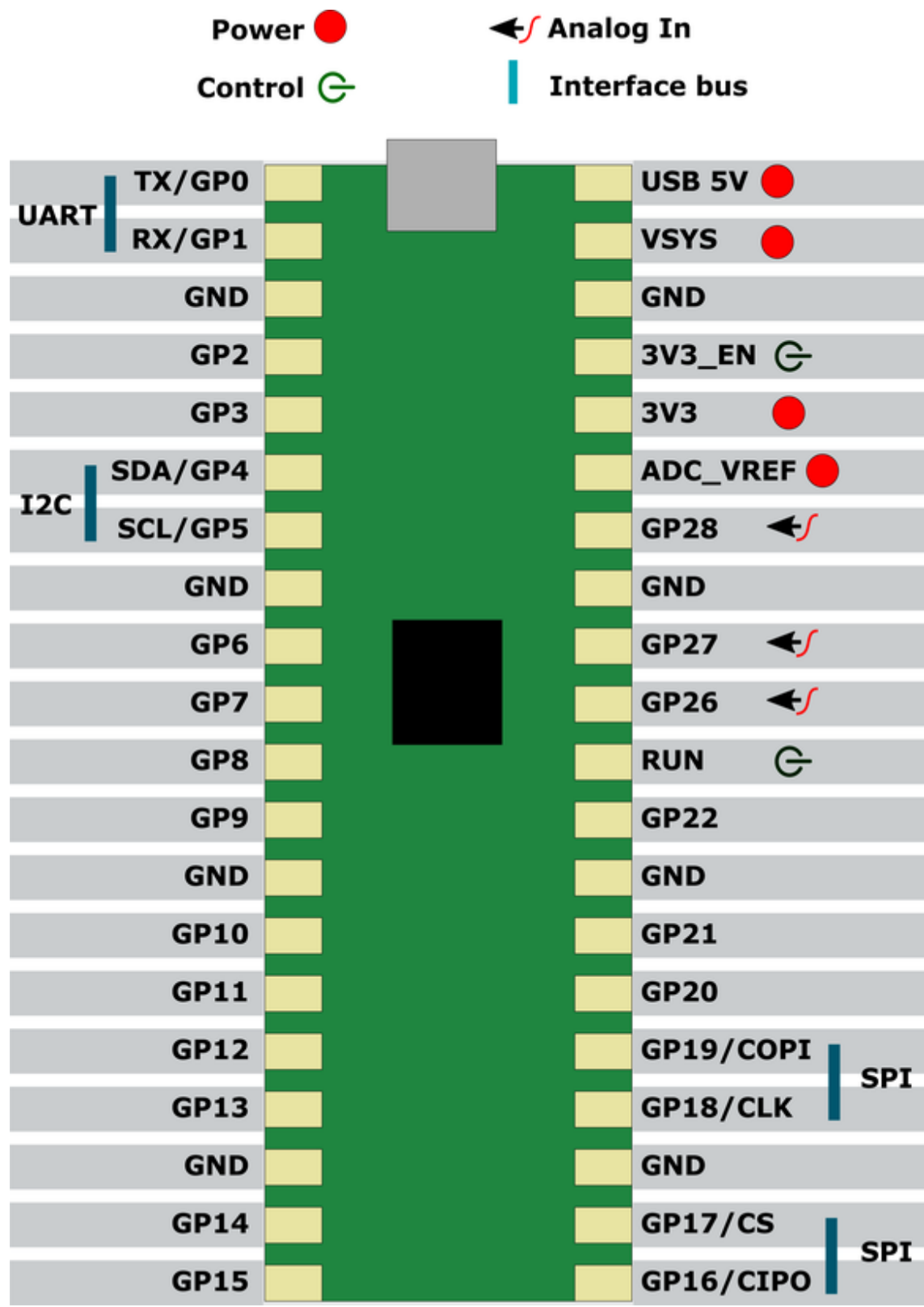


Figure B-4. Raspberry Pi Pico GPIO pinout

# Index

---

## Symbols

! (running command from history), [Discussion](#)

!= (not equal to), [Discussion](#)

# (prompt), [Discussion](#)

#! (shebang), [Discussion](#)

\$ (prompt), [Discussion](#)

\$ (regex end of string), [Solution](#)

& (file run in background), [Problem](#)  
autorun files, [Solution](#)

""" (documentation strings in Python), [Solution](#)  
reading contents of, [Solution](#)

\* (regex zero or more), [Solution](#)

\* (wildcard)  
crontab for executing scripts, [Discussion](#)  
finding files, [Discussion](#)  
listing files, [Discussion](#)  
removing files, [Solution](#)

+ (regex one or more), [Solution](#)

. (period)  
hidden files, [Discussion](#)  
listing directories, [Discussion](#)  
regex single character match, [Solution](#)

.. (two periods)  
listing directories, [Discussion](#)



moving through directories, [Discussion](#)

/ (in paths), [Discussion](#)

< (less than), [Discussion](#)

< (redirecting input), [Discussion](#)

<= (less than or equal), [Discussion](#)

<> (not equal to), [Discussion](#)

= (assignment operator), [Problem](#), [Solution](#)

== (equal to), [Discussion](#)

> (greater than), [Discussion](#)

> (redirecting output), [Problem](#), [Problem](#)  
/dev/null, [Discussion](#), [Problem](#)

>= (greater than or equal), [Discussion](#)

>>> (prompt), [Discussion](#)  
MicroPython for Pico, [Solution](#)

[:] (slice)  
lists, [Problem](#)  
strings, [Solution](#)

[] (lists), [Solution](#), [Solution](#), [Discussion](#)  
dictionary entry access, [Solution](#)

[] (regex characters matched), [Solution](#)

\ (escape characters), [Discussion](#)

\d (regex digit), [Solution](#)

\s (regex whitespace), [Solution](#)

\w (regex alphanumeric), [Solution](#)

^ (regex beginning of string), [Solution](#)

\_\_ (double underscores)  
class definition constructor method, [Solution](#)  
reading documentation string, [Solution](#)

- version of computer vision, [Solution](#)
- { } (dictionaries), [Solution](#)
- { } (formatting string), [Discussion](#), [Solution](#)
- | (pipe), [Problem](#)
  - history piped to grep, [Discussion](#)
  - ps piped to grep, [Discussion](#)
- ~ (home directory), [Discussion](#)

## A

- AC devices with PowerSwitch Tail II, [Problem](#)
- acceleration measurement
  - Inertial Measurement Unit and, [Problem-See Also](#)
  - MMA8452Q module for, [Problem-See Also](#)
- account creation on first boot, [Solution](#)
  - changing password, [Problem](#)
- Adafruit
  - Air Quality board, [Discussion](#)
  - bonnets for Pi 400, [Discussion](#)
  - Capacitive Touch HAT, [Solution-See Also](#)
  - CircuitPython boards, [Discussion](#)
  - CircuitPython libraries, [Discussion](#)
  - Feather RP2040 Pico-compatible, [Discussion](#)
  - I2C devices, [Discussion](#)
  - NeoPixels guide online, [See Also](#)
  - NeoPixels RGB LED matrix, [Problem-See Also](#)
  - NeoPixels RGB LED strip, [Problem-See Also](#)
  - servo control, [See Also](#)
  - servomotor HAT, [Solution-Discussion](#)

servomotor HAT alternative, [See Also](#)  
Stepper Motor HAT, [Problem](#)  
ADC chip (see analog-to-digital converter (ADC) chip)  
Add/Remove Software tool, [Discussion](#)  
addressable RGB LED matrix, [Problem-See Also](#)  
addressable RGB LED strips, [Problem-See Also](#)  
Air Quality board (Adafruit), [Discussion](#)  
aliases, [Problem](#)  
amperage maximum for Raspberry Pi, [Solution](#)  
analog inputs lacking, [See Also](#), [Introduction](#), [Solution](#)  
(see also analog-to-digital converter (ADC) chip; sensors)  
analog inputs on Pico boards, [Introduction](#), [Discussion](#)  
analog temperature sensor, [Discussion](#), [Problem-See Also](#)  
Pimoroni Badger temperature sensor, [Discussion](#)  
voltage measurement, [Problem-Discussion](#)  
analog outputs on Pico boards, [Problem-See Also](#)  
analog voltmeters, [Problem-See Also](#), [Problem-Discussion](#)  
analog-to-digital converter (ADC) chip  
analog inputs lacking, [See Also](#), [Introduction](#), [Solution](#)  
resistive sensors with, [Problem-See Also](#)  
temperature measurement with, [Problem-See Also](#)  
voltage measurement, [Problem-See Also](#), [Problem-Discussion](#)  
and logical operator, [Problem](#)  
Android for controlling hardware, [Problem-See Also](#)  
AngularServo class (gpiozero), [Solution](#)  
aplay command, [Discussion](#), [Solution](#)  
documentation online, [See Also](#)  
append function (Python), [Solution](#)

- apt for installing software, [Problem](#)
  - removing installed software, [Problem](#)
  - searching for software to install, [Discussion](#)
  - updating, [Discussion](#)
  - updating operating system, [Solution](#)

## Arduino

- converting 5V signals to 3.3V with two resistors, [Problem](#)
- integrated development environment, [Solution](#)
- Wemos D1 programming, [Solution-Discussion](#)

- arecord command, [Solution](#)

- argv variable for command-line arguments, [Solution](#)

- arithmetic in Python, [Problem](#)

- ASCII value of string characters, [Discussion](#)

- assignment (=) operator (Python), [Problem](#), [Solution](#)

- Atari 2600 emulator, [Solution-See Also](#)

- atmospheric pressure measurement, [Problem-See Also](#)

- audio (see sound)

- autocomplete via Tab key, [Discussion](#)

- automation (see home automation)

- autorunning programs or scripts

- as a service, [Problem-See Also](#)

- command-line arguments for, [Discussion](#)

- Node-RED visual programming tool, [Discussion](#)

- on startup, [Problem](#)

- regular intervals, [Problem](#)

## B

- background processes, [Solution](#), [Problem](#)

LED blinking, [Discussion](#)

Badger Pico-compatible board (Pimoroni), [Discussion-See Also](#)  
information online, [See Also](#)

badger2040 library for E Ink display (MicroPython), [Discussion](#)  
reference online, [See Also](#)

batteries

AA powering Pico boards, [Problem](#)

LiPo battery powering Pico-compatible boards, [Discussion](#)

LiPo battery powering Raspberry Pi, [Problem](#)

BBC micro:bit boards, [Discussion](#)

binary–base 10 conversions

number formatting, [Discussion](#)

string-to-number conversion, [Discussion](#)

bipolar stepper motors, [Problem-See Also](#)

Stepper Motor HAT to drive, [Problem](#)

bitmap image editor, [Problem-See Also](#)

blinking an LED, [Discussion](#)

Blue Dot (Android app), [Solution-See Also](#)

Blue Dot library for remote (Python), [See Also](#)

bluedot library for Android and Bluetooth control, [Solution](#)

Bluetooth

controlling hardware with, [Problem-See Also](#)

making Raspberry Pi discoverable, [Solution](#)

Raspberry Pi Pico W, [Introduction](#)

setup, [Problem-See Also](#)

bonnets (add-on boards; Adafruit), [Discussion](#)

Bookshelf software, [Problem](#)

boost regulator module, [Solution](#)

booting up

after shutdown, [Solution](#)

date and time set, [Discussion](#)

first time, [Problem](#)

hard disk or USB flash drive, [Problem-See Also](#)

Raspberry Pi 400, [Solution](#)

rebooting, [Problem](#)

running program as service, [Problem-See Also](#)

running program on startup, [Problem](#)

bottle library (Python), [Solution](#), [Solution](#), [Solution-Discussion](#)

documentation online, [See Also](#), [See Also](#)

breadboard

jumper leads to GPIO, [Problem-See Also](#)

Pico breadboard kits, [Problem-See Also](#)

power supply leads, [Solution](#)

sources for, [Prototyping Equipment and Kits](#)

break statement (Python), [Solution](#)

browser (see web interface)

Button class (gpiozero), [Solution](#), [Solution](#)

documentation online, [See Also](#)

button click running function, [Solution](#)

Blue Dot app for Android, [Problem-See Also](#)

push switch connection, [Problem-See Also](#)

user interface for, [Problem](#)

buzzer, [Problem-See Also](#)

## C

camera

Raspberry Pi Camera Module, [Problem-See Also](#)

Raspberry Pi Camera Module for computer vision, [Problem](#)  
(see also computer vision)

Raspberry Pi Camera Module for machine learning, [Introduction](#)

USB webcam for computer vision, [Problem-See Also](#)  
(see also computer vision)

USB webcam for machine learning, [Introduction](#)

capacitive touch sensing, [Problem-See Also](#)

capacitors, [Discussion](#)

sources for, [Resistors and Capacitors](#)

carbon dioxide concentration measured, [Problem-See Also](#)

case sensitivity

Python, [Discussion](#)

string functions, [Discussion](#)

Terminal, [Discussion](#)

cases (enclosures), [Problem-See Also](#)

cat command

concatenating files, [Solution](#)

version of operating system displayed, [Solution](#)

view file contents, [Problem](#)

cd command, [Solution](#)

