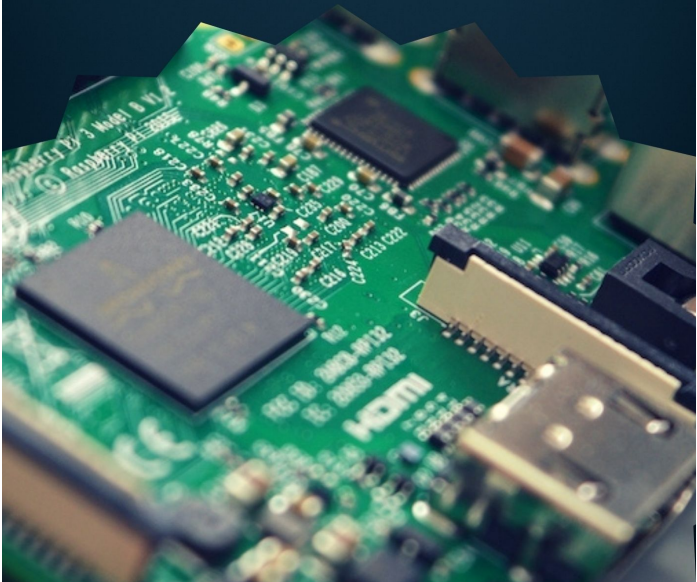


RASPBERRY PI

A “GETTING STARTED” GUIDE



Peter Dalmaris, PhD

Raspberry Pi

Getting Started

Welcome to this special collection of articles, meticulously curated from the Tech Explorations blog and guides. As a token of appreciation for joining our email list, we offer these documents for you to download at no cost. Our aim is to provide you with valuable insights and knowledge in a convenient format. You can read these PDFs on your device, or print.

Please note that these PDFs are derived from our blog posts and articles with limited editing. We prioritize updating content and ensuring all links are functional, striving to enhance quality continually. However, the editing level does not match the comprehensive standards applied to our Tech Explorations books and courses.

We regularly update these documents to include the latest content from our website, ensuring you have access to fresh and relevant information.

License statement for the PDF documents on this page

Permitted Use: This document is available for both educational and commercial purposes, subject to the terms and conditions outlined in this license statement.

Author and Ownership: The author of this work is Peter Dalmaris, and the owner of the Intellectual Property is Tech Explorations (<https://techexplorations.com>). All rights are reserved.

Credit Requirement: Any use of this document, whether in part or in full, for educational or commercial purposes, must include clear and visible credit to Peter Dalmaris as the author and Tech Explorations as the owner of the Intellectual Property. The credit must be displayed in any copies, distributions, or derivative works and must include a link to <https://techexplorations.com>.

Restrictions: This license does not grant permission to sell the document or any of its parts without explicit written consent from Peter Dalmaris and Tech Explorations. The document must not be modified, altered, or used in a way that suggests endorsement by the author or Tech Explorations without their explicit written consent.

Liability: The document is provided "as is," without warranty of any kind, express or implied. In no event shall the author or Tech Explorations be liable for any claim, damages, or other liability arising from the use of the document.

By using this document, you agree to abide by the terms of this license. Failure to comply with these terms may result in legal action and termination of the license granted herein.

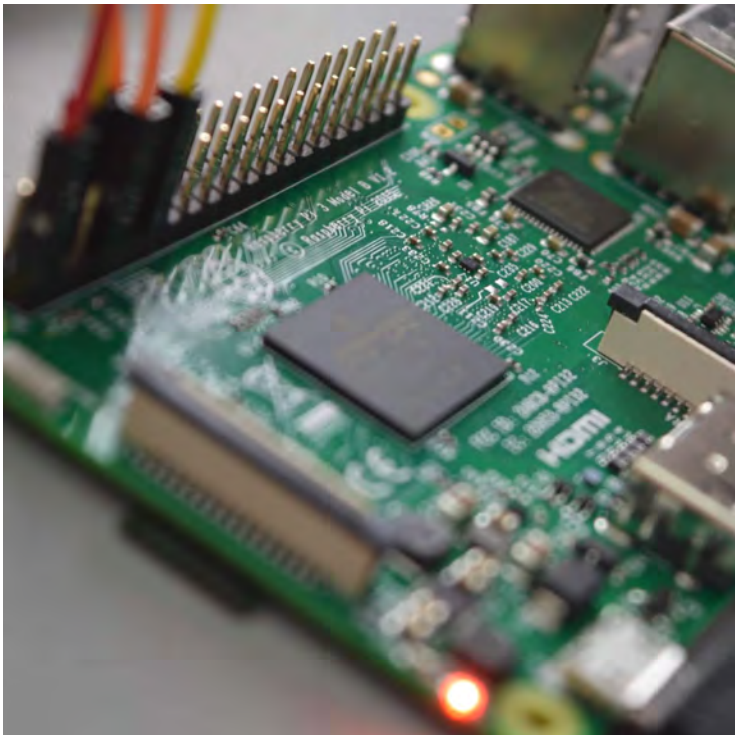
1. What is the Raspberry Pi?

RASPBERRY PI GETTING STARTED Guide SERIES

Introduction To The Raspberry Pi: What Is It?

