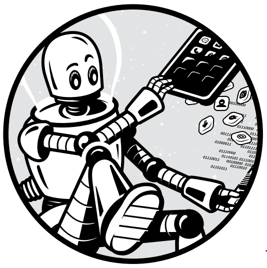
B
RESOURCES



This appendix contains information to help you get started with the projects. We cover how to find electronic components for purchase, how to power digital circuits, and how to set up a Raspberry Pi.

Buying Electronic Components for the Projects

Working with electronics and programming in a hands-on way helps you learn the concepts in this book, but trying to obtain various components can be intimidating. This section helps you find the electronic components you'll need for the projects.

Here's the full list of all the components used in the projects, in case you want to order everything at once:

- Breadboards (at least one 830-point breadboard. You can get by with only one breadboard if you plan to tear down each circuit between exercises. If you wish to keep your circuits intact, you need more than one.)
- Resistors (an assortment of resistors. Here are the specific values used: 47kΩ, 10kΩ, 4.7kΩ, 1kΩ, 470Ω, 330Ω, 220Ω.)
- Digital multimeter
- 9-volt battery
- 9-volt battery clip connector
- Pack of 5mm or 3mm red LEDs (light-emitting diodes)
- Two NPN BJT transistors, model 2N2222 in TO-92 packaging (also known as PN2222)
- Jumper wires designed for use in a breadboard (both male-to-male and male-to-female)
- Pushbuttons or slide switches that fit in a breadboard
- 7402 integrated circuit
- 7408 integrated circuit
- 7432 integrated circuit
- Two 7473 integrated circuits
- 7486 integrated circuit
- 220μF electrolytic capacitor
- 10μF electrolytic capacitor
- 5-volt power supply (See the section "Powering Digital Circuits" on page 336 for details.)
- Raspberry Pi and related items (See the section "Raspberry Pi" on page 341 for details.)
- Recommended: Alligator clips (These can make it easier to connect a battery to a breadboard or your multimeter to a circuit.)
- Optional: Wire stripper (You may need one to strip away plastic from the ends of wires and expose the copper.)

Although this list calls out specific counts for certain components, for some parts, you probably want to buy a few more than the stated number in case of damage or in case you wish to experiment. I recommend a few spare transistors and one spare of each integrated circuit.

7400 Part Numbers

Finding an appropriate 7400 series integrated circuit (IC) can be challenging, since these chips are identified with part numbers that include more detail than just the 74*xx* identifier. The 7400 series has a number of subfamilies, each with its own part numbering scheme. Plus, manufacturers add their own prefixes and suffixes to the part numbers. This can be confusing at first, so let's look at an example. I recently wanted to purchase a 7408 IC, but the part number I actually ordered was an SN74LS08N. Let's break down that part number in Figure B-1.

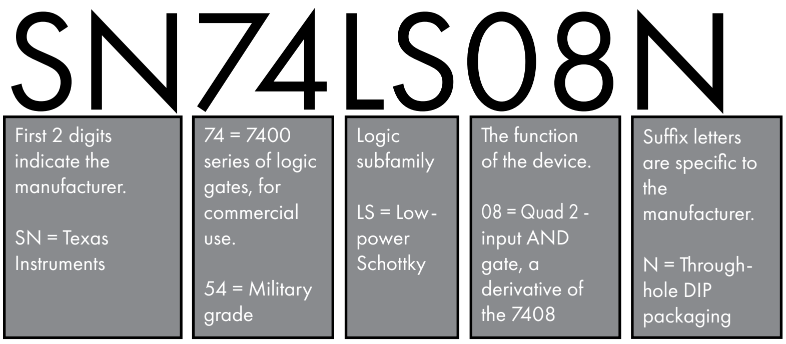

SN74LS08N
First 2 digits indicate the manufacturer. SN = Texas Instruments
74 = 7400 series of logic gates, for commercial use. 54 = Military grade
Logic subfamily LS = Low-power Schottky
The function of the device. 08 = Quad 2-input AND gate, a derivative of the 7408
Suffix letters are specific to the manufacturer. N = Through-hole DIP packaging

Figure B-1: Interpreting a 7400 series part number

So the SN74LS08N is a 7408 AND gate manufactured by Texas Instruments, in the low-power Schottky subfamily, packaged in through-hole packaging. No need to worry about the details of "low-power Schottky," other than to know that it's a common subfamily of parts that works for our purposes.

For the projects in this book, you want to ensure that you use parts that are compatible with each other. The projects assume that you are using parts that are compatible with the original 7400 logic levels (5V). Considering what is readily available, I'd recommend that you buy parts in the LS or HCT series. So if you want a 7408, you could buy an SN74**LS**08N or SN74**HCT**08N. Generally speaking, you should be able to mix LS and HCT parts in the same circuit. The prefix letters, SN in this example, don't matter when it comes to compatibility; you don't need to buy parts from a specific manufacturer. The suffix does matter, since it indicates the package type. Be sure to get parts that fit on a breadboard—N parts work well.

Shopping

If you happen to have a local electronics store nearby that can supply these parts, then I'd suggest you visit it. The store employees can help ensure that you get what you need. However, I've found that such stores are increasingly rare; you may not be able to get everything you need locally.

Your next option is to shop online. To make things easier for you, I've put together a web page with links to parts you need from various stores: *https://www.howcomputersreallywork.com/parts/*. Or if you want to shop around yourself, Table B-1 is a list of some of the popular stores where you can get parts; I've included notes on each. I'm not endorsing these shops in particular; you may find parts elsewhere. Of course, the status of these online stores could change after this book is published.