Celsius–Fahrenheit conversion (Python), [Discussion](#)

f-string for, [Discussion](#)

more than one return value, [Solution](#)

number formatting, [Discussion](#)

center-off toggle switch, [Problem-See Also](#)

central processing unit (see CPU)

check\_output function (Python), [Discussion](#)

CheerLights, [Problem-See Also](#)

chmod command, [Solution](#)

file executable for owner, [Discussion](#)

octal calculator, [See Also](#)

choice command (Python), [Discussion](#)

chown command, [Solution](#)

classes (Python)

about classes, [Solution](#)

defining, [Problem-See Also](#)

documentation strings, [Solution](#)

inheritance, [Problem](#)

methods defined, [Problem](#)

multiple inheritance, [See Also](#)

closing a file, [Solution](#), [Solution](#), [Discussion](#), [Discussion](#)

coins counted via computer vision, [Problem-See Also](#)

Color class (colorzero), [Solution](#)

colorzero library for Color class, [Solution](#)

comma-separated value (CSV) file written to USB flash drive, [Problem](#)

command history

Python console command history, [Discussion](#)

Terminal command-line history, [Problem](#)

command line, [Problem-See Also](#)

(see also Terminal)

common cathode LED, [Solution](#)

Common Unix Printing System (CUPS), [Problem-See Also](#)

comparing values (Python), [Problem-See Also](#)

compass for magnetic north, [Problem](#)

composite video connection, [Problem-See Also](#)



- comprehensions to build new lists, [Problem](#)
- compressed files, [Problem](#)
- computer vision (CV), [Introduction-See Also](#)
  - counting coins, [Problem-See Also](#)
  - face detection, [Problem-See Also](#)
  - installing OpenCV, [Problem](#)
  - motion detection, [Problem-See Also](#)
  - OCR, [Problem](#)
  - Raspberry Pi Camera Module for, [Problem](#)
  - USB camera for, [Problem-See Also](#)
- concatenation
  - files, [Problem](#)
  - strings, [Problem](#)
- conditional expressions (Python), [Problem](#)
- console cable connection, [Problem-See Also](#)
  - connecting, [Solution](#)
  - Pi Zero WiFi connection, [Discussion](#)
  - purchasing, [Solution](#), [See Also](#)
- console in Python, [Problem-See Also](#)
- copy command for lists (Python), [Discussion](#)
- copy library (Python), [Discussion](#)
- copying files or folders, [Problem-See Also](#)
- counting coins via computer vision, [Problem-See Also](#)
- cp command, [Solution](#)
  - parameter information online, [See Also](#)
- CPU (central processing unit)
  - displaying temperature readings on web page, [Problem-See Also](#)
  - monitoring usage of, [Problem-See Also](#)

sleep function reducing load, [Discussion](#)

temperature measured, [Problem](#)

temperature notification emailed via IFTTT, [Solution-See Also](#)

temperature sensor of Pico board, [Discussion](#)

writing temperature readings to USB flash drive, [Solution](#)

crontab command, [Solution](#)

CSV file format written to USB flash drive, [Problem](#)

Ctrl-C to stop execution, [Discussion](#), [Solution](#)

bottle web server, [Solution](#)

computer vision coin counting, [Discussion](#)

does not always work, [Solution](#)

sound playback stopped, [Solution](#)

spoken command recognition, [Discussion](#)

CUPS (Common Unix Printing System), [Problem-See Also](#)

CV (see computer vision)

Cyberdeck Bonnet (Adafruit), [Discussion](#)

## D

Darlington transistor, [Discussion](#)

dashboard for Node-RED

devices on and off, [Problem-See Also](#)

scheduling events, [Problem-See Also](#)

data entry validated via regex, [Problem](#)

data structure pickled, [Problem](#)

date

formatting output of, [Problem](#)

setting, [Problem](#)

datetime command (Python), [Solution](#)

datetime library (Python), [Solution](#)

DC motor

speed and direction controller, [Problem-See Also](#)

speed controller, [Problem](#)

debouncing a button press, [Problem-See Also](#)

debugging in Python, [Discussion](#)

/dev devices

/dev/null, [Problem](#)

serial port, [Discussion](#), [Solution](#)

SPI, [Discussion](#)

df command, [Solution](#)

DHCP (Dynamic Host Configuration Protocol), [Discussion](#), [Solution-Discussion](#)

lease time, [Solution](#)

reservation, [Solution](#)

dictionaries in Python, [Problem-See Also](#)

creating, [Problem](#)

finding and changing entries, [Problem](#)

internal order essentially random, [Discussion](#), [Solution](#)

iterating over, [Problem](#)

removing entries from, [Problem](#)

saved as JSON files, [Problem](#)

digital inputs, [Introduction-See Also](#)

debouncing a button press, [Problem-See Also](#)

external pull-up resistors, [Problem](#)

GPS, [Problem-See Also](#)

keypad, [Problem-See Also](#)

keypresses intercepted, [Problem](#)

mouse movements intercepted, [Problem](#)  
movement detector, [Problem](#)  
Pico boards, [Problem-See Also](#)  
push switch connection, [Problem-See Also](#)  
push switch toggling, [Problem-See Also](#)  
reset button, [Problem-See Also](#)  
rotary (quadrature) encoder, [Problem-See Also](#)  
three-position (center-off) toggle or slide switch, [Problem-See Also](#)  
two-position toggle or slide switch, [Problem](#)

digital outputs on Pico boards, [Problem-See Also](#)  
digital temperature sensor, [Problem-See Also](#)  
diodes, sources for, [Transistors and Diodes](#)

directories (folders)  
  copying, [Problem-See Also](#)  
  creating, [Problem](#)  
  current directory via pwd, [Solution](#)  
  deleting, [Problem](#)  
  ownership, [Problem](#)  
  permissions, [Problem-See Also](#)  
  permissions changed, [Problem](#)  
  renaming, [Problem](#)

displays, [Introduction-See Also](#)  
  addressable RGB LED matrix, [Problem-See Also](#)  
  addressable RGB LED strips, [Problem-See Also](#)  
  analog voltmeter, [Problem-See Also](#), [Problem-Discussion](#)  
  E Ink on Pimoroni Badger, [Discussion-See Also](#)  
  ePaper display, [Problem-See Also](#)  
  four-digit LED, [Problem](#)

- I2C LED matrix display, [Problem-See Also](#)
- monitor attached to board (see monitor)
- OLED graphical display, [Problem-See Also](#)
- Pimoroni Unicorn HAT, [Problem-See Also](#)
- Sense HAT LED matrix display, [Problem-See Also](#)
- sensor values displayed, [Problem](#)
- dist-upgrade command, [Discussion](#)
- distance measurement
  - time-of-flight sensor for, [Problem-See Also](#)
  - ultrasound for, [Problem-See Also](#)
- DistanceSensor class (gpiozero), [Solution](#), [Discussion](#)
  - documentation online, [See Also](#)
- documentation strings (Python), [Solution](#)
  - reading contents of, [Solution](#)
- double buffering, [Discussion](#)
- DS18B20 digital temperature sensor, [Solution-Discussion](#)
  - datasheet online, [See Also](#)
- dump function (json), [Solution](#)
- DVI monitor connected, [Problem](#)
- dweepy library for Dweet (Python), [Solution](#)
- Dweet to respond to tweets, [Problem-See Also](#)
  - about dweet.io web service, [Solution](#)
- Dynamic Host Configuration Protocol (see DHCP)
- dynamic overclocking, [Solution](#)

## E

- E Ink display on Pimoroni Badger, [Discussion-See Also](#)
  - badger2040 library, [Discussion](#), [See Also](#)

echo command, [Solution](#)

Edge Impulse platform for machine learning  
about, [Introduction](#)

AudioImpulseRunner, [Solution](#)

Dashboard, [Discussion](#)

information online, [See Also](#), [See Also](#)

installing and using, [Problem](#)

Linux information online, [See Also](#)

Linux runner for local performance, [Problem-See Also](#)

spoken command recognized, [Problem-See Also](#)

editing a file, [Problem-See Also](#)

editors

Mu, [Problem-See Also](#)

Python, [Problem](#)

Thonny, [Problem](#)

Visual Studio Code, [Solution-See Also](#)

EEPROM chip on HAT board, [Solution-Discussion](#)

reading data from, [Discussion](#)

writing data to, [Discussion](#)

email

address validation via regex, [Solution](#)

IFTTT for sending, [Problem-See Also](#)

motion detector sending, [See Also](#)

sending from Python, [Problem-See Also](#)

embedding

Raspberry Pi Zero, [See Also](#)

Raspberry Pi Zero W, [Solution](#), [See Also](#)

enclosures (cases), [Problem-See Also](#)

enumerate command (Python), [Solution](#)  
enumerating a list, [Problem](#)  
ePaper display, [Problem-See Also](#)  
error handling (Python), [Discussion](#), [Problem](#)  
escape characters (Python), [Discussion](#)  
ESP8266-based WiFi board publishing MQTT messages, [Problem-See Also](#)  
esptool for flashing firmware, [Solution](#)  
/etc/hostname file for network name, [Setting the network name using the command line \(the hard way\)](#)  
/etc/hosts file for network name, [Setting the network name using the command line \(the hard way\)](#)  
/etc/init.d/ folder for scripts as services, [Solution](#)  
/etc/os-release file, [Solution](#)  
/etc/rc.local file for running files on startup, [Solution](#)  
exception handling (Python), [Discussion](#), [Problem](#)  
Explorer HAT Pro (Pimoroni), [Problem-See Also](#), [Discussion](#)  
    ADC chip, [See Also](#)  
extend function (Python), [Discussion](#)  
extracting ZIP file, [Solution](#), [Discussion](#)

## F

f-strings for evaluating code in strings, [Discussion](#)  
face detection via computer vision, [Problem-See Also](#)  
Facebook notifications, [Problem-See Also](#)  
Fahrenheit–Celsius conversion (Python), [Discussion](#)  
    f-string for, [Discussion](#)  
    more than one return value, [Solution](#)  
    number formatting, [Discussion](#)

fg command, [Solution](#)

file archives, uncompressing, [Problem](#)

File Manager utility for GUI browsing, [Solution](#)

copying files onto a USB flash drive, [Problem-See Also](#)

files

browsing in GUI, [Problem-See Also](#)

concatenating, [Problem](#)

copying, [Problem-See Also](#)

copying onto a USB flash drive, [Problem-See Also](#)

creating without using editor, [Problem](#)

deleting, [Problem](#)

editing, [Problem-See Also](#)

errors caught by try/except, [Discussion](#), [Problem](#)

fetching code files for this book, [Problem](#)

fetching from command line, [Problem](#)

finding, [Problem](#)

hidden via period prefix, [Discussion](#)

JSON files from dictionaries, [Problem](#)

JSON files read and parsed, [Problem-See Also](#)

navigating with Terminal, [Problem-See Also](#)

ownership changed, [Problem](#)

permissions, [Problem-See Also](#)

permissions changed, [Problem](#)

pickling data into, [Problem](#)

Pico and Pico W boards, [Problem-See Also](#)

Pico files transferred to Raspberry Pi, [Solution](#)

Python file extension .py, [Solution](#), [Discussion](#)

reading from, [Problem](#), [Problem](#)



- reading Pico files, [Solution](#), [Discussion](#)
- redirecting command line output to a file, [Problem](#)
- renaming, [Problem](#)
- uncompressing, [Solution](#), [Problem](#)
- viewing contents of, [Problem](#), [Solution](#), [Discussion](#)
- writing to, [Problem](#)
- writing to on Pico board, [Problem-See Also](#)
- ZIP file extracted, [Solution](#), [Discussion](#)
- filesystem navigation, [Problem-See Also](#)
  - finding a file, [Problem](#)
  - Pico and Pico W boards, [Problem-See Also](#)
- find command, [Solution](#)
- find function (Python), [Solution](#)
- firmware (Tasmota) flashed, [Problem-See Also](#)
  - Tasmota information online, [See Also](#)
- firmware installed by download for Pico, [Solution-Solution](#)
  - filesystem contents lost, [Discussion](#)
- flash drive (see USB flash drive)
- flashing Sonoff WiFi Smart Switch, [Problem-See Also](#)
- float function (Python), [Solution](#)
- FM radio transmitter, [Problem-See Also](#)
- folders (directories)
  - copying, [Problem-See Also](#)
  - creating, [Problem](#)
  - current directory via pwd, [Solution](#)
  - deleting, [Problem](#)
  - ownership, [Problem](#)
  - permissions, [Problem-See Also](#)

- permissions changed, [Problem](#)
- renaming, [Problem](#)
- for command (Python), [Solution](#)
  - enumerating a list, [Solution](#), [Solution](#)
  - iterating over a list, [Solution](#)
  - repeating a command, [Solution](#)
- format method (Python), [Solution](#), [Solution](#)
- formatting
  - dates and times, [Problem](#)
  - numbers, [Problem-See Also](#)
- four-digit LED display, [Problem](#)
- FQP30N06L MOSFET, [Solution-See Also](#), [Solution](#)
- functions (Python)
  - button click running, [Solution](#)
  - command-line arguments, [Problem](#)
  - creating, [Problem](#)
  - methods in classes, [Problem](#)
  - more than one running at a time, [Problem-See Also](#), [Problem](#)
  - parameters, [Discussion](#)
  - return value, [Discussion](#)
  - returning more than one value, [Problem](#)