The Raspberry Pi is a low cost computer, popular with people who want direct access to its hardware.

It has revolutionised computer education by combining low price with accessibility.



The student, for the first time since the early days of the PC revolution, has direct physical access to the hardware. The Raspberry Pi is made for learning. To make the most of it, you must spend time to gain an understanding of the basics of its hardware, its operating system, programming, and the peripherals that you can connect to it.

The Raspberry Pi models

The Raspberry Pi, at the time I am creating this project, is available in several different models.

Model A, Model A+, Model B and Model B+. In this project I am using the Model B.

RPi 2



RPi A



RPi B



RPi Compute module

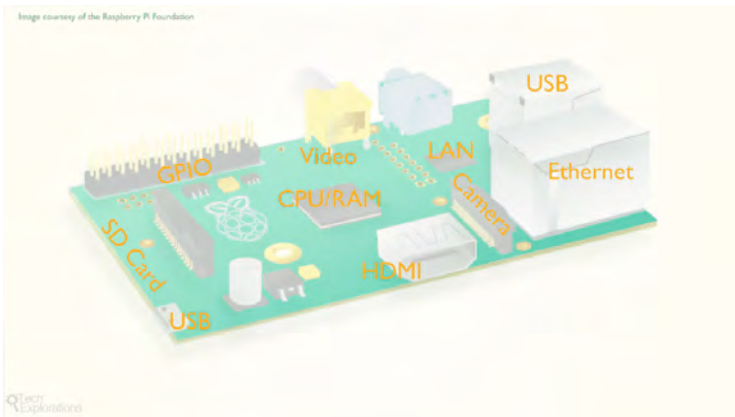


Some of the available Raspberry Pi models

All Raspberry Pi's share some common features. Looking at the circuit board below the list, you can see:

- The Processor and RAM chip

- The LAN controller chip
- A HDMI video output connector
- A composite analog video connector (on models A and B)
- An SD card connector
- A micro-USB power connector
- A USB port
- An Ethernet port (on model Bs)
- A camera connector
- And the very important GPIO headers



The Raspberry Pi, common components

The Raspberry Pi comes as a single PCB. No keyboard and mouse, no screen, not even a power supply. You have to provide all that. In this project, we will be using the Raspberry Pi in so called “headless” mode. This means that we will not be connecting it to a keyboard or mouse. Instead, we will work with the Raspberry Pi via an SSH network connection. Don’t worry if you don’t quite understand what this means, I will show you everything you need to know, step by step.

The operating system

As a computer, and unlike micro-controllers like the Arduino, the Raspberry Pi needs an operating system. There are several options to choose from:

- Raspbian, the Raspberry Pi Foundation's preferred operating system distribution
- Ubuntu,
- Openelec
- OSMC
- Pidora
- RISC OS
- Minibian, a minimal version of Raspbian.

All of them except for RISC OS are flavours of Linux.

Minibian is a minimalist version of Raspbian. It keeps everything that is important and throws away the graphical user interface and a few other things that are not really needed for our purposes. In return, we get a small disk footprint so we can use even small 4GByte SD Cards.

Of course, you can use Raspbian, which is the "official" operating system offered by the Raspberry Pi Foundation.

In this guide, you will learn how to install Minibian. The process for installing Raspbian is identical.

The Raspberry Pi comes with 512 MBytes or 1GByte of RAM, depending on the model. Video memory is shared with general purpose memory. In our project, we will not be using any video output, so we will configure our Pi to not use any video memory.

What can you do with a Raspberry Pi?

Although this amount of RAM may seem too little at a time when computers come with multiple of gigabytes, for an embedded computer it is more than enough. People run multiplayer game servers like Minecraft on it, and small production web servers and database server. Others have even used the RPi as a node for small supercomputers. With a bit of planning, the Raspberry Pi can do amazing things.



Raspberry Pi running a Minecraft server

To make something useful with the Raspberry Pi, just like with any computer, you need application software. You can either download ready made software, or write your own. In this guide, I will show you both. You will download, install and configure various types of servers, and you will write your own programs in Python. You can learn how to create a full web application by enrolling to my course [Raspberry Pi Full Stack Raspbian](#).

You will not become an expert Python programmer, but you will become familiar with it enough to be both useful and dangerous. That's a great start!

The Raspberry Pi rarely works in isolation. It has a fast

Ethernet communications socket through which you can connect it to the Internet. The newer Raspberry Pi have integrated Wifi and Bluetooth capabilities, which makes communications even easier. In Raspberry Pi Full Stack, we take advantage of this capability and make it possible for our application to interact with Internet based web services. You will also be able to access your application via a web browser, potentially making it possible to access your Raspberry Pi from anywhere in the world.

Ok, enough with this general introduction to the Raspberry Pi. In the **next article**, I will show you how to instal the Raspbian operating system on your Raspberry Pi.

2: Raspberry Pi vs Arduino

RASPBERRY PI GETTING STARTED GUIDE SERIES

Raspberry Pi Versus Arduino

In this lesson, I will discuss the 10 most important differences (and similarities) between the Raspberry Pi and the Arduino.



You can watch the video, or, if you are the “reading” type, you can read the text below.