Table B-1: Shopping Online for Electronic Components

Store	Comments
Adafruit *https://www.adafruit.com/*	Adafruit has a great selection of electronics parts; however, the last time I checked, they don't carry individual 7400 series logic gates.
Amazon *https://www.amazon.com/*	Amazon carries popular items like the Raspberry Pi, but ordering integrated circuits often either isn't an option or is costly. Some items are only available in packs of 50 or 100, for example. You probably don't need 100 transistors, but if you are a Prime member, it actually might be more cost-effective to buy 100 transistors from Amazon than to buy 5 from other stores, once you factor in shipping.
Digi-Key Electronics *https://www.digikey.com/*	Digi-Key is targeted at professionals rather than hobbyists, and the site can be a bit intimidating, but you can find integrated circuits here, including the 7400 series logic circuits.
Mouser Electronics *https://www.mouser.com/*	Mouser is similar to Digi-Key in that it targets professionals and carries integrated circuits.
SparkFun *https://www.sparkfun.com/*	As with Adafruit, you can find lots of electronics here, but no 7400 logic gates, at least when I last checked.
Texas Instruments *https://store.ti.com/*	Texas Instruments manufactures 7400 series logic circuits, and they also sell them directly through their website. I've found that their prices are reasonable, and shipping options are good.

A number of websites are dedicated to helping people find electronic parts. These sites provide a user interface that's easy to navigate, and they allow for price comparison across multiple retailers. Two that I've found useful are Octopart (*https://octopart.com*) and Findchips (*https://www.findchips.com*).

Powering Digital Circuits

7400 series logic chips need 5V, so using a 9-volt battery isn't an option for powering these integrated circuits. Let's look at some options for powering your 7400 circuits.

USB Charger

Many smartphone chargers made since 2010 have a micro-USB connector. Around 2016, USB Type-C (or just USB-C) connectors began to become more common. Fortunately, USB of either variety provides 5V DC. So a USB charger, like the one shown in Figure B-2, is great for powering your 7400 series integrated circuits. If you're like me, you may have a bunch of old micro-USB phone chargers around your home already.

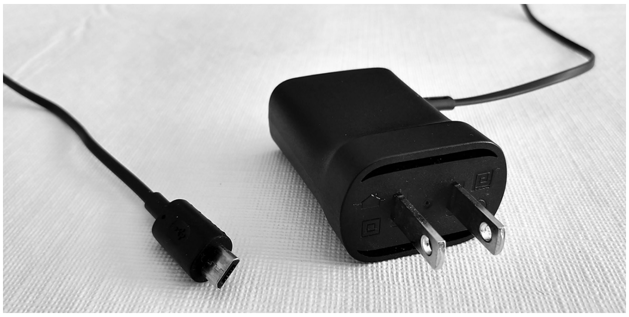


Figure B-2: A micro-USB phone charger

However, there's a challenge; a micro-USB connector doesn't plug into a breadboard, at least not without some help! A good option for using a USB charger with a breadboard is to buy a *micro-USB breakout board*, like the one shown in Figure B-3. This lets you plug your USB charger into the breakout board, and the breakout board plugs into your breadboard. Adafruit, SparkFun, and Amazon all carry these. Some soldering may be required. These boards typically have five pins, but for this purpose, you only need to concern yourself with the VCC (5V) pin and GND (ground) pin. When connecting this to a breadboard, remember to orient the pins in such a way that they are not connected to each other, as shown in Figure B-3.

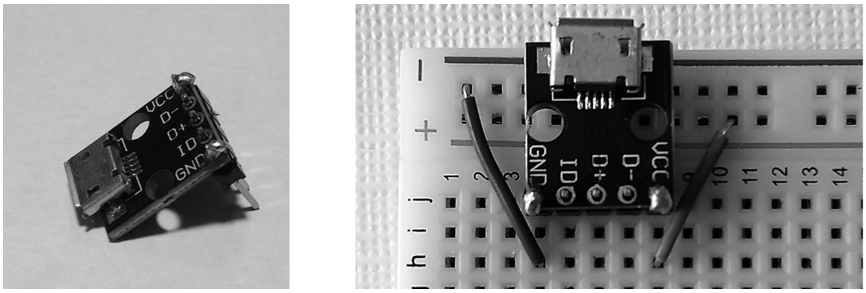


Figure B-3: A micro-USB breakout board, inserted in breadboard (right)

Breadboard Power Supply

Another option is to buy a breadboard power supply, like DFRobot DFR0140 or YwRobot Power MB V2 545043. These handy devices plug into your breadboard and are powered by a wall DC power supply with a 2.1mm barrel jack. The source DC supply should provide a voltage between 6V and 12V (be sure to verify the specific voltage allowed for the particular board you use). These 2.1mm DC power supplies are fairly common for powering consumer electronics—you may have several already—and this type of board just makes it easy to convert the voltage to 5V and connect it to a breadboard. Figure B-4 shows one of these common DC power supplies with a 2.1mm barrel jack along with a breadboard power supply.