## G

- game console emulator, [Problem-See Also](#)
- gas detection/measurement
  - carbon dioxide concentration, [Problem-See Also](#)
  - methane detection, [Problem-See Also](#)
- general-purpose input/output connector (see GPIO)

GIMP (GNU image manipulation program), [Problem-See Also](#)

Git

fetching source code, [Problem-See Also](#)

git clone command, [Solution](#)

GitHub versus, [Discussion](#)

information online, [See Also](#)

version control with, [Discussion](#)

your GitHub repository, [Discussion](#)

GitHub

about code in book, [Preface to the Fourth Edition](#)

creating a repository on, [Discussion](#)

fetching code files for this book, [Problem](#)

Git versus, [Discussion](#)

information online, [See Also](#)

Node-RED flows from book, [Discussion](#)

this book's page, [Solution](#)

GitLab, [See Also](#)

GNU image manipulation program (GIMP), [Problem-See Also](#)

Google Gmail example, [Solution](#)

GPIO (general-purpose input/output) connector

3.3V for inputs and outputs, [Solution](#), [Solution](#)

amperage maximum, [Solution](#), [Solution](#), [Discussion](#)

analog inputs on Pico boards, [Introduction](#), [Discussion](#)

analog inputs via ADC chip, [See Also](#)

basics, [Introduction-See Also](#)

breadboard with jumper leads, [Problem-See Also](#)

controlling hardware through, [Introduction-See Also](#)

converting 5V signals to 3.3V with level converter module, [Problem](#)

converting 5V signals to 3.3V with two resistors, [Problem](#)  
digital inputs (see digital inputs)  
digital outputs on Pico boards, [Problem-See Also](#)  
external electronics safely connected to Raspberry Pi, [Problem](#)  
high-power LEDs needing transistor, [Problem-See Also](#)  
LEDs connected to Pico boards, [Problem-See Also](#)  
LEDs connected to Raspberry Pi, [Problem-See Also](#)  
MQTT messages controlling, [Problem-See Also](#)  
output capabilities, [See Also](#)  
output control via web interface, [Problem-See Also](#)  
pinouts, [Problem-See Also](#), [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero-Raspberry Pi Pico](#)  
pinouts template, [Discussion](#)  
pins left in safe state, [Problem](#)  
pins “floating”, [Discussion](#)  
power supply, [Discussion](#)  
(see also power supply)  
pull-up resistors, [Solution](#), [Discussion](#)  
Raspberry Pi 400, [Problem-See Also](#)  
solid-state relay for switching, [Problem](#)

gpiozero library (Python)  
about, [Solution](#)  
Button class documentation online, [See Also](#)  
buzzer sound, [Solution](#)  
CheerLights RGB color, [Solution](#)  
CPU temperature measured, [Solution](#)  
debouncing a button press, [Solution](#)  
distance measurement with ultrasound, [Solution](#)

GPIO controlled via web interface, [Solution](#)  
GPIO pins left in safe state, [Solution](#)  
LED notification of tweets, [Solution](#)  
LED on and off, [Solution](#), [Solution](#)  
machine library versus, [Discussion](#)  
motion sensor, [Solution](#)  
on/off switch via user interface, [Solution](#)  
push switch, [Solution-See Also](#)  
PWM feature, [Solution-See Also](#)  
PWM via user interface, [Solution](#)  
Raspberry Squid Button, [Solution](#)  
Raspberry Squid RGB LED, [Solution](#)  
rotary (quadrature) encoder, [Solution](#)  
servo positioning, [Solution](#), [Problem](#)  
speed and direction of DC motor, [Solution](#)  
stepper motor control, [Discussion](#)  
temperature measurement with ADC, [Solution](#)  
temperature notifications via IFTTT, [Solution](#)  
temperature notifications via ThingSpeak, [Solution](#), [Solution](#)  
three-position (center-off) switch, [Solution](#)  
voltage measurement with ADC, [Solution](#), [Solution](#)

GPS, [Problem-See Also](#)  
    meaning of fields online, [See Also](#)  
    tutorial on tracking using Python, [See Also](#)

graphical user interface (GUI)  
    browsing files, [Problem-See Also](#)  
    creating user interfaces, [Problem-See Also](#)  
        (see also user interface)

VNC for remote access to Pi, [Problem-See Also](#)

grep

history piped to, [Discussion](#)

ps piped to, [Discussion](#)

grounding against static, [Solution](#)

GUI (see graphical user interface)

guizero library (Python)

displaying values, [Solution](#)

documentation online, [See Also](#)

GUI design with, [Solution-See Also](#)

soil moisture measurement, [Solution](#)

user interface for on/off switch, [Solution](#)

user interface for PWM, [Solution](#)

gunzip command, [Solution](#)

.gz compressed files, [Solution](#)

gzip command, [Discussion](#)

## H

H-bridge chip, [Solution-Discussion](#), [Solution](#)

HAAR (High Altitude Aerial Reconnaissance), [Solution](#)

hard disk, booting from, [Problem-See Also](#)

hardware

analog voltmeters, [Problem-See Also](#), [Problem-Discussion](#)

basics, [Introduction-See Also](#)

breadboard with jumper leads, [Problem-See Also](#)

breadboard with Pico board, [Problem-See Also](#)

controlling, [Introduction-See Also](#)

controlling with Android and Bluetooth, [Problem-See Also](#)

converting 5V signals to 3.3V with level converter module, [Problem](#)  
converting 5V signals to 3.3V with two resistors, [Problem](#)  
digital inputs (see digital inputs)  
digital outputs of Pico boards, [Problem-See Also](#)  
Explorer HAT Pro basics, [Problem-See Also](#)  
external electronics safely connected to Raspberry Pi, [Problem](#)  
GPIO pinouts, [Problem-See Also](#), [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero-Raspberry Pi Pico](#)  
(see also GPIO)  
GPIO pins left in safe state, [Problem](#)  
I2C setup, [Problem-See Also](#)  
i2c-tools, [Problem-See Also](#)  
LED brightness control, [Problem-See Also](#), [Problem-See Also](#)  
LED connections, [Problem-See Also](#), [Problem-See Also](#)  
LiPo battery powering Pico-compatible boards, [Discussion](#)  
LiPo battery powering Raspberry Pi, [Problem](#)  
Minicom installation, [Problem](#)  
Perma-Proto Pi HAT, [Problem-See Also](#)  
Pico boards (see [Raspberry Pi Pico](#); [Raspberry Pi Pico W](#))  
pinouts for GPIO, [Problem-See Also](#), [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero-Raspberry Pi Pico](#)  
pySerial installation, [Problem](#)  
Raspberry Pi Zero 2/Pi Zero 2 W, [Problem](#)  
Raspberry Squid Button, [Problem](#), [Solution](#)  
Raspberry Squid RGB LED, [Problem-See Also](#), [Solution](#), [Problem-See Also](#)  
Sense HAT basics, [Problem](#)  
(see also [Sense HAT](#) )

servomotors (see servomotors)

SPI setup, [Problem-See Also](#)

switching a high-power DC device using a transistor, [Problem-See Also](#)

switching a high-power device using a relay, [Problem-See Also](#)

switching high-voltage AC devices, [Problem](#)

switching using a solid-state relay, [Problem](#)

user interface for RGB LED color, [Problem-See Also](#)

user interface to control PWM, [Problem-See Also](#)

user interface to turn things on and off, [Problem](#)

Hardware Attached on Top (HAT) standard (see HAT (Hardware Attached on Top) standard)

hash table, [Discussion](#)

hashtags responded to via Dweet and IFTTT, [Problem-See Also](#)

HAT (Hardware Attached on Top) standard, [Discussion](#), [Solution](#)  
design guide online, [See Also](#)

Raspberry Pi 400 adapter, [Discussion](#)

ready-made HATs on the market, [See Also](#)

HDMI

HDMI to DVI adapters, [Solution](#)

monitor connected, [Problem-See Also](#)

ServoBlaster and no audio, [Solution](#), [Solution](#)

sound from Raspberry Pi, [Discussion](#)

headless Raspberry Pi installation, [Problem-See Also](#)

console cables for headless Pi, [Solution](#)

remote connection for, [Introduction](#)

(see also networking)

sound output options, [Discussion](#)

headphones connected, [Discussion](#)



help

manpages for OS, [See Also](#)

Python console help command, [Discussion](#)

hexadecimal notation, [Solution](#)

hexadecimal–base 10 conversions

number formatting, [Discussion](#)

string-to-number conversion, [Discussion](#)

High Altitude Aerial Reconnaissance (HAAR), [Solution](#)

high-voltage AC devices

connecting Sonoff to, [Solution](#)

switching with PowerSwitch Tail II, [Problem](#)

history command, [Solution](#)

! running command, [Discussion](#)

piped to grep, [Discussion](#)

home automation, [Introduction-See Also](#)

about, [Introduction](#)

flashing a Sonoff WiFi Smart Switch for MQTT, [Problem-See Also](#)

Mosquitto message broker, [Problem-See Also](#)

Node-RED dashboard for, [Problem-See Also](#)

Node-RED scheduled events, [Problem-See Also](#)

Node-RED with an MQTT server, [Problem-See Also](#)

Sonoff switches with Node-RED, [Problem-See Also](#)

Sonoff web switches with MQTT, [Problem-See Also](#)

Sonoff WiFi Smart Switch configuration, [Problem-See Also](#)

Wemos D1 publishing MQTT messages, [Problem-See Also](#)

Wemos D1 with Node-RED, [Problem-See Also](#)

home directory, [Solution](#), [Discussion](#)

~ shorthand for, [Discussion](#)

hostname command, [Solution](#)

hostname of Raspberry Pi

- \$ prompt, [Discussion](#)
- changing, [Problem-See Also](#)
- default raspberrypi, [Problem](#)
- more than one Pi on network, [Solution](#), [Discussion](#)
- network-attached storage name, [See Also](#)

HTML parsed with regex, [Problem](#)

htop command, [Discussion](#)

HTTP requests, [Problem](#)

- CPU too hot, [Solution](#)
- GPIO controlled via web interface, [Problem-See Also](#)
- IFTTT trigger, [Solution](#)
- sensor data sent to ThingSpeak, [Solution-See Also](#)
- tweet via ThingSpeak, [Solution](#)
- tweets responded to via Dweet and IFTTT, [Problem-See Also](#)
- web server in Python, [Problem](#)

humidity measurement, [Problem-See Also](#)

## I

I2C (Inter-Integrated Circuit)

- HAT EEPROM, [Discussion](#)
- i2c-tools, [Problem-See Also](#)
- LED matrix display, [Problem-See Also](#)
- LED module, [Solution](#)
- setup, [Problem-See Also](#)

i2c-tools, [Problem-See Also](#)

- HAT EEPROM, [Discussion](#)

- if command (Python), [Problem](#)
  - elif conditions, [Discussion](#)
  - else statements, [Discussion](#)
  - logical operators, [Problem](#)
- ifconfig command, [Discussion](#)
- IFTTT (If This Then That)
  - account sign-up, [Solution](#)
  - responding to tweets with, [Problem-See Also](#)
  - sending email and other notifications, [Problem-See Also](#)
- Imager (see Raspberry Pi Imager)
- import command (Python), [Solution](#)
- in keyword for lists (Python), [Solution](#)
- indentation in Python code, [Discussion](#)
- Inertial Measurement Unit (IMU), [Problem-See Also](#)
- infrared (IR) remote, [See Also](#)
- inheritance of classes, [Problem](#)
  - multiple inheritance, [See Also](#)
- Inkscape vector image editor, [Problem](#)
- Inky pHAT or wHAT (Pimoroni), [Solution-See Also](#)
  - documentation online, [See Also](#)
- input command (Python 3), [Solution](#)
  - example of use, [Solution](#)
  - Python 2 raw\_input command, [Discussion](#)
  - validation via regex, [Problem](#)
- inputs, digital (see digital inputs)
- insert command (Python), [Discussion](#)
- int function (Python), [Solution](#)
- integrated circuits, sources for, [Integrated Circuits](#)

Inter-Integrated Circuit (see I2C)

interface board (Perma-Proto Pi HAT), [Problem-See Also](#)

internet connection

first boot, [Solution](#)

networking, [Introduction](#)

(see also networking)

Internet of Things (IoT), [Introduction-See Also](#)

CheerLights, [Problem-See Also](#)

displaying sensor readings on web page, [Problem-See Also](#)

GPIO control via web interface, [Problem-See Also](#)

home automation (see home automation)

IFTTT for email and other notifications, [Problem-See Also](#)

Node-RED visual programming tool, [Problem-See Also](#)

(see also Node-RED)

Raspberry Pi Zero 2 W for, [Solution](#)

responding to tweets using Dweet and IFTTT, [Problem-See Also](#)

sensor data sent to ThingSpeak, [Problem-See Also](#)

ThingSpeak for sending tweets, [Problem-See Also](#)

internet radio, [Problem-See Also](#)

interrupts driving code execution, [Solution](#)