Raspberry Pi vs Arduino

Raspberry Pi	Arduino
Microcomputer	Microcontroller
Needs an operating system	Does not need an operating system
Complicated	Simple
Video out, Camera, Ethernet ports, Wifi, Bluetooth, USB, I2C, SPI, UART etc. on board	USB only for power and serial in/out, I2C, SPI, UART
Best for general computer	Best for small tasks that constantly repeat
Capable of performing a huge range of tasks	Optimised for sensing and controlling the world around it
Best for more advanced makers	Best for beginners
Programmed in many languages, including C/C++, Python, Ruby	Programmed in C/C++
Relatively high power consumption	Relatively low power consumption

Raspberry Pi Full Stack 

A comparison between the Raspberry Pi and the Arduino.

Hardware

To begin with, let's consider the fact that the Raspberry Pi is a normal computer. It just comes in a tiny package. But, it's got everything that you'd expect a computer to have. Think about it.

- You can connect it to a screen.
- You can connect it to a keyboard, a mouse.
- You can connect it to external hard disks.
- You can connect it to networks, including Wifi (Raspberry Pi 3 and newer boards).
- You can connect it to Bluetooth devices (Raspberry Pi 3 and newer boards).

The Raspberry Pi 3 has all the features that you expect to have on a computer. And it's also got respectable specifications. The Raspbian Pi 3, for example, has a 1.4GHz processor, with four processing cores, and a full gigabyte of RAM. It can run Linux and even Windows! On the Arduino side: The Arduino is not a computer, as per the general use of the term. It's a

microcontroller. Think of a microcontroller as a tiny, tiny, computer without any of the things that we expect in a desktop or a laptop computer.

The Arduino Uno, for example, has got very little RAM. Just 2 kilobytes of RAM. That is just 0.2% of the available RAM of the Raspberry Pi 3! It can't connect to a network like the Internet without the additional components. It doesn't have much storage for your programs, just The Arduino Uno only has 32 kilobytes of flash memory for such purpose, where the Raspberry Pi can use gigabytes and gigabytes of SD card to provide memory for that purpose.

Performance

The next big difference is performance. A tiny computer with a 1.4 gigahertz CPU clock rate, four processing cores, a dedicated HD graphic processor, a gigabyte of memory and lots of gigabytes of disk space, you have serious resources at your disposal.

You can use these resources to do very interesting things. You can run an operating system like Linux or Windows. You can also use it to play games or to write large applications and even run web servers. None of that is possible with the Arduino, which is limited to what its microcontroller can do. Right from the bat, you've got two devices that are totally different. That's because when they were designed to be different.

A microcontroller is designed to predictable, small, and endlessly repeatable tasks, and to do them efficiently and for a very long time without supervision.

For example, if you want a gadget that measures the temperature and the humidity in a room, and then, depending on those specific two values to turn on a fan or an air conditioner, then something like the Arduino (or a microcontroller in general) will be the best technology on

which you can build such a gadget. But, if you want something that can do all of the above, plus a few more things such as to control a security system while performing face recognition from a live video stream, while keeping a log of its activities in a database, then you need a computer, like the Raspberry Pi. The second example involves the performance of many more tasks that need to be individually programmed, supported, and provisioned for. The Arduino would be totally inadequate for something like that. For that, you'd be looking at the Raspberry Pi. So the Raspberry Pi is great for general computer stuff, whereas the Arduino is great for things that are small in complexity, small in programming size, and repeat constantly. A great thing about the Arduino is that because of all its input and output pins and ability to connect them to actuators and sensors, things such as relays, switches and motors, it is really a good choice, for sensing and controlling the world around it. As long as this level of sensing and controlling doesn't become too complicated. The Raspberry Pi, on the other hand, can do a lot of what the Arduino can, plus a lot more that the Arduino definitely cannot do. The Raspberry Pi is capable of performing a huge range of tasks. Just like the Arduino, the Raspberry Pi does have a bunch of general purpose input output pins that you can connect to actuators and sensors.

Hardware-wise, the Raspberry Pi is more complicated also because of the amount of built-in capability that comes on board.

Where the Arduino shines

There is definitely a good case to be put forward that says that the Raspberry Pi can actually do everything that the Arduino can, therefore you don't really need the Arduino. But there is a flip side to that. The great thing about the Arduino, and that's why I love it personally, is that it's simple. The fact that it has no operating system to worry about means that you don't have to gain additional skills, just to interact with basic hardware like LED and motors. To acquire such you would need

to invest a significant amount of time. It's a steep learning effort to learn how to use an operating system in order to just be able to turn the on and off or an external appliance on and off via a very simple relay wiring. If learning an OS is one of your goals, than this is not a problem, but if not, it will hinder and slow your progress. For beginners, I always say that the best way to get started with electronics and with control applications is through the Arduino. The Arduino can help you to get the understanding, the capability and the skill to using the bare minimal infrastructure that a microcontroller provides before moving onto something more complicated. And as a matter of fact, if you think about, it when a new person comes into the world of electronics and control their objectives tend to be fairly simple: just blinking an LCD very often is a great achievement and it is really an achievement when it is something that you've done for the first time, and then spinning a motor. It's these simple things that you can do really quickly with the Arduino that make it the ideal learning platform for new makers. Simplicity is where the Arduino wins big time.