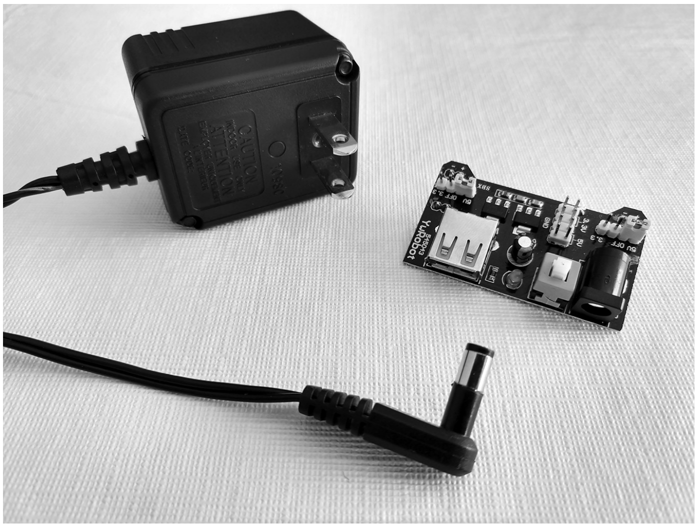


Figure B-4: A 2.1mm barrel jack power supply and a breadboard power supply

One word of caution: I've personally seen these boards experience a failure in their voltage regulator, causing the board to output a voltage higher than 5V. When connecting one of these power supplies, don't assume that the output voltage is actually 5V. Test the output voltage before connecting your circuit! Using a lower input DC voltage should help reduce this risk as well, so I'd recommend that you use a 9-volt or lower DC power supply, given an allowed range of 6V to 12V. These boards also have the option to output 3.3V instead of 5V, controlled with a jumper setting on the board, so make sure the jumpers are in the correct place.

Power from a Raspberry Pi

If you're going to buy a Raspberry Pi for the projects starting in Chapter 8, you're in luck; it has the side benefit of doubling as a 5-volt power supply! The GPIO pins on the Pi have various functions, but for this purpose, all you need to know is that pin 6 is ground and pin 2 supplies 5V. You can connect those pins to your breadboard as your power supply. See Figure 13-11 on page 312 for a GPIO pin diagram. There's no need to even install any Raspberry Pi software, because as soon as the Pi powers on, the 5-volt pin will turn on. Just connect your Pi to power. As a bonus, pin 1 can supply 3.3V if needed. To be clear, if you do this, you aren't using any of the computing power of the Pi; it is just acting as a 5-volt power supply. There is a limit to how much current the Raspberry Pi can supply. The power adapter you use for the Pi will have a maximum current rating, and the Pi itself will draw some current, maybe 300mA when idle. This probably goes without saying, but if you go with this option, be careful that you connect your circuit properly; you don't want to accidentally damage your Raspberry Pi! Figure B-5 shows a Raspberry Pi being used as a power supply.

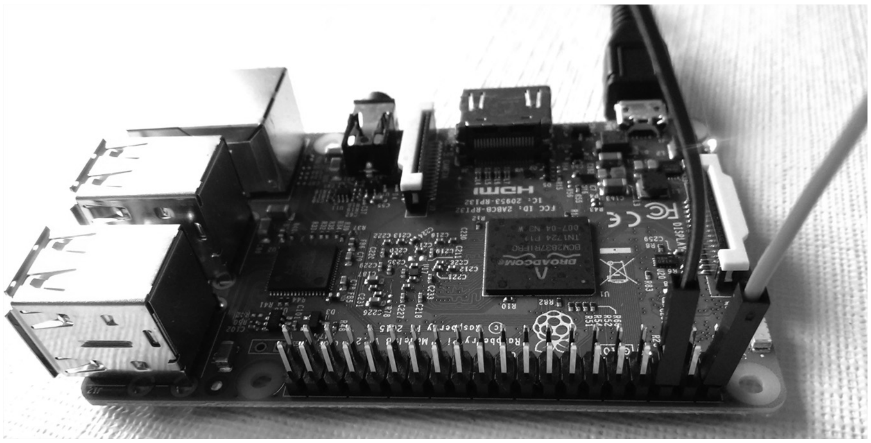


Figure B-5: Using a Raspberry Pi as a power supply

AA Batteries

You can also use AA batteries to power a digital circuit. A single AA battery supplies 1.5V, so three AA batteries can be connected in series to provide 4.5V. Although this voltage is less than the recommended voltage for 7400 series components, it should work for the circuits in this book, although your results may vary. You can buy a battery holder for three AA batteries and connect its output wires to your breadboard, as shown in Figure B-6.

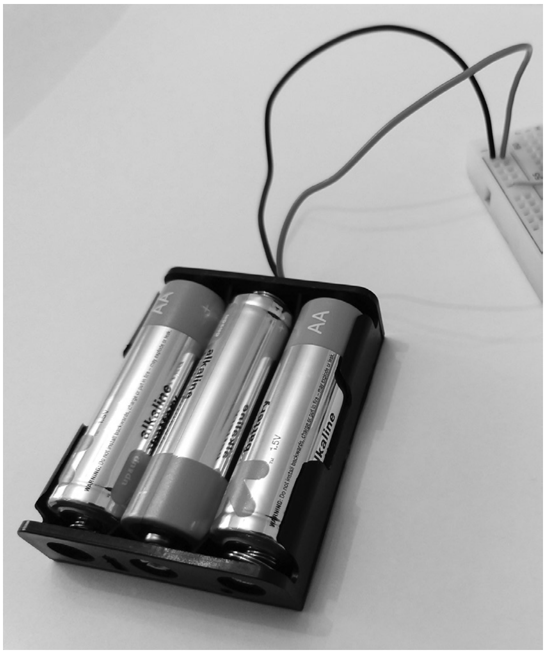


Figure B-6: Using three AA batteries to power a circuit on a breadboard

Troubleshooting Circuits

Sometimes you build a circuit, expecting things to work a certain way, but the result is something else entirely. Maybe the circuit doesn't appear to do anything, or maybe it behaves in a way you did not intend. Don't worry, this happens to everyone who builds circuits! It's easy to make a wiring mistake or to have a loose connection that throws everything off. Troubleshooting and diagnosing problems in a circuit is a valuable skill that can actually help you expand your understanding of how things work. Here I share some troubleshooting approaches that I use when my circuits don't work as expected.