IP address

DHCP allocating, [Discussion](#), [Solution-Discussion](#)

finding, [Discussion](#), [Problem](#)

finding for Sonoff Smart Switch, [Solution](#)

information online, [See Also](#)

IPv4 address, [Solution](#)

setting static address, [Problem-See Also](#)

IR (infrared) remote, [See Also](#)

iterating

over a dictionary, [Problem](#)

over a list, [Problem](#)

over a list via comprehension, [Solution](#)

## J

JavaScript function in HTML, [Discussion](#), [Solution](#), [Discussion](#)

join function (Python), [Discussion](#)

JSON files

dictionary saved as, [Problem](#)

JSON data parsed, [Problem-See Also](#)

pickling versus, [Discussion](#)

json library (Python)

dictionary saved as JSON file, [Solution](#)

parsing JSON data, [Solution](#)

jumper wires, sources for, [Prototyping Equipment and Kits](#)

justgage library for sensor values (Python), [Discussion](#)

## K

K40 Whisperer software, [Solution](#)

keyboard connected, [Problem-See Also](#)

headless Raspberry Pi instead, [Problem-See Also](#)

keypad for input, [Problem-See Also](#)

keypresses intercepted, [Problem](#)

kill command, [Discussion](#)

killall command, [Discussion](#)

Kitronik Discover Kit, [Solution](#)

Kodi media center software, [Solution-See Also](#)

## L

L293D H-bridge chip, [Solution-Discussion](#), [Solution](#)  
datasheet online, [See Also](#)

laser cutter controller, [Problem-See Also](#)

lease time for DHCP, [Solution](#)

### LEDs

addressable RGB LED matrix, [Problem-See Also](#)

addressable RGB LED strips, [Problem-See Also](#)

amperage maximum, [Solution](#)

blinking an LED, [Discussion](#), [Problem-See Also](#)

CheerLights, [Problem-See Also](#)

connecting to Raspberry Pi, [Problem-See Also](#)

connecting to Raspberry Pi Pico, [Problem-See Also](#)

controlling brightness of, [Problem-See Also](#), [Problem-See Also](#)

four-digit display, [Problem](#)

matrix display, [Problem-See Also](#)

matrix display of RGB, [Problem-See Also](#)

opto-isolator using, [Discussion](#)

power supply, [Solution](#), [Discussion](#)

Raspberry Squid RGB, [Problem-See Also](#), [Solution](#), [Problem-See Also](#)

resistor needed for connecting, [Discussion](#), [Discussion](#)

resistor selection, [Discussion](#), [Discussion](#)

switching high-power LEDs using a transistor, [Problem-See Also](#)

toggling with a push switch, [Solution-See Also](#)

user interface for RGB LED color, [Problem-See Also](#)

user interface to control PWM, [Problem-See Also](#)

len function (Python), [Solution](#), [Solution](#)

less command, [See Also](#)

level converter module, [Problem](#)

libraries (MicroPython)

badger2040 for E Ink display, [Discussion](#)

machine for Pico hardware support, [Discussion](#)

microdot for web server, [Solution](#), [See Also](#)

mm\_wlan for WiFi connection, [Solution](#), [See Also](#)

\_thread for threads, [Solution](#)

utime for sleep function, [Solution](#)

libraries (Python), [Problem](#)

Blue Dot for remote, [See Also](#)

bluedot for Android and Bluetooth control, [Solution](#)

bottle for HTTP requests, [Solution](#), [Solution](#)

colorzero for Color class, [Solution](#)

copy for copy command, [Discussion](#)

datetime, [Solution](#)

dweepy for Dweet, [Solution](#)

gpiozero for GPIO support, [Solution](#)

(see also gpiozero library)

guizero for GUI, [See Also](#)

(see also guizero library)

I2C, [Solution](#)

json, [Solution](#), [Solution](#)

justgage for sensor values, [Discussion](#)

NumPy, [Solution](#), [Solution](#), [Solution](#), [Solution](#)

OS for system command, [Solution](#)

PiAnalog for using analog sensors, [Solution](#), [Solution](#), [Solution](#)

PyAudio, [Solution](#)

pygame for keypress detection, [Solution](#)

pygame for mouse movement detection, [Solution](#)

pyserial, [Solution](#)

Python Image Library, [Solution](#)

Python Standard Library, [See Also](#)

random, [Solution](#)

re for regular expressions, [Solution](#), [Solution](#)

RFID card reader/writer, [Solution-See Also](#)

smtplib for email, [Solution](#)

subprocess for Linux commands, [Discussion](#), [Solution](#), [Solution](#)

sys for command-line arguments, [Solution](#), [Solution](#), [Solution](#)

threading for threads, [Solution](#)

time for sleep function, [Solution](#)

urllib for HTTP requests, [Solution](#)

using libraries in programs, [Problem](#)

LibreELEC media center software, [Solution](#)

LibreOffice software, [Problem-See Also](#)

light box for computer vision, [Solution](#)

light, measuring, [Problem-See Also](#)

lightweight processes, [Discussion](#)

Linux

network-attached storage connection, [Discussion](#)

operating system, [Introduction](#)

(see also operating system)

running commands from Python, [Problem](#)

SSH remote connection to Pi, [Solution](#)

VNC remote connection to Pi, [Problem-See Also](#)

LiPo (lithium-ion polymer) battery, [Problem](#)



Pico-compatible boards, [Discussion](#)

lists in Python, [Introduction-See Also](#)

- accessing elements, [Problem](#)
- adding elements to, [Problem](#)
- command-line arguments, [Solution](#)
- comprehensions to build new lists, [Problem](#)
- creating, [Problem](#)
- creating by parsing a string, [Problem](#)
- creating sublists, [Problem](#)
- dump function to save as JSON, [Discussion](#)
- enumerating, [Problem](#), [Solution](#)
- iterating over a list, [Problem](#)
- length of, [Problem](#)
- random selection of element, [Discussion](#)
- removing elements from, [Problem](#)
- sorting, [Problem](#)
- testing for certain element in list, [Problem](#)

logging

- sensor data sent to ThingSpeak, [Problem-See Also](#)
- sensor data to file on Pico, [Problem-See Also](#)
- to USB flash drive, [Problem](#)

logging out, [Problem](#)

logical operators, [Problem](#)

looping to repeat instructions (Python)

- exiting loop, [Problem](#)
- for so many times, [Problem](#)
- loop variable, [Discussion](#)
- while some condition, [Problem](#)

loudspeaker connected, [Problem-See Also](#)

lower function (Python), [Solution](#)

ls command, [Discussion-Discussion](#)

file permissions, [Solution](#)

output piped to grep, [Solution](#)

SPI check, [Discussion](#)

lsusb command, [Solution](#)

## M

machine learning (ML), [Introduction-See Also](#)

about, [Introduction](#)

classification, [Introduction](#)

Edge Impulse platform, [Introduction](#), [Problem](#)

identifying objects in video, [Problem-See Also](#)

reacting to a whistle, [Problem-See Also](#)

reacting to objects in video, [Problem-See Also](#)

sound identification, [Problem-See Also](#)

spoken command recognized in Python, [Problem-See Also](#)

spoken command recognized locally, [Problem-See Also](#)

spoken command recognized via cloud, [Problem-See Also](#)

machine library for Pico hardware support, [Discussion](#)

gpiozero library versus, [Discussion](#)

macOS

console cable connection, [Solution](#)

network-attached storage connection, [Discussion](#)

SSH remote connection to Pi, [Solution](#)

VNC remote connection to Pi, [Problem-See Also](#)

magazines on Raspberry Pi, [Problem](#)

magnetic north detected, [Problem](#)

magnets

sensing with reed switch, [Problem](#)

sensing with Sense HAT, [Problem](#), [Problem](#)

MagPi magazine, [Problem](#)

manpages (manual pages), [See Also](#)

math in Python, [Problem](#)

MCP3008 ADC chip

analog voltage measurement, [Solution-Discussion](#)

datasheet online, [See Also](#)

resistive sensors with, [Problem-See Also](#)

voltages reduced for measurement, [Problem-See Also](#)

media center software, [Problem-See Also](#)

/media/pi folder for USB flash drive, [Solution](#), [Solution](#), [Discussion](#)

member variables (Python), [Discussion](#)

Message Queuing Telemetry Transport (see MQTT)

metal-oxide-semiconductor field-effect transistor (see MOSFET)

methane detection, [Problem-See Also](#)

sensor datasheet online, [See Also](#)

methods in Python classes, [Problem](#)

microdot web server library (MicroPython), [Solution](#)

information online, [See Also](#)

microphone, [Problem-See Also](#)

MicroPython for Raspberry Pi Pico, [Introduction](#)

about, [Discussion](#)

digital outputs, [Solution-See Also](#)

documentation online, [See Also](#)

installing with Thonny editor, [Solution](#)

Pimoroni Badger Pico-compatible board, [Discussion](#)

sleep function in utime module, [Solution](#)

threads, [Problem](#)

microSD card

booting for the first time, [Problem](#)

booting from hard disk or USB flash drive, [Problem-See Also](#)

installing operating system onto, [Problem-See Also](#)

networking, [Discussion](#)

selecting, [Discussion](#)

shutting down properly, [Discussion](#)

test utility, [Discussion](#)

trying different operating systems, [Discussion](#)

Minicom

GPS raw data, [Solution](#)

serial port test, [Problem](#)

mkdir command, [Solution](#)

ML (see machine learning)

MMA8452Q accelerometer module, [Problem-See Also](#)

datasheet online, [See Also](#)

mm\_wlan WiFi connection library (MicroPython), [Solution](#)

information online, [See Also](#)

models of Raspberry Pi, [Problem-See Also](#)

differences between models, [Discussion](#)

recommended models, [Preface to the Fourth Edition](#)

selection based on usage, [Solution](#), [Discussion](#)

modules (Python)

level converter module, [Problem](#)

picamera module, [See Also](#)

Raspberry Pi Camera Module, [Problem-See Also](#), [Problem](#)  
sources for, [Modules](#)  
using modules in programs, [Problem](#)  
(see also libraries (Python))

## monitor

adjusting picture size, [Problem-See Also](#)  
connecting DVI or VGA, [Problem](#)  
connecting HDMI, [Problem-See Also](#)  
connecting TV or composite video, [Problem-See Also](#)  
displaying output from board (see displays)  
headless Raspberry Pi instead, [Problem-See Also](#)  
modifying resolution, [Problem-See Also](#)  
sound from Raspberry Pi, [Discussion](#)  
VNC virtual monitor, [Solution](#)

## MonkMakes

breadboard kit for Pico, [Solution](#)  
GPIO Adapter for Pi 400, [Solution](#)  
Plant Monitor board, [Problem-See Also](#), [Discussion-See Also](#)  
Project Box 1 kit for Raspberry Pi, [Solution](#), [Solution](#), [Prototyping Equipment and Kits](#)  
Servo Six board, [Discussion](#), [Discussion](#)  
Servo Six board documentation online, [See Also](#)  
solid-state relay, [Solution](#)  
Speaker Kit for Raspberry Pi, [Solution](#), [See Also](#)  
Squid RGB LED, [Solution](#)

Mopidy music player daemon, [See Also](#)

more command, [Discussion](#)

MOSFET (metal-oxide-semiconductor field-effect transistor), [Solution-See Also](#)

datasheet online, [See Also](#)

logic-level MOSFET, [Discussion](#)

resistor with, [Discussion](#)

Mosquitto, [Problem-See Also](#)

(see also MQTT)

information online, [See Also](#)

motion detection

computer vision for, [Problem-See Also](#)

passive infrared module, [Problem](#)

Motor class (gpiozero), [Solution](#), [Discussion](#)

motors, [Introduction-See Also](#)

about, [Introduction](#)

bipolar stepper motors, [Problem-See Also](#)

motor controller modules, [Discussion](#)

multiple servomotors precisely controlled, [Problem-See Also](#)

servomotor position control, [Problem-See Also](#), [Problem-See Also](#)  
(see also servomotors)

single servomotor precisely controlled, [Problem-See Also](#)

speed and direction of DC motor controlled, [Problem-See Also](#)

speed of DC motor controlled, [Problem](#)

stepper motor control with Stepper Motor HAT, [Problem](#)

stepper motors explained, [Solution](#), [Discussion](#)

unipolar stepper motors, [Problem-See Also](#)

user interface to control PWM, [Problem-See Also](#)

mouse connected, [Problem-See Also](#)

headless Raspberry Pi instead, [Problem-See Also](#)

mouse movements intercepted, [Problem](#)  
MPD (Music Player Daemon), [See Also](#)  
MQTT (Message Queuing Telemetry Transport)  
  basics, [Discussion-Discussion](#)  
  flashing a Sonoff WiFi Smart Switch for use with, [Problem-See Also](#)  
  Mosquitto message broker, [Problem-See Also](#)  
  Node-RED with an MQTT server, [Problem-See Also](#)  
  QoS level information online, [See Also](#)  
  Sonoff web switches with, [Problem-See Also](#)  
  Wemos D1 publishing MQTT messages, [Problem-See Also](#)  
Mu editor, [Problem-See Also](#)  
  REPL button for console, [Solution](#)  
Music Player Daemon (MPD), [See Also](#)  
mv command, [Solution](#)