Programming

In terms of programming languages, the Arduino is programmed in C/C++. The Arduino team has built its so-called "Arduino language", based on C and C++. This is another great advantage of the Arduino. C/C++ on its own is a fairly complicated language. C/C++ has a steep learning curve. But thanks to the simplification that the Arduino layer on top of C/C++ provides, it is relatively easy to learn the basics of, and quick to be able to be productive with the Arduino. On the other hand the Raspberry Pi, being a full microcomputer, has a wide range of languages that you can use to program it. Almost any language that you can use in Linux is also available for the Raspberry Pi. By default, the Raspberry Pi, when using Raspbian, it's most commonly used operating system, comes with support for C and C++ and Python. You can also install other languages, like Ruby, PHP and many other things. So, make your choice. If you're familiar with in any programming

language, chances are that you'll be able to use it with the Raspberry Pi. Most of the Raspberry Pi documentation especially, for educational purposes, is written in Python, which is a high level language.

Python has a very short learning curve, so even if you've never done Python the past you'll be able to pick it fairly quickly, at least when you compare it to other languages like C and C++.

Power consumption

Finally, let's look at power consumption. As a computer with a lot more resources and hardware to power, the Raspberry Pi requires a lot more current and power compared to the Arduino. You need a 1 A power supply (at a minimum) to power your Raspberry Pi reliably, and more than that if you have things such as touch screens and cameras connected to it. The Arduino, on the other hand, is very nimble. You can power it with just 20 mA, for example. Of course, these numbers depends on what else you have connected to your boards. Another power feature of the Arduino and the microcontroller technology is that you can programmatically control its power consumption and reduce or increase it on schedule as needed. You can put your Arduino to sleep, for a few hours, days, and wake up a while later to perform a task. Then, it can go back to sleep again. It's possible to power an Arduino using small lithium batteries, for a year, operating in a low power mode. This is a very useful capability when you want to deploy an Arduino gadget at locations where fixed power is not available.

5. Setup Raspbian

RASPBERRY PI GETTING STARTED GUIDE SERIES

How To Setup Raspbian Lite

In this lesson, I will walk you through the full process of installing the latest Raspbian Stretch operating system on an SD card. In this guide, and in the [Raspberry Pi Full Stack](#) course, we use the minimal version of Raspbian.



You Raspberry Pi will not be able to do anything without an operating system. This is very different to how a microcontroller, like the Arduino, works. A microcontroller does not need an operating system. The Raspberry Pi, though, does, as it really is a full computer.

In this lesson, I will walk you through the full process of installing the latest Raspbian Stretch operating system on an SD card. In this guide, and in the [Raspberry Pi Full Stack course](#), we use the minimal version of Raspbian. This means that we will not be needed any of the graphical user interface elements of the operating system, so the operating system will need a much smaller SD card.

I describe the process in this video (or, if you prefer to read, do so below):

Needed hardware

To follow along with the instructions in the video, you will need a Class 10 SD card with at least 8GB available space. You will also need a computer running Windows or Mac OS, with an SD card slot or an external SD card reader/writer device.

Needed software

You will need to download the Raspbian Lite version of the operating system, and a small program needed to “burn” (i.e. copy) this OS on your SD card.

To download the Raspbian OS image, go to [the Raspberry Pi Foundation download page](#). Download the Zip version of the option titled “Raspbian Stretch Lite”.

The term “image” refers to an archive of the contents of a disk, or in our case, an SD card. It usually is distributed compressed as a ZIP file. When you extract the contents of the ZIP file, you have the disk image.

To burn the image on the SD Card, I recommend a small free program called Etcher. [You can download Etcher from its website](#). Install it on your computer as you do with any other program.

You now have everything you need to proceed with the setup.

Download the operating system image file

To do that, we'll go in to raspberrypi.org/downloads and then click onto the Raspbian operating system logo. That will take you to the page where you have the option to downloading either the desktop version, the full version of Raspbian with the GUI elements or the light version. At the time writing this, the latest Raspbian operating system version is called "Stretch". Go ahead and click on the download button to download it. It's a process that takes a few minutes to a few hours to complete, depending on the speed of your internet at your location. The zip file that you have downloaded is an archive that contains the compressed version of the image that we want to put onto the SD card. First you will need to expand it, as you do with any other ZIP file.

Once the ZIP file is expanded is completed you'll have an image file that is around 1.6 GBytes in size. Look for a file with the "IMG" extension. This is the image file that we want to copy onto the SD card.