If any component in your circuit is too hot to touch, immediately disconnect the circuit from its power source. Wiring mistakes can lead to a component overheating. This often causes damage to the component if it's left connected for more than a few seconds.

Your primary tool for circuit troubleshooting is a multimeter. With a multimeter, you can easily check the voltage at various points around a circuit. Ask yourself, "What do I expect the voltage to be at this point or at that point in my circuit?" For 5-volt digital circuits, the expected voltages are usually going to be approximately 0V or approximately 5V. If your meter shows an unexpected voltage anywhere in the circuit, ask yourself, "What things could influence this voltage?" and check those things.

For digital circuits, I usually take a "work backward" approach, starting with the component that is misbehaving. Confirm that its output voltage is wrong, and then check its inputs. Is one of those inputs also an unexpected voltage? If so, move back to the component that feeds that input, and check its output. Repeat until you find the source of the problem.

When checking voltages, I've found that the simplest approach is to connect your black/negative/COM lead to a ground point in your circuit and leave it there. If there isn't an obvious place to connect the lead to ground, just add a jumper wire to a ground point on the breadboard, then use an alligator clip to connect that jumper wire to your COM lead. With your COM lead anchored to ground, you can easily use your positive lead, usually colored red, to poke around at various points in the circuit and check voltages relative to ground.

The other thing I regularly check with a multimeter during troubleshooting is resistance. Sometimes I know the expected resistance between two points, and I want to verify that resistance value. If there's more than one path connecting the points you are measuring, be sure you know the expected resistance so you can correctly interpret your measurement.

Usually I'm checking resistance to simply ensure that two points are connected, in which case I expect a resistance of approximately 0Ω. Conversely, sometimes I want to ensure that two points are *not* connected, and then I look for a very high resistance, an open circuit. Some meters also include a *continuity check* feature, where the meter emits an audible tone if two points are connected. If you are just checking for connectivity, sometimes this is preferable to checking resistance.

Some specific things to verify when troubleshooting:

Breadboard power Does your breadboard have the appropriate voltage along the long power columns? The positive voltage column should equal the voltage from your source (say, a 9-volt battery or a 5-volt supply). Be sure to check both sides of the breadboard if both sides are in use.

Breadboard connections Verify that your wiring on the breadboard is sound. Are wires fully inserted, or do you have loose connections? Double-check the alignment of connections on the breadboard; is your wire in the right row? Is anything extra connected in the row you are checking?

Resistors Are your resistors the correct values? If needed, remove each one from the circuit and verify with a multimeter.

LEDs Are your LEDs oriented properly? The shorter lead should be connected closer to ground.

Capacitors If your capacitor is polarized, ensure you have the positive and negative leads properly oriented. Also check the capacitance value.

Integrated circuits Are your ICs properly connected to ground and to positive voltage? Is the chip fully seated in the breadboard, placed across the gap in the center? Check that your IC is aligned correctly by looking for the notch. Are you using the right part number?

Digital input switches/buttons When using pull-down resistors, is one side of the switch connected to positive voltage, and the other side connected through a pull-down resistor to ground? Is the digital input pin on the related chip connected to the same side of the switch as the pull-down resistor?

Raspberry Pi

The Raspberry Pi is a tiny, inexpensive computer. It was developed to promote teaching of computer science, and it has gained a following among technology enthusiasts. It's our computer of choice for this book, so here we cover the basics of setting up and using a Raspberry Pi.

Why Raspberry Pi

Before we go into the details of how to configure your Raspberry Pi, I'd like to explain why I chose the Raspberry Pi for this book. Some of the projects require a computer of some sort to interact with. Now, you may be saying to yourself, "I already own a computer; why do I need another?" Yes, since you are reading a book on computing you probably already own a computer, or maybe several! However, not everyone owns the same kind of computer, and some types of computing devices are better suited than others for education. Additionally, some of the projects in this book deal with low-level details of computing, so everyone following along needs the same kind of device.

The Raspberry Pi was a natural choice since it's inexpensive (about $35) and designed with computer education in mind. My goal isn't for you to switch to the Raspberry Pi as your primary computer or to make you a Raspberry Pi expert. Instead, we use the Raspberry Pi to learn core concepts that you can then apply to any computing device. The Raspberry Pi uses an ARM processor, and we'll be running *Raspberry Pi OS* (previously called *Raspbian*) on it, a version of Linux optimized for the Raspberry Pi.