## N

nano editor, [Solution-See Also](#)  
  alternatives to, [See Also](#)  
NeoPixels  
  Adafruit guide online, [See Also](#)  
  RGB LED matrix, [Problem-See Also](#)  
  RGB LED strip, [Problem-See Also](#)  
network name for Raspberry Pi, [Problem-See Also](#)  
  \$ prompt, [Discussion](#)  
  network-attached storage name, [See Also](#)  
network-attached storage (NAS), [Problem-See Also](#)  
networking, [Introduction-See Also](#)  
  all network connections shown, [Discussion](#)

connecting to a wired network, [Problem-See Also](#)  
connecting via WiFi wireless, [Problem](#)  
connecting with a console cable, [Problem-See Also](#)  
controlling Pi remotely with VNC, [Problem-See Also](#)  
DHCP, [Discussion](#)  
finding IP address, [Problem](#)  
headless Raspberry Pi, [Problem-See Also](#)  
network name for network-attached storage, [See Also](#)  
network name for Raspberry Pi, [Problem-See Also](#)  
network-attached storage, [Problem-See Also](#)  
printing, [Problem-See Also](#)  
setting a static IP address, [Problem-See Also](#)  
setup on first boot, [Solution](#)  
SSH for remote access, [Problem-See Also](#)

neural network machine learning, [Introduction](#)  
spoken language recognition, [Solution-See Also](#)  
(see also machine learning)

## Node-RED

about visual programming, [Problem-See Also](#)  
autorunning on reboot, [Discussion](#)  
basics, [Solution](#)  
dashboard for home automation, [Problem-See Also](#)  
documentation online, [See Also](#), [See Also](#), [See Also](#), [See Also](#), [See Also](#),  
[See Also](#)  
flows from book on GitHub, [Discussion](#)  
flows imported and exported as JSON, [Discussion](#)  
inject node for scheduled events, [Problem-See Also](#)  
MQTT server with, [Problem-See Also](#)



Sonoff switches with, [Problem-See Also](#)

videos that introduce, [See Also](#)

Wemos D1 with, [Problem-See Also](#)

not logical operator, [Problem](#)

notifications sent by IFTTT, [Problem-See Also](#)

numbers

comparing values, [Problem](#)

converting strings to, [Problem](#)

converting to different base, [Discussion](#), [Discussion](#)

converting to strings, [Problem](#)

formatting, [Problem-See Also](#)

random number generation, [Problem](#)

NumPy library, [Solution](#)

Edge Impulse installation, [Solution](#)

motion detector, [Solution](#)

OpenCV installation, [Solution](#)

updating, [Solution](#), [Solution](#)

## O

objects identified via machine learning, [Problem-See Also](#)

objects reacted to via machine learning, [Problem-See Also](#)

OCR (optical character recognition), [Problem](#)

octal calculator, [See Also](#)

Octopart component search engine, [Parts](#)

OLED graphical display, [Problem-See Also](#)

one-wire interface for device data, [Solution](#)

enabling, [Solution](#)

open source code in book, [Preface to the Fourth Edition](#)

OpenCV, [Problem](#)

(see also computer vision)

opening a file

file modes, [Discussion](#)

for reading, [Solution](#), [Discussion](#)

for reading a pickled file, [Discussion](#)

for writing, [Solution](#), [Discussion](#)

for writing a pickled file, [Discussion](#)

operating system, [Introduction-See Also](#)

browsing files in GUI, [Problem-See Also](#)

command aliases, [Problem](#)

command-line history, [Problem](#)

concatenating files, [Problem](#)

copying file or folder, [Problem-See Also](#)

copying files onto a USB flash drive, [Problem-See Also](#)

creating a folder, [Problem](#)

creating file without using editor, [Problem](#)

deleting file or directory, [Problem](#)

devices available for recording from, [Solution](#)

editing a file, [Problem-See Also](#)

fetching code files for this book, [Problem-See Also](#)

fetching files from command line, [Problem](#)

fetching source code with git, [Problem-See Also](#)

file ownership, [Problem](#)

file permissions, [Problem-See Also](#)

filesystem navigation with Terminal, [Problem-See Also](#)

finding files, [Problem](#)

hiding output to the Terminal, [Problem](#)

installation guide online, [See Also](#)

installing onto microSD card, [Problem-See Also](#)

MicroPython for Raspberry Pi Pico, [Introduction](#)  
(see also [MicroPython for Raspberry Pi Pico](#))

microSD card test utility, [Discussion](#)

monitoring processor activity, [Problem-See Also](#)

pipe command, [Problem](#)

Python packages installed with pip, [Problem](#)

redirecting command line output to a file, [Problem](#)

renaming file or folder, [Problem](#)

running a program or script automatically as a service, [Problem-See Also](#)

running a program or script automatically at regular intervals, [Problem](#)

running a program or script automatically on startup, [Problem](#)

running programs in the background, [Problem](#)

screen captures, [Problem](#)

SD card free space determination, [Problem](#)

selecting, [Problem](#)

setting date and time, [Problem](#)

software installed with apt, [Problem](#)

software removed with apt, [Problem](#)

superuser privileges, [Problem](#)

Terminal session started, [Problem-See Also](#)

uncompressing files, [Solution](#), [Problem](#)

updating, [Problem](#)

USB devices listed, [Problem](#)

version displayed, [Problem](#)

viewing contents of file, [Problem](#)

WiFi setup, [Setting up WiFi from the desktop](#)

optical character recognition (OCR), [Problem](#)  
opto-isolator, [Discussion](#)  
optoelectronics, sources for, [OptoElectronics](#)  
or logical operator, [Problem](#)  
organic LED (OLED) graphical display, [Problem-See Also](#)  
orientation sensing, [Problem-See Also](#)  
OS library for system command, [Solution](#)  
    documentation online, [See Also](#)  
OSMC media center software, [See Also](#)  
overclocking, [Problem-See Also](#)  
ownership of files or folders, [Problem](#)

## P

parameters for functions, [Discussion](#)  
    command-line arguments, [Problem](#)  
parts, sources for, [Parts-Miscellaneous](#)  
    Octopart component search engine, [Parts](#)  
passive infrared (PIR) motion detector, [Solution](#)  
passwd command, [Discussion](#)  
password  
    caution about including in code, [Solution](#)  
    changing, [Problem](#)  
    default, [Solution](#)  
    strong with username pi, [Solution](#)  
period (.)  
    hidden files, [Discussion](#)  
    listing directories, [Discussion](#)  
    moving through directories, [Discussion](#)

regex single character match, [Solution](#)

Perma-Proto Pi HAT, [Problem-See Also](#)

permissions for files or folders, [Problem-See Also](#)

changing permissions, [Problem](#)

execute permissions, [Discussion](#)

file executable for owner, [Discussion](#)

photoresistors, [Problem-See Also](#)

about, [Discussion](#)

resistance converted to voltage, [Problem-See Also](#)

phototransistors, [Solution](#)

about, [Discussion](#)

Pi-View, [Discussion](#)

PiAnalog library for using analog sensors, [Solution](#), [Solution](#), [Solution](#)

Pibow Coupé enclosure, [Solution](#)

PiCade arcade machine kit, [Discussion](#)

picamera module, [See Also](#)

pickling data into files, [Problem](#)

JSON file format versus, [Discussion](#)

Pico-compatible boards, [Solution-See Also](#)

(see also Raspberry Pi Pico; Raspberry Pi Pico W)

Adafruit Feather RP2040, [Discussion](#)

LiPo battery attachments for, [Discussion](#)

Pimoroni Badger, [Discussion-See Also](#)

Pimoroni PicoLiPo, [Discussion](#)

piezo-electric buzzer, [Problem-See Also](#)

PIL (Python Image Library), [Solution](#)

Pimoroni

Audio DAC Shim, [Solution](#)

Badger Pico-compatible board, [Discussion-See Also](#)

Breakout Garden for Pi 400, [Discussion](#)

Breakout Garden with time-of-flight sensor, [Solution](#)

Explorer HAT Pro, [Problem-See Also](#), [See Also](#), [Discussion](#)

Inky pHAT or wHAT, [Solution-See Also](#)

PicoLiPo Pico-compatible, [Discussion](#)

Unicorn HAT, [Problem-See Also](#)

pinouts for GPIO, [Problem-See Also](#), [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero-Raspberry Pi Pico](#)

template for, [Discussion](#)

pip Python package manager, [Problem](#)

pipe command (`|`), [Problem](#)

history piped to grep, [Discussion](#)

ps piped to grep, [Discussion](#)

PIR (passive infrared) motion detector, [Solution](#)

pitch (orientation), [Solution](#)

plant soil moisture measured, [Problem-See Also](#), [Discussion-See Also](#)

playing back recorded sound, [Solution](#)

Pololu motor controller boards, [Discussion](#)

pop function (Python)

dictionaries, [Solution](#)

lists, [Solution](#)

potential divider, [Discussion](#)

(see also voltage divider)

power supply

AA four-cell battery holder, [Solution](#)

batteries for Pico boards, [Problem](#)

breadboard holding leads, [Solution](#)

console cable connection, [Solution](#)  
converting 5V to 3.3V with level converter module, [Problem](#)  
converting 5V to 3.3V with two resistors, [Problem](#)  
current draw maximum, [Solution](#)  
current used by Pi, [Discussion](#)  
devices 5V, [Solution](#)  
GPIO 3V, [Solution](#), [Solution](#), [Problem](#)  
GPIO-connected devices, [Discussion](#)  
LEDs, [Solution](#), [Discussion](#)  
LiPo battery powering Picos, [Discussion](#)  
LiPo battery powering Raspberry Pi, [Problem](#)  
low-power mode of shutdown, [Discussion](#)  
Pico boards, [Introduction](#)  
selecting, [Problem-See Also](#)  
servomotors, [Discussion](#)  
WiFi power usage, [Discussion](#)

power up after shutdown, [Solution](#)  
reset button for, [Problem-See Also](#)  
Start button for Raspberry Pi, [See Also](#)

PowerSwitch Tail II for AC-devices, [Problem](#)  
pressure of atmosphere measured, [Problem-See Also](#)  
print command (MicroPython), [Discussion](#), [Discussion](#)  
print command (Python), [Solution](#)  
formatting dates and times, [Solution](#)  
formatting numbers, [Problem-See Also](#)

printing over the network, [Problem-See Also](#)  
privileges of superuser, [Problem](#)  
processor usage monitoring, [Problem-See Also](#)

prompts

# prompt, [Discussion](#)

\$ prompt, [Discussion](#)

>>> prompt, [Discussion](#), [Solution](#)

prototyping board

about which to buy, [Preface to the Fourth Edition](#)

breadboard with jumper leads, [Problem-See Also](#)

Perma-Proto Pi HAT, [Problem-See Also](#)

sources for, [Prototyping Equipment and Kits](#)

prototyping equipment and kits, [Preface to the Fourth Edition](#), [Prototyping Equipment and Kits](#)

ps command, [Discussion](#)

pseudorandom number sequence, [Discussion](#)

publish and subscribe model of MQTT, [Discussion](#)

pull-up resistors, [Problem](#)

Pico boards, [Discussion](#)

pulse-width modulation (PWM)

about, [Discussion](#)

analog outputs on Pico boards, [Introduction](#), [Problem-See Also](#)

analog voltmeter display, [Problem-See Also](#)

information online, [See Also](#)

LED brightness control, [Solution-See Also](#), [Problem-See Also](#)

relays damaged by, [Solution](#)

servomotor position controlled with, [Problem-See Also](#), [Problem-See Also](#)

user interface for RGB LED color, [Problem-See Also](#)

user interface to control power for LEDs and motors, [Problem-See Also](#)

push switches



connecting, [Problem-See Also](#)

debouncing a button press, [Problem-See Also](#)

keypad for input, [Problem-See Also](#)

reset button, [Problem-See Also](#)

Squid Button, [Problem](#)

switch bounce, [Discussion](#)

toggling with, [Problem-See Also](#)

PuTTY terminal software for Windows, [Solution](#)

pwd command, [Solution](#)

PWM (see pulse-width modulation)

PyAudio library (Python), [Solution](#)

pygame library (Python)

documentation for mouse online, [See Also](#)

keypress detection with, [Solution](#)

mouse motion detection with, [Solution](#)

sounds played with, [Discussion](#)

pyserial library, [Solution](#)

pySerial to use serial port, [Problem](#)

Python

about code in book, [Preface to the Fourth Edition](#)

advanced concepts, [Introduction-See Also](#)

arithmetic, [Problem](#)

basics, [Introduction-See Also](#)

case sensitivity, [Discussion](#)

class inheritance, [Problem](#)

command-line arguments for programs, [Problem](#)

comparing values, [Problem-See Also](#)

conditional expressions, [Problem](#)

console, [Problem-See Also](#)

converting numbers to strings, [Problem](#)

converting strings to numbers, [Problem](#)

defining a class, [Problem-See Also](#)

defining a function, [Problem](#)

defining a method, [Problem](#)

dictionaries, [Problem-See Also](#)

displaying output, [Problem](#)

doing more than one thing at a time, [Problem-See Also](#)

editing programs with Mu, [Problem-See Also](#)

editing with Thonny, [Problem](#)

editor for, [Problem](#)

exception handling, [Discussion](#), [Problem](#)

f-strings for evaluating code in strings, [Discussion](#)

file extension .py, [Solution](#), [Discussion](#)

formatting dates and times, [Problem](#)

formatting language link, [See Also](#)

formatting numbers, [Problem-See Also](#)

function returning more than one value, [Problem](#)

help command, [Discussion](#)

indentation, [Discussion](#)

installing Python 2, [Solution](#)