Transfer the image to your SD card

When it comes to transferring this image file onto the SD card there's a few different ways of which you can do that. Different utilities allow you to do this sort of thing or you can also do it on the command line. But it seems like the easiest way to go about this is to use the Etcher application. The nice thing about this application is that it's cross-platform compatible so you can download a version for the Mac, for Windows, or for Linux,

and it works exactly the same way on all three operating systems. [Download the version of Etcher for your operating system](#). Install and then start the application. The Raspberry Pi requires a mini SD card that goes into the connector at the bottom of the board. In my setup, I'll use an SD card that is 16 gigabytes in size. I've tested the process with an 8 gigabyte SD card on a Raspberry Pi 2 and it worked with no problem at all. I am going to have a lot of empty space with this card. After you have started Etcher, select the image that you want to transfer to the SD card. Look in your download folder, for the file with the IMG extension, then click on "Open". After that, you will want to confirm the target. Etcher should be able to automatically detected the target SD card. If not, select it manually. A nice feature of Etcher is that it will only show you the SD card drives on your computer. It isn't going to show you any other hard disks or network drives that very often confuse people. It will only show you the actual SD cards, so it makes it very safe to use Etcher as opposed to other SD card utilities. Accept the correct target SD card with click on Continue, and then click on the Flash button. Etcher will probably ask for your password at this point. Provide it, and click Ok to continue. Then the process of copying the image onto the SD card will begin and take about a minute. Etcher will show you the progress. Etcher will flash that image onto the SD card and then verify it as well to make sure that the copying was correct. When the process is completed you can now close Etcher and remove the SD card from your computer.

There's a couple more things to do before you actually plug the SD card into your Raspberry Pi for the first time. Because we are going to be using the Raspberry Pi in headless mode, which means no monitor and no keyboard attached to it, we need a way to be able to remotely access the Raspberry Pi in order to be able to find its IP address. If you're using a Raspberry Pi with integrated Wi-Fi, such as the Raspberry Pi 3 or the Raspberry Pi Zero W, then you also like to be able to setup Wi-Fi before you actually boot it for the first time. Before you go ahead to insert the SD card in the Raspberry Pi and boot for the first time, you should setup SSH and Wifi. You can learn how to do this in the **next lesson** in this guide.

4: Headless and graphical (GUI) operating systems

RASPBERRY PI GETTING STARTED GUIDE SERIES

What Is A 'Headless' Operating System?

In this project you will be using Raspbian Lite which, as you know, is a Linux-based operating system with no graphical user interface and only the bare-minimum software pre-installed. In addition, you will setup your Raspberry Pi Zero W without a monitor, keyboard or mouse.



The only cable you will connect to it is the mini-USB power cable.

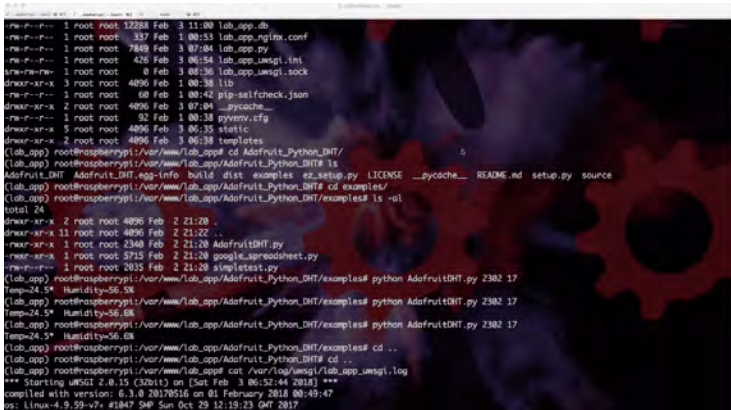
This type of setup is known as “headless”.

A headless computer is one that is configured without the typical input and output interfaces (monitor, keyboard, mouse). A user will interact with a headless computer through a network interface and a separate, client computer.

In this project, you will setup your regular work computer with a terminal program, like Putty, iTerm or the Windows Console and the SSH (Secure SHell) protocol to connect to your Raspberry Pi via your Wifi network.

You will issue text commands on the command line, as in the

example in the screenshot below.



There are a few more differences between a headless computer system and a GUI-based one that I want to highlight.

I have summarised these differences in this Table :

Headless	GUI
No monitor, mouse, keyboard.	Requires monitor, mouse, keyboard.
OS occupies much less disk space.	OS is larger to accommodate for the graphical components and applications.
Operated using a separate computer via a remote connection.	Can be operated directly, no need for separate computer.
Best for server or remote applications.	Best for desktop applications.
Faster to backup and update (due to its smaller footprint).	Longer to backup and update (due to its larger footprint).
Improved reliability (fewer things to break).	Acceptable reliability for desktop use.
Improved performance that is memory-bound, RAM is dedicated to applications.	RAM has to be shared with video and applications.

Headless

User must be familiar with the command line.

GUI

User can use point and click interface.

Apart from those differences I have already mentioned, the table a few additional yet just as important ones.

A headless operating system without GUI components and applications has a very small footprint on the disk. This means that maintenance and backup/restore operations are faster simply because there are fewer bytes to work with. In this project I will show you how to backup and restore your work stored on the SD Card. Backing up an 8GByte SD Card (which can easily accommodate Raspbian Lite) take half the amount of time that you need for the same thing when you use a 16GByte SD Card (which is the smallest recommended size for the full Raspbian).

Another advantage is reliability. Headless computers typically run minimal operating systems so that the risk software bugs and conflicts between services is minimised. Those computers can run for years without a reboot, reliably servicing end user requests. This is impossible to ask from a full desktop operating system. Even the best of those must be restarted regularly. In this project, you will create an application that operates as an automated service, reliably, non-stop, for ever.

My Raspberry Pi 1 is still running the original Full Stack application and Raspbian Wheezy (the first version of Raspbian from 2013) since 2014. I admit that I had to restart it a few times due to power outages. The longest uninterrupted stretch was 9 months, which is amazing.