Parts Needed

First things first, you are going to need to obtain a Raspberry Pi and some accessories. Here's what you need:

- **Raspberry Pi.** These are usually about $35 and can be purchased online. At the time of this writing, the latest model is the Raspberry Pi 4 Model B, and the exercises in this book were tested with this version and with the Raspberry Pi 3 Model B+. If a newer model is released, it likely will be acceptable as well, given the Raspberry Pi's track record of backward compatibility. The Raspberry Pi 4 Model B is available in multiple memory configurations (1GB, 2GB, 4GB, and 8GB)—any of them are fine for this book.
- **USB-C power supply (only for Raspberry Pi 4).** The Raspberry Pi 4 uses a USB-C power supply. The power supply needs to provide 5V and at least 3A. Certain USB-C power supplies are incompatible with some Raspberry Pi 4 devices, so I recommend you purchase a USB-C power supply specifically designed for the Raspberry Pi 4.
- **Micro-USB power supply (only for Raspberry Pi 3).** Unlike the Raspberry Pi 4, the Raspberry Pi 3 is powered by a micro-USB power adapter, like the ones that many smartphones use. If you have a smartphone charger already, it might work with the Pi. Just be sure that the connector is micro-USB. The standard voltage output from such chargers is 5V, but the maximum amount of current they supply varies. For the Raspberry Pi 3, it's recommended that you use a power supply that can provide at least 2.5A. The current requirements vary depending on what you have connected to the Pi. So check your smartphone charger to see how much current it can supply; you may need to purchase a micro-USB power supply designed for the Pi.
- **MicroSD card, 8GB or larger.** The Raspberry Pi doesn't come with any storage, so you need to add it yourself via a microSD card. These cards are commonly used in smartphones and cameras, so you may have an extra one lying around already. The process of installing Raspberry Pi OS will erase existing data, so be sure to back up anything you have stored on your microSD card.
- **USB keyboard and USB mouse.** Any standard USB keyboard and USB mouse will do.
- **Television or monitor that supports HDMI.** All modern televisions and many computer monitors support HDMI connections.

- **HDMI cable.** The Raspberry Pi 3 uses a standard, full-sized HDMI cable, but the Raspberry Pi 4 has a micro-HDMI port. Assuming your display device accepts full-sized HDMI input, this means that for a Raspberry Pi 4 you need a micro-HDMI to HDMI cable or adapter.
- **Optional: Raspberry Pi case.** This isn't required, but it is nice to have. Note that the Raspberry Pi 3 and Raspberry Pi 4 have different physical layouts, so they need differently shaped cases.

See the "Shopping" section of "Buying Electronic Components" earlier in this appendix for help getting these parts.

Setting Up a Raspberry Pi

The Raspberry Pi website (*https://www.raspberrypi.org*) contains detailed setup guides that walk through setting up a Raspberry Pi. I don't cover all the details here since the online documentation already exists, and it changes over time. Instead let me give you a brief overview of the steps required.

You have several options for installing Raspberry Pi OS on your Raspberry Pi. If you have a computer with a microSD card reader/writer, the simplest option is to use the *Raspberry Pi Imager*. Here's how to use this tool to get going quickly with your Raspberry Pi:

1. Insert your microSD card into your computer.
2. Download the Raspberry Pi Imager from *https://www.raspberrypi.org/downloads*.
3. Install and run the Raspberry Pi Imager on your computer.
4. Choose the operating system: Raspberry Pi OS (32-bit).
5. Choose the SD card you want to use.
6. Click "Write" and Raspberry Pi OS will be copied to your microSD card.
7. Remove the microSD card from your computer.
8. Insert the microSD card into the Raspberry Pi.
9. Connect the Raspberry Pi to a USB keyboard, USB mouse, and monitor or TV using HDMI, and finally to the power supply.
10. The Raspberry Pi should boot into Raspberry Pi OS.

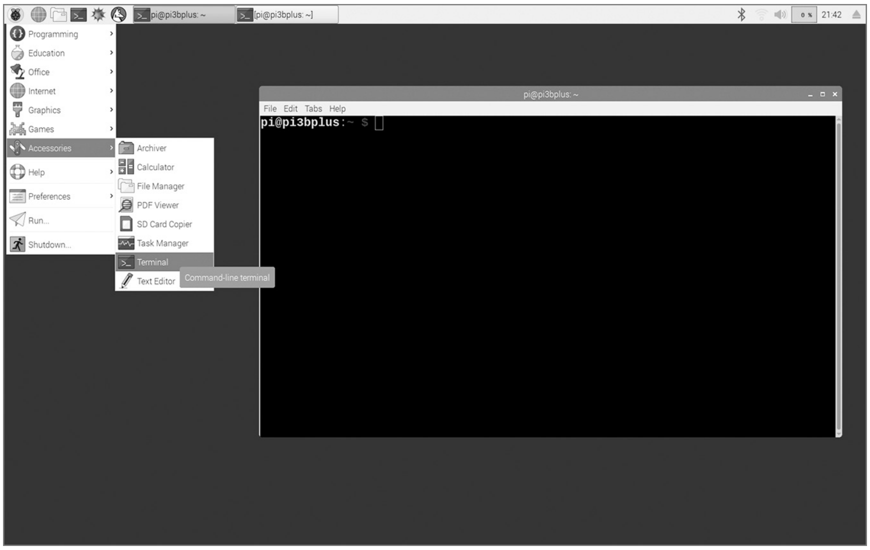
Another good option for installing Raspberry Pi OS is to use the Raspberry Pi *New Out of Box Software (NOOBS)*. To use NOOBS, download it from *https://www.raspberrypi.org/downloads* and copy it to a blank microSD card. If you don't have another computer available to do this, you can buy a microSD card with a copy of NOOBS preloaded. In either case, once you have NOOBS on the microSD card, insert the card into your Raspberry Pi and power it on. Follow the on-screen instructions to install Raspberry Pi OS.