JSON data parsed, [Problem-See Also](#)

libraries, [Problem](#)  
(see also [libraries \(Python\)](#))

lists, [Introduction-See Also](#)

logical constants True and False, [Discussion](#)

logical operators, [Problem](#)

looping broken out of, [Problem](#)  
looping for so many times, [Problem](#)  
looping while some condition, [Problem](#)  
MicroPython explained, [Discussion](#)  
modules, [Problem](#)  
    (see also libraries (Python))  
pickling data into files, [Problem](#)  
pip package manager, [Problem](#)  
playing sound file from, [Problem](#)  
Python Shell for Pico, [Problem-See Also](#)  
random number generation, [Problem](#)  
reading from a file, [Problem](#)  
reading user input, [Problem](#)  
Reference Manual online, [See Also](#)  
regular expressions for data entry validation, [Problem](#)  
regular expressions for pattern searches, [Problem-See Also](#)  
regular expressions for web scraping, [Problem](#)  
running Linux commands from, [Problem](#)  
running programs in Terminal, [Problem](#)  
running programs with shebang, [Discussion](#)  
SDK installation, [Solution](#)  
searches using regular expressions, [Problem-See Also](#)  
sending email from, [Problem-See Also](#)  
serial port access via pySerial, [Problem](#)  
sleep function, [Problem](#)  
spoken language recognized, [Problem-See Also](#)  
string concatenation, [Problem](#)  
string extraction from another string, [Problem](#)

string holding Python code, [Discussion](#)  
string length, [Problem](#)  
string position within another, [Problem](#)  
string replacement within another string, [Problem](#)  
string variable created, [Problem](#)  
strings converted to uppercase or lowercase, [Problem](#)  
threading, [Problem-See Also](#)  
user interfaces for, [Problem-See Also](#)  
variables, [Problem](#)  
version 2 versus version 3, [Problem](#)  
web requests, [Problem](#)  
web server, [Problem](#)  
writing to a file, [Problem](#)

Python Image Library (PIL), [Solution](#)  
python-smbus I2C library, [Solution](#)

## Q

quadrature encoder, [Problem-See Also](#)

## R

radio transmitter, [Problem-See Also](#)  
radio, internet, [Problem-See Also](#)  
radio-frequency identification (RFID) reader/writer, [Problem-See Also](#)  
random library (Python), [Solution](#)  
random number generation (Python), [Problem](#)  
    pseudorandom actually, [Discussion](#)  
    random package information online, [See Also](#)  
Raspberry Pi (generally)

about, [Preface to the Fourth Edition](#)

Arduino and (see [Arduino](#))

Bluetooth setup, [Problem-See Also](#)

booting for the first time, [Problem](#)

booting from hard disk or USB flash drive, [Problem-See Also](#)

Camera Module installation, [Problem-See Also](#)

cases for, [Problem-See Also](#)

connecting a DVI or VGA monitor, [Problem](#)

connecting the monitor, keyboard, mouse, [Problem-See Also](#)

CPU temperature measured, [Problem](#)

enclosures for, [Problem-See Also](#)

GPIO connector (see [GPIO](#))

maximizing performance, [Problem-See Also](#)

model differences summarized, [Discussion](#)

model selection, [Preface to the Fourth Edition](#), [Problem-See Also](#)

models based on usage, [Solution](#), [Discussion](#)

monitor picture size adjustment, [Problem-See Also](#)

operating system (see [operating system](#))

password change, [Problem](#)

pinouts, [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero-Raspberry Pi Pico](#)

power supply, [Problem-See Also](#)

setup and management, [Introduction](#)  
(see also [setup](#))

shutting down, [Problem](#)

Raspberry Pi 1

overclocked, [Problem-See Also](#)

servomotor power supply, [Solution](#)

Raspberry Pi 2

overclocked, [Problem-See Also](#)

pinout, [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero](#)

Raspberry Pi 3 model B+

pinout, [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero](#)

recommendation, [Solution](#)

Raspberry Pi 4

about, [Preface to the Fourth Edition](#)

fan for, [Discussion](#)

power supply, [Solution](#)

Raspberry Pi 4 model A+ pinout, [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero](#)

Raspberry Pi 4 model B

basics, [Solution](#)

pinout, [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero](#)

Raspberry Pi 4 model B+ pinout, [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero](#)

Raspberry Pi 4 model Zero pinout, [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero](#)

Raspberry Pi 400

about, [Preface to the Fourth Edition](#)

basics, [Solution](#)

F10 key to turn on, [Solution](#), [Discussion](#)

GPIO connector, [Problem-See Also](#)

HAT adapter, [Discussion](#)

MonkMakes GPIO Adapter, [Solution](#)

pinout, [Raspberry Pi 400/4/3/2 Model B, B+, A+, Zero](#)

Raspberry Pi A+, [Discussion](#)

Raspberry Pi Camera Module

computer vision, [Problem](#)

installing, [Problem-See Also](#)

machine learning object recognition, [Introduction](#)

Raspberry Pi Compute 4, [Discussion](#)

Raspberry Pi Configuration tool

I2C setup, [Solution](#)

monitor underscan, [Solution](#)

network name changed, [Setting the network name using the Raspberry Pi Configuration tool](#)

one-wire interface, [Solution](#)

password change, [Solution](#)

serial interface enabled, [Solution](#)

SPI enabled, [Solution](#)

SSH enabled, [Solution](#)

VNC enabled, [Solution](#)

VNC virtual monitor, [Solution](#)

Raspberry Pi Imager, [Problem-See Also](#)

booting from hard disk or USB flash drive, [Problem-See Also](#)

documentation online, [See Also](#)

headless Raspberry Pi, [Problem-See Also](#)

media center software, [Solution](#)

SSH enabled, [Solution](#), [Solution](#)

Raspberry Pi model B rev. 1 pinout, [Raspberry Pi Model B revision 1](#)

Raspberry Pi model B rev. 2 pinout, [Raspberry Pi Model B revision 2, A](#)

Raspberry Pi OS (see operating system)

Raspberry Pi Pico, [Introduction-See Also](#)

about, [Introduction-Introduction](#)

analog (PWM) outputs, [Problem-See Also](#)  
analog inputs, [Introduction](#), [Introduction](#), [Problem-See Also](#)  
battery power, [Problem](#)  
breadboard with, [Problem-See Also](#)  
connecting to a computer, [Problem-See Also](#)  
digital inputs, [Problem-See Also](#)  
digital outputs, [Problem-See Also](#)  
filesystem, [Problem-See Also](#)  
firmware installed by download, [Solution-Solution](#)  
firmware installed losing filesystem contents, [Discussion](#)  
MicroPython installation, [Solution](#)  
Pico-compatible boards, [Solution-See Also](#)  
pinout, [Raspberry Pi Pico](#)  
pull-up resistor on board, [Discussion](#)  
Python shell, [Problem-See Also](#)  
RP2040 chip datasheet online, [See Also](#)  
second core (processor) use, [Problem](#)  
servomotor control, [Problem-See Also](#)  
Thonny support for, [Solution-See Also](#)

Raspberry Pi Pico W, [Introduction-See Also](#)  
about, [Introduction-Introduction](#)  
analog (PWM) outputs, [Problem-See Also](#)  
analog inputs, [Introduction](#), [Introduction](#), [Problem-Discussion](#)  
battery power, [Problem](#)  
breadboard with, [Problem-See Also](#)  
connecting to a computer, [Problem-See Also](#)  
digital inputs, [Problem-See Also](#)  
digital outputs, [Problem-See Also](#)



filesystem, [Problem-See Also](#)  
firmware installed by download, [Solution-Solution](#)  
firmware installed losing filesystem contents, [Discussion](#)  
MicroPython installation, [Solution](#)  
Pico-compatible boards, [Solution-See Also](#)  
pinout, [Raspberry Pi Pico](#)  
pull-up resistor on board, [Discussion](#)  
RP2040 chip datasheet online, [See Also](#)  
second core (processor) use, [Problem](#)  
servomotor control, [Problem-See Also](#)  
Thonny support for, [Solution-See Also](#)  
WiFi and Bluetooth, [Introduction](#)  
WiFi web server, [Problem-See Also](#)

Raspberry Pi Sense HAT (see [Sense HAT](#))  
Raspberry Pi Zero, [Solution](#), [See Also](#)  
Raspberry Pi Zero 2, [Problem](#)  
Raspberry Pi Zero 2 W, [Problem](#)  
Raspberry Pi Zero W  
basics, [Discussion](#)  
embedding, [Solution](#), [See Also](#)

Raspberry Squid Button, [Problem](#), [Solution](#)  
Raspberry Squid RGB LED, [Problem-See Also](#), [Solution](#), [Problem-See Also](#)  
Raspbian version of OS, [Problem](#)  
updating, [Problem](#)  
version of OS displayed, [Problem](#)

raspi-config utility  
camera configuration, [Solution](#)

- changing password, [See Also](#)
- I2C setup, [Solution](#)
- information online, [See Also](#)
- overclocking, [Problem](#)
- serial interface enabled, [Solution](#)
- setting hostname, [Setting the network name using the command line \(the easy way\)](#)
- sound output options, [Discussion](#)
- SPI enabled, [Solution](#)
- SSH enabled, [Solution](#)
- underscanning, [See Also](#)
- WiFi setup, [Setting up WiFi using the command line](#)

raw\_input command (Python 2), [Discussion](#)

RC-522 RFID card reader/writer, [Solution-See Also](#)

re library for regular expressions (Python), [Solution](#), [Solution](#)

read function (Python)

- computer vision camera, [Discussion](#), [Solution](#)
- file, [Solution](#)
- keypresses, [Solution](#)
- serial port, [Discussion](#)
- webpage, [Solution](#)

read function for files (MicroPython), [Discussion](#)

reading from a file, [Problem](#)

- JSON file, [Problem-See Also](#)
- Pico files, [Discussion](#)
- Pico files in Thonny, [Solution](#)

readline function (Python), [Discussion](#)

RealVNC client software, [Solution](#)

reboot command, [Discussion](#)

rebooting Raspberry Pi, [Problem](#)

Recommended Software tool, [Problem-See Also](#)

recording and playing back sound, [Solution](#)

redirecting input (<), [Discussion](#)

redirecting output (>)

- creating file without editor, [Problem](#)
- /dev/null, [Discussion](#), [Problem](#)
- to a file, [Problem](#)

reed switch sensing magnet, [Problem](#)

regular expressions (regex)

- for data entry validation, [Problem](#)
- for pattern searches, [Problem-See Also](#)
- for web scraping, [Problem](#)
- information online, [Solution](#), [See Also](#)
- tester online, [Discussion](#)

relays

- about, [Discussion](#)
- electromechanical for high-power switching, [Problem-See Also](#)
- PWM damaging, [Solution](#)
- solid-state for low-voltage switching, [Problem](#)

remote access to Raspberry Pi, [Solution](#)

- console cable connection, [Problem-See Also](#)
- networking, [Introduction](#)  
(see also networking)
- SSH for command line access, [Problem-See Also](#)
- VNC for graphical access, [Problem-See Also](#)

remove function (Python), [Discussion](#)

renaming files or folders, [Problem](#)

replace function (Python), [See Also](#), [Solution](#)

reset button, [Problem-See Also](#)

resistive sensors, [Problem-See Also](#), [Problem-See Also](#)

resistors

- calculator online for series resistor values, [See Also](#)
- calculator online for voltage divider, [See Also](#)
- chart online for colored stripe values, [See Also](#)
- converting 5V signals to 3.3V with, [Problem](#)
- LED connection resistor selection, [Discussion](#), [Discussion](#)
- LED connections requiring, [Discussion](#), [Discussion](#)
- measuring resistance, [Solution](#)
- methane gas sensor, [Discussion](#)
- MOSFET needing, [Discussion](#)
- photoresistors, [Problem-See Also](#)
- pull-up resistors, [Problem](#)
- pull-up resistors on Pico boards, [Discussion](#)
- rotary encoders for variable resistors, [Discussion](#)
- sources for, [Resistors and Capacitors](#)
- thermistors, [Problem-See Also](#)
- variable resistor step response, [Discussion](#)
- variable resistors as sensors, [Problem-See Also](#)
- voltage divider, [Solution-See Also](#)
- voltage divider resistor calculator online, [See Also](#)

resolution, monitor, [Problem-See Also](#)

RetroPie software, [Solution](#)

return value

- function return value, [Discussion](#)

list sort modifies original list, [Discussion](#)  
returning more than one, [Problem](#)  
string functions returning modified copies, [Discussion](#)