Another advantage of headless computers and Lite operating systems is performance. A regular GUI operating systems contains hundreds of background services that are constantly running. These services do things such as checking for mouse of keyboard events, redraw the screen and handle user notifications.

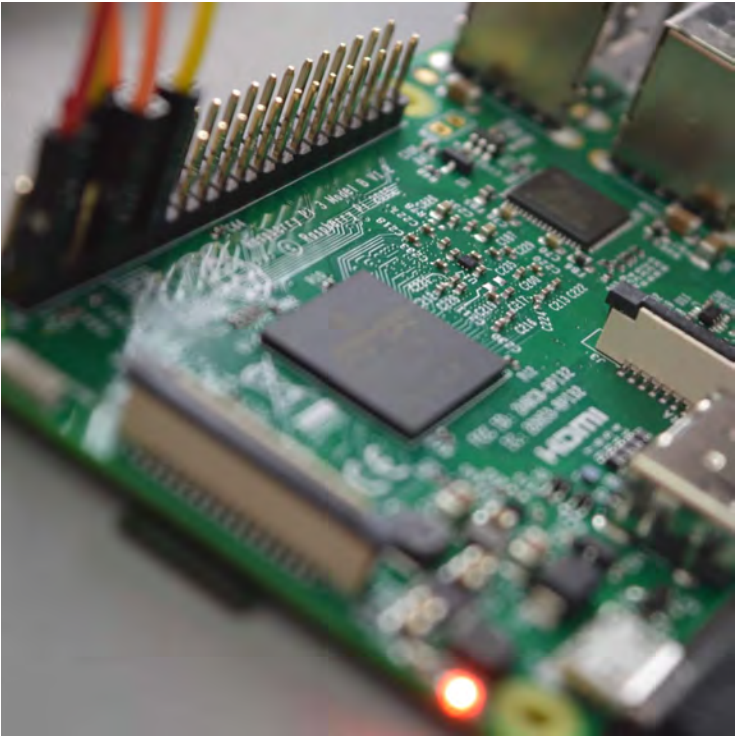
All these things take up valuable CPU and memory resources. In a computer like a modern PC with multiple CPU cores and lots of RAM, this is not a big consideration. After all, these computers are designed for GUI interfaces from the ground up. However, for a Raspberry Pi Zero, with a single processing core and 512MB RAM, useless services quickly become a performance bottleneck. A Lite and headless operation released resources that we can better use to develop and operate our web application.

3: Raspberry Pi operating systems

RASPBERRY PI GETTING STARTED GUIDE SERIES

Raspberry Pi Operating Systems

The Raspberry Pi requires an operating system to be able to do anything useful with it. After all they're Raspberry Pi is a computer, is just miniaturized, a very small computer. So the operating system on any computer is what makes it possible for our applications to utilize the hardware of the computer itself.



In this lesson, you will learn about some of the operating systems that you can use in the Raspberry Pi.

You can watch the video, or, if you are the “reading” type, you can read the text below.

What is the “operating system”?

As the end user of a computer, we use a program, like a web browser or a word processor to carry out some work. If we happen to have programming skills, we can write such programs ourselves.

In the early days of computing, our programs would interact directly with the hardware. For example, to do a simple

calculation, we would write instructions that would store values in specific memory locations, and use other instructions to trigger arithmetic calculations between the values that we previously stored in these specific memory locations.

As the computer hardware and programs became more complex, this direct approach became impossible to handle. Computer scientists and engineers created an abstraction layer. A software layer above the hardware that allows programmers to write programs more efficiently, without having to worry about the details of the hardware.

The abstraction layer is the operating system and the compiler software that we use to “translate” a program written in a high-level language (like C or Python) into something that the computer can understand natively.

In fact, specific types of computers, like microcontrollers, just like the very first computers, don’t use an operating system, though they do use a compiler. While our programs run directly on the hardware, we can still handle the complexity because the microcontroller hardware is simple compared to that of a computer, and also because of the programming tools (the IDE and the compiler, in particular).

With a modern computer, from a high-level perspective, this is what happens:

1. We write a program in a language like C or Python.
2. We compile the program into a language that the operating system can understand.
3. Our compiled program interacts with the operating system
4. The operating system executes our compiled program and gives access to computer resources such as the file system, the GPIOs,

the screen, etc.

Without an operating system the Raspberry Pi is just paperweight. Over the last few years, as the Raspberry Pi went through its iterations from the original Raspberry Pi 1 to Raspberry Pi 2 to the current version Raspberry Pi 3 and it became more powerful, more and more operating systems started becoming available for the Raspberry Pi. When you go to the [Raspberry Pi downloads page](#), you'll see that there are many operating systems available. There are also many third party websites from where you can download many more customised and specialist operating systems for the Raspberry Pi. You can find one of many growing lists [here](#).

Raspbian

The operating system that the Raspberry Pi Foundation supports and keeps current and looks after is [Raspbian](#). There's also a [version of the Raspberry Pi operating system](#) that you can install on your computer, a Mac or PC computer. With this version of Raspbian, you can get the same educational applications programming languages and so on that are available on Raspbian for the Raspberry Pi, but for your computer; and you don't even need a Raspberry Pi!