NOTE *At the time of this writing, a 64-bit version of Raspberry Pi OS is available as a beta release. However, the projects in this book were tested with the 32-bit Raspberry Pi OS, so I recommend that you stick with the 32-bit version for the projects.*

Using Raspberry Pi OS

Once your Raspberry Pi is set up, I recommend you take a little time to familiarize yourself with the user interface of Raspberry Pi OS. If you've used a Mac or a Windows PC before, the Raspberry Pi OS desktop environment should be somewhat familiar. You can open applications in windows, move those windows around, close them, and so forth.

That said, most of the projects in the book won't require you to use any of the Pi's graphical applications. Nearly everything can be done from a terminal, and most projects require at least some use of the terminal, so let's take a minute to get familiar with it. From the Raspberry Pi OS desktop, you can open a terminal window by clicking **Raspberry** (icon in the upper-left corner) ▸ **Accessories** ▸ **Terminal** as shown in Figure B-7.



```
pi@pi3bplus:~ $
```

Figure B-7: Opening the Raspberry Pi terminal

The terminal is a *command line interface (CLI)*, where everything you do is accomplished by typing in commands. Raspberry Pi OS, like all versions of Linux, has excellent support for the CLI, and you can do just about anything from a terminal if you know the right commands. By default, the terminal on Raspberry Pi OS runs a shell called *bash*. A *shell* is a user interface for an operating system, which can be either graphical (like the desktop) or command line based. The initial text in the bash command line should look something like this:

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

Let's examine each part of that text string:

pi This is the username of the currently logged-on user. The default user's name is "pi."

raspberrypi Separated from the username by an @ sign, this is the name of the computer.

~ This indicates the current directory (folder) you are working in. The ~ character has a special meaning; it refers to the current user's home directory.

$ The dollar sign is the CLI prompt, an indicator that you can type your commands here.

In this book, when I list commands that should be typed in a terminal, I prefix the line with a $ prompt. As an example, this command lists the files in the current directory:

```
$ ls
```

To run a command, you don't need to type the dollar sign; just type the text following it, and then press ENTER. If you want to run a command that you previously entered, you can press the up arrow on the keyboard to cycle through previously issued commands.

If you prefer to work in a terminal, you can set up your Raspberry Pi to boot directly to a command line rather than to the desktop: **Raspberry ▸ Preferences ▸ Raspberry Pi Configuration ▸ System tab ▸ Boot ▸ To CLI**. Once you've made this configuration change, the next start of the system goes directly to the CLI rather than the desktop. While in a CLI-only environment, if you want to start the desktop environment, just run the following command:

```
$ startx
```

As a terminal user, another option is to control your Raspberry Pi from a different computer or even from a phone by using *Secure Shell (SSH)* over the network. The end result of this approach is that your Pi can run anywhere on your network, even without a monitor or keyboard attached, and you can use the keyboard and monitor of another device to control it. To pull this off, you must enable SSH on the Pi (**Raspberry ▸ Preferences ▸ Raspberry Pi Configuration ▸ Interfaces tab ▸ SSH ▸ Enable**), and then run an SSH client application on a second device. I won't include detailed steps for setting this up, but there are plenty of guides online for how to do this.

When you are finished working with your Pi for a while, you'll want to shut it down gracefully to avoid data corruption, rather than just turning it off. From the desktop, you can shut down the system with **Raspberry ▸ Shutdown... ▸ Shutdown**. Or from a terminal, you can use this command to halt the system:

```
$ sudo shutdown -h now
```

You'll know the system has completely shut down when the attached monitor no longer displays anything and the activity light on the Raspberry Pi board stops blinking. You can then unplug your Pi.

Working with Files and Folders

The projects in this book regularly direct you to create or edit text files and then run some terminal commands on them. Let's talk about working with files and folders in Raspberry Pi OS, both from a command line and from the graphical desktop. Operating systems organize the data on a storage device, like the microSD card in your Raspberry Pi, with a filesystem. A *file* is a container of data, and a *folder* (also known as a directory) is a container of files or other folders. The filesystem's structure is a *hierarchy*, a tree of folders. On Linux systems, the root of that hierarchy is represented with */*. The root is the topmost folder—all other folders and files are "under" the root.

A folder directly under the root folder would be represented like this: */<foldername>*. A text file in that folder would look something like this: */<foldername>/<filename>.txt*. Note the *.txt* file extension, the last part of the filename. It is convention to end a filename with a period followed by a few characters to indicate what kind of data is in the file. In the case of text files, "txt" is often used. The use of file extensions is not a requirement, but it is a common practice, helpful for keeping your data organized.

Every user in Raspberry Pi OS has a home folder to work in. The default user in Raspberry Pi OS is named pi, and the pi user's home folder is located in */home/pi*. While you are logged on as the pi user, that same home folder can also be referred to with the ~ character. Let's say you create a folder in your home folder called *pizza*. Its full path would be */home/pi/pizza*, or when logged in as pi, you can refer to it as *~/pizza*. Let's try creating a *pizza* folder from a terminal window, using the **mkdir** command, short for "make directory." Don't forget to press ENTER after typing the command.

```
$ mkdir pizza
```

From a terminal, you can see your newly created folder by using the **ls** command:

```
$ ls
```

When you type **ls** and press ENTER, you should see the *pizza* folder alongside a set of other folders that were already present in your home folder, such as *Desktop*, *Downloads*, and *Pictures*.