RFID reader/writer, [Problem-See Also](#)

RGB LEDs

CheerLights, [Problem-See Also](#)  
matrix display, [Problem-See Also](#)  
Raspberry Squid, [Problem-See Also](#)  
strip display, [Problem-See Also](#)  
user interface for changing color, [Problem-See Also](#)

rm command, [Solution](#)

alias requiring confirmation, [Discussion](#)

roll (orientation), [Solution](#)

rotary (quadrature) encoder, [Problem-See Also](#)

RP2040 chip datasheet online, [See Also](#)

(see also Raspberry Pi Pico; Raspberry Pi Pico W)

Pico-compatible boards, [Solution-See Also](#)

rptix project, [Problem-See Also](#)

## S

Samba for network-attached storage, [Problem-See Also](#)

Scalable Vector Graphics (SVG) drawings via Inkscape, [Problem](#)

screen captures, [Problem](#)

scrot command, [Solution](#)

SD card

booting for the first time, [Problem](#)  
booting from hard disk or USB flash drive, [Problem-See Also](#)  
determining free space on, [Problem](#)

installing operating system onto, [Problem-See Also](#)

selecting, [Discussion](#)

shutting down properly, [Discussion](#)

test utility, [Discussion](#)

trying different operating systems, [Discussion](#)

SDK (software development kit) for Python installation, [Solution](#)

searches using regular expressions, [Problem-See Also](#)

Secure Shell (see SSH)

self variable (Python), [Discussion](#)

Sense HAT

acceleration measurement with inertial measurement, [Problem-See Also](#)

basics, [Problem](#)

documentation online, [See Also](#), [See Also](#), [See Also](#)

IMU information online, [See Also](#)

LED matrix display, [Problem-See Also](#)

magnet sensing with, [Problem](#), [Problem](#)

magnetic north detected, [Problem](#)

programming reference online, [See Also](#)

Raspberry Pi 400 adapter for, [Discussion](#)

temperature/humidity/pressure measurement with, [Problem-See Also](#)

sensors, [Introduction-See Also](#)

acceleration measurement with MMA8452Q module, [Problem-See Also](#)

carbon dioxide concentration, [Problem-See Also](#)

CPU temperature measured, [Problem](#)

displaying readings on web page, [Problem-See Also](#), [Discussion-See Also](#)

displaying values, [Problem](#)

distance measurement with time-of-flight sensor, [Problem-See Also](#)

distance measurement with ultrasound, [Problem-See Also](#)

Inertial Measurement Unit, [Problem-See Also](#)

light measurement, [Problem-See Also](#)

logging to USB flash drive, [Problem](#)

magnet sensing with reed switch, [Problem](#)

magnet sensing with Sense HAT, [Problem](#), [Problem](#)

magnetic north detection with Sense HAT, [Problem](#)

methane detection, [Problem-See Also](#)

Pico analog temperature sensor, [Discussion](#)

Pico writing temperatures to file, [Problem-See Also](#)

Pimoroni Badger analog temperature sensor, [Discussion](#)

resistive, [Problem-See Also](#)

resistive sensors with ADC, [Problem-See Also](#)

sending data to ThingSpeak, [Problem-See Also](#)

Sense HAT, [Solution](#)  
(see also [Sense HAT](#))

smartcard and RFID reader/writer, [Problem-See Also](#)

soil moisture, [Problem-See Also](#), [Discussion-See Also](#)

temperature measurement, [Problem-See Also](#)  
(see also [temperature measurement](#))

temperature/humidity/pressure measurement with Sense HAT, [Problem-See Also](#)

touch interface with capacitive touch sensing, [Problem-See Also](#)

voltage measured higher than 3.3V, [Problem-See Also](#)

voltage measurement, [Problem-See Also](#), [Problem-Discussion](#)

serial interface for console cable connection, [Solution-See Also](#)  
tutorial online, [See Also](#)

Serial Peripheral Interface (SPI), [Problem-See Also](#)

serial port

GPS raw data, [Solution](#)

pySerial installation for access to, [Problem](#)

testing with Minicom, [Problem](#)

ServoBlaster, [Solution-See Also](#)

disabling, [Solution](#)

documentation online, [See Also](#)

no audio, [Solution](#), [Solution](#)

servomotors

about, [Discussion](#)

Adafruit servomotor HAT, [Problem-See Also](#)

continuous, [Discussion](#)

MonkMakes Servo Six board, [Discussion](#), [Discussion](#)

MonkMakes Servo Six board documentation, [See Also](#)

multiple motors precisely controlled, [Problem-See Also](#)

Pico or Pico W controlling, [Problem-See Also](#)

position controlled with PWM, [Problem-See Also](#)

power supply, [Discussion](#)

ServoBlaster, [Problem-See Also](#)

single motor precisely controlled, [Problem-See Also](#)

setup, Raspberry Pi, [Introduction-See Also](#)

Bluetooth, [Problem-See Also](#)

booting for the first time, [Problem](#)

booting from hard disk or USB flash drive, [Problem-See Also](#)

Camera Module installation, [Problem-See Also](#)

connecting a DVI or VGA monitor, [Problem](#)

connecting the monitor, keyboard, mouse, [Problem-See Also](#)

enclosure, [Problem-See Also](#)



headless Raspberry Pi, [Problem-See Also](#)  
maximizing performance, [Problem-See Also](#)  
model selection, [Preface to the Fourth Edition](#), [Problem-See Also](#)  
models based on usage, [Solution](#), [Discussion](#)  
monitor picture size adjustment, [Problem-See Also](#)  
operating system installation guide online, [See Also](#)  
operating system onto microSD card, [Problem-See Also](#)  
operating systems, [Problem](#)  
password change, [Problem](#)  
power supply, [Problem-See Also](#)  
shutting down, [Problem](#)  
time zone, [Discussion](#)

seven-segment LED display, [Problem-See Also](#)  
shebang for running Python files, [Discussion](#)  
Shell (Python) for Pico boards, [Problem-See Also](#)  
shutting down Raspberry Pi, [Problem](#)  
    low-power mode of shutdown, [Discussion](#)  
    reset button for power up, [Problem-See Also](#)

signal conversion  
    with level converter module, [Problem](#)  
    with two resistors, [Problem](#)

Simple Mail Transfer Protocol (SMTP), [Solution](#)  
SimpleMFRC522 library (Python), [Solution-See Also](#)  
    documentation online, [See Also](#)

Slack notifications via IFTTT, [Problem-See Also](#)  
sleep function (MicroPython), [Solution](#)  
sleep function (Python), [Problem](#)  
slide switch

- three-position (center-off), [Problem-See Also](#)
- two-position, [Problem](#)
- smartcard and RFID reader/writer, [Problem-See Also](#)
- SMTP (Simple Mail Transfer Protocol), [Solution](#)
- smtplib library (Python), [Solution](#)
  - documentation online, [See Also](#)
- software (generally)
  - installing with apt, [Problem](#)
  - removing with apt, [Problem](#)
- software (ready-made for Raspberry Pi), [Introduction-See Also](#)
  - Add/Remove Software tool, [Discussion](#)
  - bitmap image editor, [Problem-See Also](#)
  - Bookshelf for books and magazines, [Problem](#)
  - game console emulator, [Problem-See Also](#)
  - internet radio, [Problem-See Also](#)
  - laser cutter controller, [Problem-See Also](#)
  - media center, [Problem-See Also](#)
  - office software, [Problem-See Also](#)
  - radio transmitter, [Problem-See Also](#)
  - Recommended Software tool, [Problem-See Also](#)
  - vector image editor, [Problem](#)
  - Visual Studio Code, [Solution-See Also](#)
- software development kit (SDK) for Python installation, [Solution](#)
- soil moisture measured, [Problem-See Also](#), [Discussion-See Also](#)
- Sonoff WiFi Smart Switch
  - about, [Solution](#)
  - configuring, [Problem-See Also](#)
  - controlling with MQTT, [Problem-See Also](#)

- flashing for MQTT use, [Problem-See Also](#)
- Node-RED with, [Problem-See Also](#)
- sort command (Python), [Solution](#)
- sorting a list, [Problem](#)
- sound, [Introduction-See Also](#)
  - audio output without audio jack, [Problem-See Also](#)
  - buzzer, [Problem-See Also](#)
  - headphones connected, [Discussion](#)
  - loudspeaker connected, [Problem-See Also](#)
  - output options, [Problem](#)
  - playing sound file from command line, [Problem](#)
  - playing sound file from Python, [Problem](#)
  - recording and playing back, [Solution](#)
  - ServoBlaster and no audio, [Solution](#), [Solution](#)
  - speaker test Terminal command, [Solution](#)
  - USB microphone, [Problem-See Also](#)
- sounds identified via machine learning, [Problem-See Also](#)
  - spoken language in Python, [Problem-See Also](#)
  - spoken language locally, [Problem-See Also](#)
  - spoken language via cloud service, [Problem-See Also](#)
- source code fetched with git, [Problem-See Also](#)
- SparkFun
  - boost regulator module, [Solution](#)
  - gas sensor breakout board, [Solution](#)
  - I2C devices, [Discussion](#)
  - Motor Driver module product page, [See Also](#)
  - sensors offered, [See Also](#)
- speakers connected, [Problem-See Also](#)

SPI (Serial Peripheral Interface), [Problem-See Also](#)

split function (Python), [Solution](#)

spoken command recognized via ML

Edge Impulse cloud service, [Problem-See Also](#)

locally, [Problem-See Also](#)

Python, [Problem-See Also](#)

spreadsheet software, [Problem-See Also](#)

Squid Button, [Problem](#), [Solution](#)

Squid RGB LEDs, [Problem-See Also](#), [Solution](#)

CheerLights, [Problem-See Also](#)

SSD1306 driver chip for OLED display, [Solution](#)

SSH (Secure Shell)

enabled, [Solution](#), [Solution](#)

remote access to Raspberry Pi, [Problem-See Also](#)

Start button for Raspberry Pi, [See Also](#)

static IP address set, [Problem-See Also](#)

static-sensitive devices, [Solution](#)

Steinhart-Hart equation, [Discussion](#)

Stepper Motor HAT, [Problem](#)

stepper motors

about, [Solution](#), [Discussion](#)

bipolar, [Problem-See Also](#)

information online, [See Also](#), [See Also](#)

Stepper Motor HAT to control, [Problem-See Also](#)

unipolar, [Problem-See Also](#)

str function (Python), [Solution](#)

strings

array of substrings from, [Problem](#)

case sensitivity of functions, [Discussion](#)  
comparing, [Discussion](#)  
concatenation, [Problem](#)  
converting numbers to, [Problem](#)  
converting to numbers, [Problem](#)  
converting to uppercase or lowercase, [Problem](#)  
creating, [Problem](#)  
escape characters, [Discussion](#)  
extracting part of, [Problem](#)  
f-strings for evaluating code in strings, [Discussion](#)  
functions returning modified copies, [Discussion](#)  
length of, [Problem](#)  
list created by parsing, [Problem](#)  
list elements joined into, [Discussion](#)  
position within another string, [Problem](#)  
replacing characters within a string, [Problem](#)  
sublists from lists, [Problem](#)  
subprocess library (Python)  
    check\_output, [Discussion](#)  
    documentation online, [See Also](#)  
    IP address fetch, [Solution](#)  
    playing sound file from Python, [Solution](#)  
sudo command, [Solution](#)  
superuser privileges, [Problem](#)  
suppliers for parts, [Parts-Miscellaneous](#)  
    Octopart component search engine, [Parts](#)  
SVG (Scalable Vector Graphics) drawings via Inkscape, [Problem](#)  
switches

debouncing, [Problem-See Also](#)  
long wire requiring pull-up resistor, [Problem](#)  
Pico board digital inputs, [Solution-See Also](#)  
push switch connection, [Problem-See Also](#)  
push switch toggling, [Problem-See Also](#)  
reset button, [Problem-See Also](#)  
rotary (quadrature) encoder, [Problem-See Also](#)  
switch bounce, [Discussion](#), [Solution](#)  
three-position (center-off) toggle or slide switch, [Problem-See Also](#)  
two-position toggle or slide switch, [Problem](#)  
user interface to turn things on and off, [Problem](#)  
sys library for command-line arguments (Python), [Solution](#), [Solution](#),  
[Solution](#)  
system command (Python), [Solution](#)

## T

Tab key to autocomplete, [Discussion](#)  
tar command, [Solution](#)  
tarballs, [Discussion](#)  
Task Manager utility, [Solution](#)  
Tasmota firmware flashed, [Problem-See Also](#)  
    information online, [See Also](#)  
temperature measurement  
    CPU notification emailed via IFTTT, [Solution-See Also](#)  
    digital sensor for, [Problem-See Also](#)  
    Pico analog temperature sensor, [Discussion](#)  
    Pico writing to file, [Problem-See Also](#)  
    Pimoroni Badger using RP2040 sensor, [Discussion](#)

Raspberry Pi CPU, [Problem](#)

Sense HAT, [Problem-See Also](#)

thermistor for, [Problem-See Also](#)

TMP36 and ADC for, [Problem-See Also](#)

writing data to USB flash drive, [Solution](#)