NOOBS

I also want to mention NOOBS. NOOBS is an operating system installer for the Raspberry Pi. It is designed for people that are absolutely new to the Raspberry Pi and don't want to deal with the nuances of actually "burning" an operating system image on an SD card. The process of "burning" the OS on an SD card is straight forward, as you will learn later in this series of articles, but it can still be overwhelming for many people.

The NOOBS operating system is typically delivered on an SD card when you purchase a new Raspberry Pi. Or, you can download it and "burn" it to an SD card (more about this later).

With NOOBS, the user only needs to connect their Raspberry Pi to a screen and a keyboard, and power it up. NOOBS will boot. Through NOOBS, you can choose which of several operating systems you would like to use, and then NOOBS will go fetch the necessary files and install that operating system on the same SD card. Through NOOBS, you can avoid having to burn the image of your selected operating system on the SD card and then to configure it. Of course, this is useful if you have NOOBS installed on the SD card that comes with a newly purchased Raspberry Pi. If you don't have it then you can still download it from here and burn it yourself on an SD card but that defeats the purpose. In that case, I think that you are better off to just go and install Raspbian which is what you will learn how to do later in this series.

Other operating systems

Apart from Raspbian, there are other operating systems available.



Some of the operating systems available for the Raspberry Pi.

Examples of alternative OSs include Ubuntu MATE, Snappy Ubuntu Core, and Windows 10 IoT Core.

Ubuntu MATE, is a flavor of the actual full Ubuntu MATE operating system which is very popular among Linux users. This makes the exact same setup of a desktop Linux computer available on your Raspberry Pi 3. Ubuntu MATE is a full operating system in terms of the computing resources it needs, so you will need at least a Raspberry Pi 3 whereas. On the other hand, the Raspbian operating system can run it on earlier Raspberry Pis, like the Raspberry Pi 2. All of these operating systems emphasize a particular attribute and use. For example, the Windows 10 IoT Core OS is optimized around the use of the Microsoft Windows IoT infrastructure from your Raspberry Pi. Depending on what your objectives

are, you can choose an operating system to match those objectives.

Minibian

In the first version of [Raspberry Pi Full Stack](#) that I released in 2015, I used [Minibian](#) instead of Raspbian. I did this because, back then, there was no “lite” version of Raspbian (hold on to that thought, I’m going to explain what the “lite” version is). At that time, I decided to use Minibian because it was the smallest possible version of Raspbian that I could find. Raspbian Lite did not, yet, exist. Because I wanted to build a minimal web application stack on my Raspberry Pi, I was looking for the smallest possible footprint version of Raspbian, i.e. an operating system that could fit in a small capacity SD card. And Minibian was that version.

What is a “minimal OS”?

A minimal operating system is an operating system that contains only the absolutely necessary components to achieve a specific mission.

In the case of Raspbian, the full version comes with a lot of components that are not really necessary when we just want to build a web application. For example, features like the graphical user interface, email and browser applications, word processors and games are useless if you want to setup a web server. By using the full version of the operating system we are wasting a lot of space that can be better used for things such as logging and database files. Minibian threw away all of the components in Raspbian that were not useful, in particular the graphical user interface, and much more, and only kept the bare minimal. From the bare minimal end users are free to build it up by installing the absolutely necessary components for their web application and the web application stack, and nothing else.

So that's why back in 2015 I decided to use Minibian.

Raspbian Lite

Fast forward a few years and the Raspberry Pi Foundation finally releases a minimal version of Raspbian: Raspbian Lite.



The screenshot displays two download options for Raspbian Stretch. Each option includes a Raspberry Pi logo icon, a title, a description, a table of metadata, and two download buttons (Torrent and ZIP). The first option is for the desktop version, and the second is for the minimal Lite version. Both versions were released in April 2019.

Version:	April 2019
Release date:	2019-04-08
Kernel version:	4.14
Release notes:	Link

SHA-256:
7e10a446f8e57210d0e9ad02f0c833aabb86e58187b4dc02431aff5a3f1c
cb83

Version:	April 2019
Release date:	2019-04-08
Kernel version:	4.14
Release notes:	Link

SHA-256:
03ec326d45c6eb6cef848cf9a1d6c7315a9410b49a276a6b28e67a40b11f
dfcf

Raspbian and Raspbian Lite

Raspbian Lite is the equivalent of Minibian from a few years ago. The difference between Raspbian Stretch Lite and

Raspbian Stretch Desktop is that the Lite version doesn't have any of the applications and the graphical user interface component that are unnecessary for an operating system that is meant to be executed and to be used on the desktop. All of those components are thrown out, and only the absolutely necessary is kept. And on that we can build with a components that we need for a web application. So that is why for this guide and course I am using [Raspbian Stretch Lite](#). We still get Debian Linux under the hood with all the necessary infrastructure that will allow us to build on.

6: How to setup SSH and Wi-Fi in headless mode

RASPBERRY PI GETTING STARTED SERIES

How To Setup SSH And Wifi In Headless Mode

In this lesson you will learn how to setup SSH using Wifi, SSH, and a remote computer connected to your Raspberry Pi in headless mode.