The terminal isn't the only way to view the files in a folder. You can also use the File Manager application, which you can launch from **Raspberry ▸ Accessories ▸ File Manager**. As shown in Figure B-8, the File Manager application opens with a default view of your home directory.

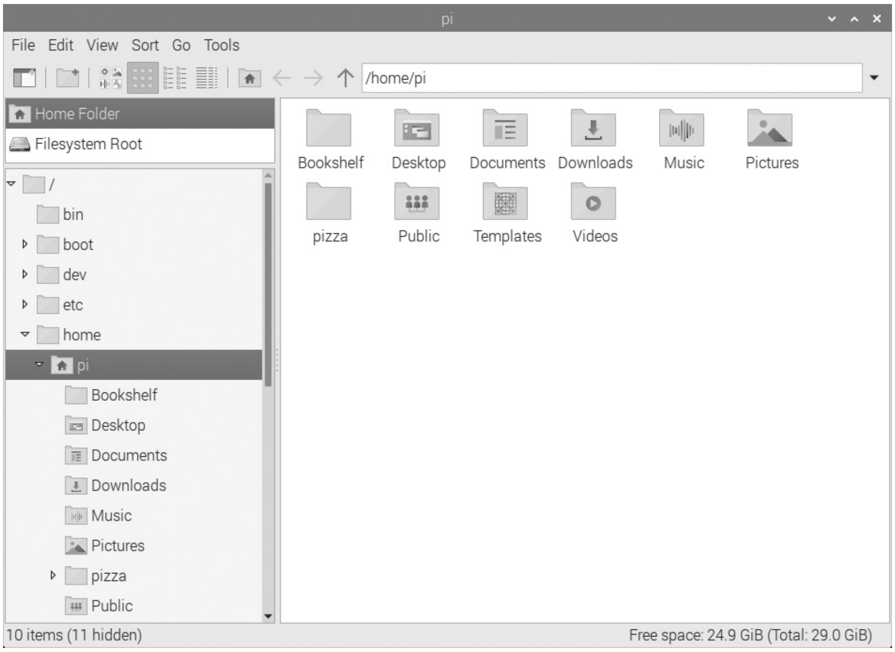


Figure B-8: Raspberry Pi OS File Manager

The left-hand side of File Manager shows the full filesystem hierarchy of folders, with the currently selected folder highlighted—*pi* in this case. The address bar at the top showing */home/pi* indicates the current folder. Now, try double-clicking the *pizza* folder; it should be empty. Let's jump back to the terminal window and create some files in this folder. First, let's change folders with the **`cd`** command (for change directory), so that our current folder is the *pizza* folder. Then let's create two empty files using the **`touch`** command. Finally, we'll list the contents of the directory with **`ls`**, expecting to see the two new filenames listed.

```
$ cd pizza
$ touch cheese.txt
$ touch crust.txt
$ ls
```

Note that when you changed to the *pizza* folder, your bash prompt should have changed as well. It should now include `~/pizza` before the `$`, indicating the current folder. Now look at the File Manager application window; it should also show the two new files under the *pizza* folder, as shown in Figure B-9.

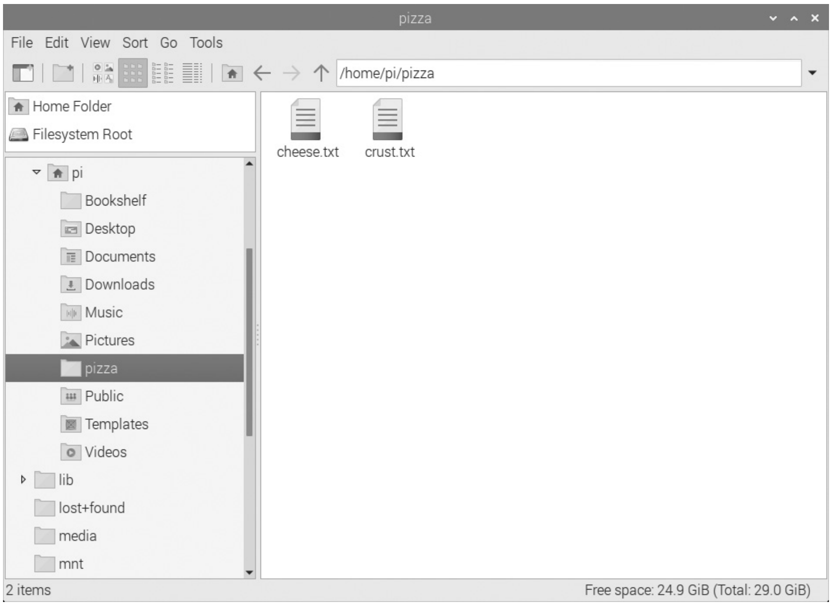
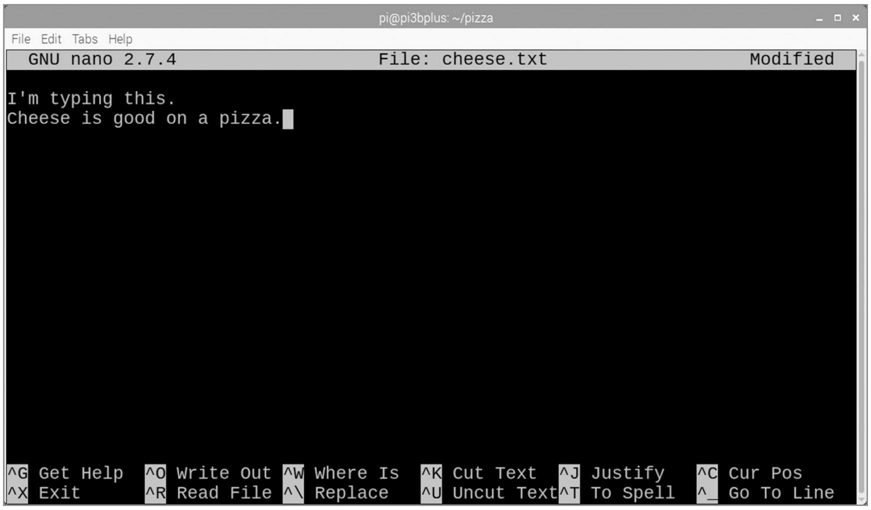