TensorFlow for machine learning, [Introduction](#)

(see also machine learning)

information online, [See Also](#), [See Also](#)

Terminal

case sensitivity, [Discussion](#)

command aliases, [Problem](#)

command-line history, [Problem](#)

console cable serial connection, [Solution](#)

copying files onto a USB flash drive, [Discussion](#)

copying files or folders, [Problem-See Also](#)

CUPS installation, [Solution](#)

deleting file or directory, [Problem](#)

devices available for recording from, [Solution](#)

directory created, [Problem](#)

fetching files, [Problem](#)

filesystem navigation, [Problem-See Also](#)

help via manpages, [See Also](#)

hiding output to, [Problem](#)

installing software with apt, [Problem](#)

IP address display, [Solution](#)

Python 3 console, [Discussion](#)

Python programs run from, [Problem](#)

Python Shell for Pico, [Problem-See Also](#)

- raspi-config utility (see raspi-config utility)
- rebooting Raspberry Pi, [Solution](#)
- redirecting output to a file, [Problem](#)
- renaming files or folders, [Problem](#)
- running Python programs from, [Problem](#)
- setting hostname, [Setting the network name using the command line \(the easy way\)](#)
- sound file played from, [Problem](#)
- speaker test command, [Solution](#)
- SSH remote connection to Pi, [Solution](#)
- starting a session, [Problem-See Also](#)
- superuser privileges, [Problem](#)
- updating operating system, [Solution](#)
- WiFi setup, [Setting up WiFi using the command line](#)
- Windows PuTTY terminal software, [Solution](#)
- Tesseract (OCR software), [Problem](#)
- testing
  - breadboard power supply leads, [Solution](#)
  - breadboard with jumper leads, [Problem-See Also](#)
  - debugging in Python, [Discussion](#)
  - measuring resistance, [Solution](#)
  - microSD card test utility, [Discussion](#)
  - Python console, [Problem-See Also](#)
  - Python console command help, [Discussion](#)
  - Squid Button, [Discussion](#)
- text commands, [Problem-See Also](#)
  - (see also Terminal)
- text editors (see editors)



text extracted from image, [Problem](#)

thermistors, [Problem-See Also](#)

ThingSpeak

about, [Solution](#)

documentation online, [See Also](#)

sensor data sent to, [Problem-See Also](#)

tweets sent with, [Problem-See Also](#)

Thonny editor

MicroPython installation onto Pico, [Solution](#)

MicroPython interpreter, [Solution](#)

Pico and Pico W support, [Solution-See Also](#)

Python program editing, [Problem](#)

reading files written by Pico boards, [Solution](#)

`_thread` library (MicroPython), [Solution](#)

threading library (Python), [Solution-See Also](#)

threads in MicroPython, [Problem](#)

threads in Python, [Problem-See Also](#)

three-position (center-off) switch, [Problem-See Also](#)

throttling, [Solution](#)

time

device on and off times scheduled, [Problem-See Also](#)

formatting output of, [Problem](#)

setting, [Problem](#)

time library for sleep function (Python), [Solution](#)

time zone, [Discussion](#)

time-delaying sleep function (Python), [Problem](#)

time-of-flight (ToF) sensor

displaying values, [Problem](#)

- distance measurement, [Problem-See Also](#)
- time.sleep function (Python), [Discussion](#)
- TMP36 temperature sensor, [Problem-See Also](#)
  - datasheet online, [See Also](#)
- toggle switches
  - information online, [See Also](#)
  - three-position (center-off), [Problem-See Also](#)
  - two-position, [Problem](#)
  - types of, [Discussion](#)
- toggling with a push switch, [Problem-See Also](#)
- top command, [Discussion](#)
  - htop command for more information, [Discussion](#)
- touch command, [Discussion](#)
- touch sensors, [Problem-See Also](#)
- transistors
  - MOSFET, [Solution-See Also](#)
  - MOSFET datasheet online, [See Also](#)
  - MOSFET logic-level version, [Discussion](#)
  - MOSFET resistor, [Discussion](#)
  - phototransistors, [Solution](#), [Discussion](#)
  - relay for switching a high-power device, [Problem-See Also](#)
  - sources for, [Transistors and Diodes](#)
  - switching high-power LEDs with, [Problem-See Also](#)
- trimpot as variable resistor, [Solution](#)
  - (see also variable resistors)
- try/except construct (Python), [Discussion](#), [Solution](#)
- tuples (Python), [Solution](#)
  - machine learning events, [Discussion](#)

TV connection, [Problem-See Also](#)

Twitter

CheerLights project, [Problem](#)

IFTTT for sending notifications, [Problem-See Also](#)

responding using Dweet and IFTTT, [Problem-See Also](#)

tweets sent using ThingSpeak, [Problem-See Also](#)

## U

ULN2803 Darlington driver chip, [Solution](#)

ultrasonic rangefinder, [Problem-See Also](#)

unipolar stepper motors, [Problem-See Also](#)

updates

apt available software list, [Discussion](#)

network connection on first boot, [Solution](#)

NumPy library, [Solution](#), [Solution](#)

operating system, [Problem](#)

upper function (Python), [Solution](#)

urllib library for HTTP requests (Python), [Solution](#)

USB camera for computer vision, [Problem-See Also](#)  
(see also computer vision)

USB devices listed, [Problem](#)

USB flash drive

booting from, [Problem-See Also](#)

copying files onto, [Problem-See Also](#)

formatting, [Solution](#)

logging to, [Problem](#)

/media/pi folder, [Discussion](#)

USB hard drive for network-attached storage, [Problem-See Also](#)

USB hub for additional connectors, [Discussion](#)

USB keyboard keypresses intercepted, [Problem](#)

USB microphone, [Problem-See Also](#)

USB sound card, [Discussion](#)

user account

caution about username in code, [Solution](#)

changing password, [Problem](#)

creating, [Solution](#)

logging out, [Problem](#)

user interface

Adafruit servomotor HAT, [Solution](#)

browsing files in GUI, [Problem-See Also](#)

creating, [Problem-See Also](#)

Node-RED visual programming tool, [Problem-See Also](#)

PWM power for LEDs and motors, [Problem-See Also](#)

RGB LED color, [Problem-See Also](#)

sensor readings on a web page, [Discussion](#)

servomotor position control, [Solution](#)

speed control of DC motor, [Solution](#)

turning things on and off, [Problem](#)

VNC for remote access to Pi, [Problem-See Also](#)

username “pi”, [Solution](#)

\$ prompt, [Discussion](#)

utime library for sleep (MicroPython), [Solution](#)

## V

variable resistors

rotary (quadrature) encoders instead, [Discussion](#)

sensors, [Problem-See Also](#)

step response, [Discussion-See Also](#)

variables

argv for command-line arguments, [Solution](#)

arithmetic operators with, [Problem](#)

assigning values to, [Problem](#)

comparing values, [Problem-See Also](#)

displaying value of, [Problem](#)

evaluated while inside a string, [Discussion](#)

lists, [Problem](#)

(see also lists in Python)

loop variable, [Discussion](#)

member variables of class definitions, [Discussion](#)

self for class methods, [Discussion](#)

strings, [Problem](#)

(see also strings)

vector image editor, [Problem](#)

VGA monitor, connecting, [Problem](#)

Vim editor, [See Also](#)

vision (see computer vision)

Visual Studio Code (VS Code; Microsoft), [Solution-See Also](#)

VL53L1X I2C ToF sensor, [Solution-See Also](#)

VLC media player, [Problem-See Also](#)

playing sound file from Python, [Solution](#)

sound played from command line, [Solution](#)

VNC (Virtual Network Computing) for remote control of Pi, [Problem-See Also](#)

RealVNC client software, [Solution](#)

voltage conversion

converting 5V signals to 3.3V with two resistors, [Problem](#)  
with level converter module, [Problem](#)

voltage divider, [Solution-See Also](#)

about, [Discussion](#)

resistance converted to voltage, [Problem-See Also](#)

resistor calculator online, [See Also](#)

voltage measurement

using an ADC, [Problem-See Also](#)

voltage higher than 3.3V, [Problem-See Also](#)

voltmeters

analog meter as display, [Problem-See Also](#)

analog meter in MicroPython on Pico, [Problem-Discussion](#)

VS Code (Visual Studio Code; Microsoft), [Solution-See Also](#)

## W

WAV files played via aplay command, [Discussion](#)

web interface

about, [Discussion](#)

displaying sensor readings on web page, [Problem-See Also](#), [Discussion-See Also](#)

GPIO output control with, [Problem-See Also](#)

Node-RED visual programming tool, [Problem-See Also](#)  
(see also Node-RED)

reloading page via JavaScript, [Discussion](#)

web page for this book, [Solution](#)

web requests from Python, [Problem](#)

CPU too hot, [Solution](#)

IFTTT trigger, [Solution](#)

sensor data sent to ThingSpeak, [Solution-See Also](#)

tweet via ThingSpeak, [Solution](#)

tweets responded to via Dweet and IFTTT, [Problem-See Also](#)

web scraping, [Problem](#)

web server in Python, [Problem](#)

web server on Pico W, [Problem-See Also](#)

web server on Sonoff Smart Switch, [Discussion](#)

webcam

computer vision, [Problem-See Also](#)

(see also computer vision)

machine learning object recognition, [Introduction](#)

Wemos D1 WiFi board

Node-RED with, [Problem-See Also](#)

publishing MQTT messages, [Problem-See Also](#)

wget command, [Solution](#)

while statement (Python), [Solution](#)

whistling detected via machine learning, [Problem-See Also](#)

WiFi board

Node-Red with, [Problem-See Also](#)

publishing MQTT messages, [Problem-See Also](#)

WiFi connection setup, [Solution](#), [Problem](#)

adapters compatible with Raspberry Pi, [See Also](#)

power supply, [Discussion](#)

WiFi module in Raspberry Pi Pico W, [Introduction](#)

WiFi web server, [Problem-See Also](#)

WiFi Smart Switch

configuring, [Problem-See Also](#)

MQTT with, [Problem-See Also](#)

wildcard \* character

crontab for executing scripts, [Discussion](#)

finding files, [Discussion](#)

listing files, [Discussion](#)

removing files, [Solution](#)

Windows

console cable connection, [Solution](#)

network-attached storage connection, [Discussion](#)

PuTTY terminal software, [Solution](#)

SSH remote connection to Pi, [Solution](#)

VNC remote connection to Pi, [Problem-See Also](#)

wired networks, connecting to, [Problem-See Also](#)

word processing software, [Problem-See Also](#)

write function (Python)

file, [Solution](#)

serial port, [Discussion](#)

write function for files (MicroPython), [Discussion](#)

writing to a file (MicroPython), [Problem-See Also](#)

writing to a file (Python), [Problem](#)

CSV format to USB flash drive, [Problem](#)

## X

Xarchiver tool, [Solution](#), [Discussion](#)

## Y

yaw (orientation), [Solution](#)

## Z



---

ZIP file extracted, **Solution**, **Discussion**

## **About the Author**

**Dr. Simon Monk** (Preston, UK) has a degree in cybernetics and computer science and a PhD in software engineering. Simon spent several years as an academic before he returned to the industry, cofounding the mobile software company Momote Ltd. Simon divides his time between writing books and designing products for MonkMakes (the company his wife Linda runs). You can find out more about his books at <http://www.simonmonk.org> or follow him on Twitter [@simonmonk2](https://twitter.com/simonmonk2).

## Colophon

The animal on the cover of *Raspberry Pi Cookbook* is a Eurasian sparrowhawk (*Accipiter nisus*), which also goes by the name northern sparrowhawk, or simply sparrowhawk. This small bird of prey is found throughout the Old World. Adult males have bluish-gray upper plumage and orange-barred underparts; females and younger birds are all brown with brown-barred underparts. Females are up to 25% larger than males.

The sparrowhawk specializes in preying on woodland birds but can be found in many habitats, hunting garden birds in towns or cities. Males favor hunting smaller birds—finches and sparrows, for example—while females tend to catch thrushes and starlings and are capable of killing birds weighing up to 18 ounces or more.

Eurasian sparrowhawks breed in nests that are built with twigs and can measure up to two feet across. Afterward, four or five pale blue, brown-spotted eggs are laid. Success of breeding relies on females maintaining a high weight; it's the male's duty to deliver food to its mate during the nesting period. After 33 days, the chicks hatch, and they fledge after 24 to 28 days.

A juvenile sparrowhawk has a 34% chance of surviving its first year. After that, its chance of survival more than doubles, with 69% of adults surviving from one year to the next. The typical lifespan of these birds is four years, with mortality being greater for young males than for young females. The use of organochlorine insecticides to treat seeds before sowing has been known to incapacitate or kill sparrowhawks; those affected lay fragile-shelled eggs that break during incubation. Despite a sharp population decline after WWII, the sparrowhawk has become the most common bird of prey in Europe, due to the banning of such chemicals.

Its conservation status is currently classified as of Least Concern. Many of the animals on O'Reilly covers are endangered; all of them are important to the world.

The cover illustration is by Karen Montgomery, based on a black-and-white engraving from *Cassell's Natural History*. The cover fonts are Gilroy

Semibold and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag's Ubuntu Mono.