Figure B-9: Raspberry Pi OS File Manager: files in the pizza folder

Now we have two empty files in the *pizza* folder. Let's add text content to the files. First, we'll edit *cheese.txt* using a command line text editor called `nano`.

```
$ nano cheese.txt
```

With the `nano` editor window open in the terminal, you can type text that will be saved to *cheese.txt*. Keep in mind that `nano` is a command line application—you can't use the mouse. You need to use the arrow keys to move the cursor. Try typing some text, as shown in Figure B-10.



```
  GNU nano 2.7.4                   File: cheese.txt                   Modified

I'm typing this.
Cheese is good on a pizza.

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

Figure B-10: Using nano to edit cheese.txt

After typing in some text in nano, press CTRL-X to exit nano. The editor prompts you to save your work ("Save modified buffer?"). This may seem like an odd question, but don't let the "buffer" term throw you off—nano is just asking if you want to save the text you entered to the file. Press **Y**, then hit ENTER to accept the suggested filename (*cheese.txt*).

I often use nano since it works from a terminal, but you may prefer to edit your text files using a graphical text editor. The Raspberry Pi OS desktop environment includes a handy text editor which you can launch from **Raspberry ▸ Accessories ▸ Text Editor**. At the time of this writing, this opens an editor called Mousepad. Let's try modifying *cheese.txt* further in this text editor. First, we need to open the file by performing the following within the text editor: **File ▸ Open ▸ Home ▸ pizza ▸ cheese.txt ▸ Open** (button), as shown in Figure B-11.

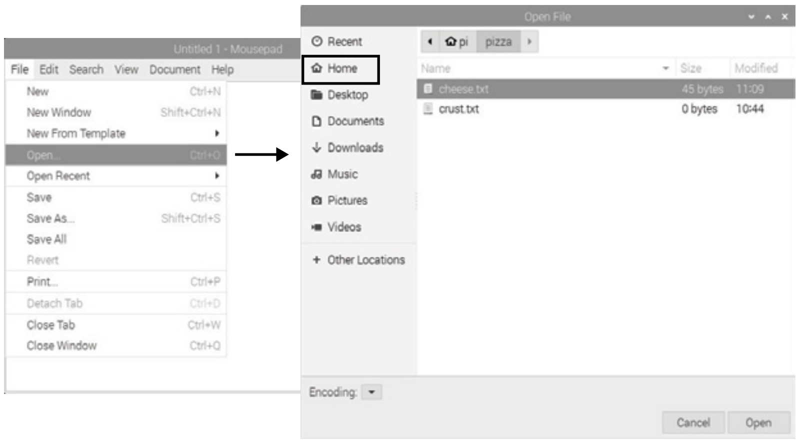


Figure B-11: Opening cheese.txt *in the text editor*

Once you have the file open in the text editor, you should see the text you typed earlier. You can edit the text as you wish. Then save your changes with **File ▸ Save**.

In addition to editing existing files, you can also use the text editor to create new files. Just launch a new editor window (**Raspberry ▸ Accessories ▸ Text Editor**) and click **File ▸ Save**. This will prompt you to save your file in the folder of your choice, with a name of your choosing. You can edit the text of your new file and save your changes as needed by using **File ▸ Save** again. Or if you want to save an existing file with a new name, choose **File ▸ Save As…** instead.

If you prefer to use nano to create a new file, from a terminal window first change to the folder where you want to save your file (if needed), and then type **nano *filename***, as shown in this example:

```
$ nano new-file.txt
```

Type your text, and when you exit `nano`, you'll be prompted to save this new file.

We've just covered ways to view, edit, and create text files in Raspberry Pi OS. If you want to remain in a terminal, `nano` is a good choice. If you prefer the desktop, then the text editor Mousepad should meet your needs. There are also other editors included with Raspberry Pi OS. Geany is a programmer's text editor, and Thonny Python IDE is tailored for Python programming. Both are found under **Raspberry ▸ Programming**. In this book's projects, I'll leave it up to you to decide which text editor you use.

You may also wish to manage your files and folders in other ways—move files around, delete files, and so forth. You can do all of this from File Manager, or you can do it from a terminal window. Here are some commands you can use from a bash prompt to get you started:

`cd` ***`folder`*** Change the current directory (folder).

`mkdir` ***`folder`*** Make a directory.

`rm` ***`file`*** Delete a file.

`rm -rf` ***`folder`*** Delete a folder and its contents, including its subfolders.

`mv` ***`file file2`*** Rename a file.

`mv` ***`file folder/`*** Move a file from one location to another.

`cp` ***`file folder/`*** Copy a file from one location to another.