Now that you have Raspbian Lite on your SD Card, its time to do some simple configuration. The purpose of this configuration is:

- Enable the SSH daemon so that you can access your Raspberry Pi via the SSH protocol and your terminal emulator.
- Setup Wifi so that your Raspberry Pi can connect automatically to your local network once you start it for the first time.

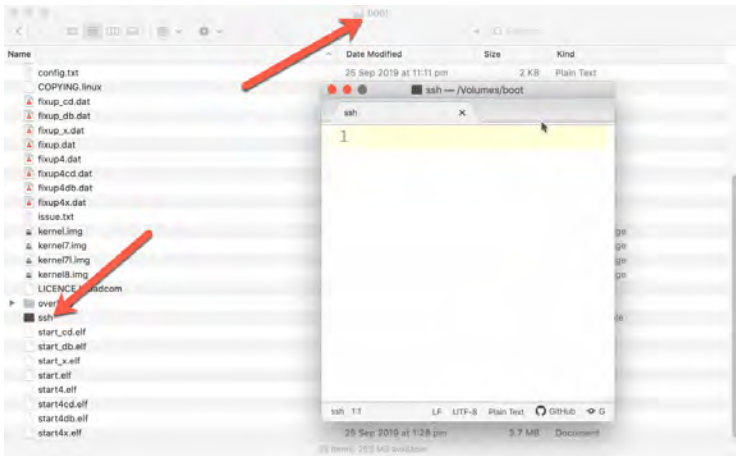
The second item is very important because the Raspberry Pi Zero W does not have an Ethernet network interface. This means that you will not be able to access it without Wifi.

Also, because you are working in headless mode, you will not b/e able to plug in a monitor and a keyboard to setup Wifi directly.

Start by inserting your Raspberry Pi SD card into your computer.

Use your computer's file browser (Finder in Mac OS, Windows Explorer in Windows) to navigate to the disk with name "boot". The Raspberry Pi looks inside this partition when it boots for special files that contain instructions like the ones we are about to create.

First, lets enable SSH. Use a text editor and create a new empty file. Save it to the boot disk with the file name "ssh". No extension, and no content. Just an empty file named "ssh", as in the screenshot below.



Create an empty text file titled “ssh” in the boot disk to enable SSH.

That’s it.

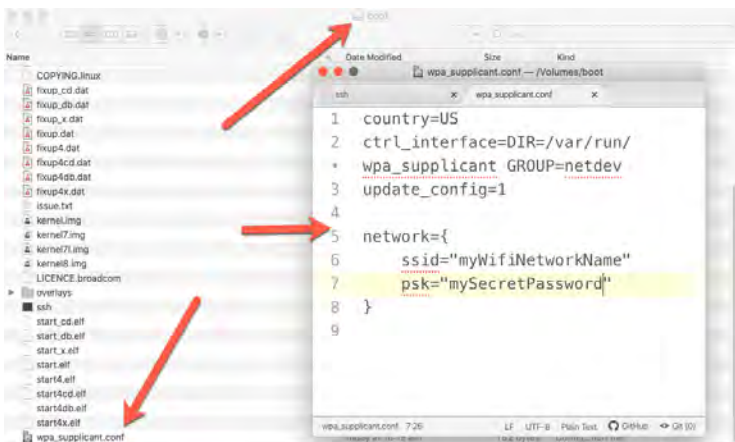
Let’s continue with the Wifi configuration.

Go to the project repository on Github, and grab a copy of the wpa_supplicant.conf.buster file. It looks like this:

```
country=USctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdevupdate_config=1network={ ssid=""—your-SSID—
psk=""—your-wifi-passkey—}
```

Use your favourite text editor to create a new text file, and copy the content from the sample supplicant file in it. Take care to type in the correct network name and password in the “ssid” and “psk” fields.

Then, save the file to the “boot” disk with the name “wpa_supplicant.conf”:



The wpa-supplciant.conf file in the boot disk configures the Wifi interface before the first boot.

When you have both the “ssh” and “wpa_supplciant.conf” files saved in the boot disk, you can eject it from your computer.

In the next lesson, you will use it to boot your Raspberry Pi for the first time.

7: How to set a hostname

RASPBERRY PI GETTING STARTED SERIES

How To Set A Hostname

In this lesson I will show you how to set the hostname of your Raspberry Pi, and give it a fixed IP address. Although these two configurations are not essential, they will save you time over the long term.



Your SD Card with Raspbian Lite is ready.

You've enabled SSH and configured the Wifi settings.

You are eager to use it and start your Raspberry Pi for the first time.

But, wait, there is one more thing you should do now, before you the first boot.

In this lesson I will show you how to define the hostname of your Raspberry Pi, and give it a fixed IP address. Although these two configurations are not essential, they will save you time over the long term.

By giving your Raspberry Pi a hostname, you will be able to access it with an easy to remember name, instead of an IP address.

For example, I can access the home page of my application in my browser with this:

```
http://raspberrypi-zero.local/lab_env_db
```

Instead of this:

```
http://192.168.1.45/lab_env_db
```

Much better with the hostname, right?

Once you have a hostname, you will be able to identify your Raspberry Pi in your router by name, instead of the IP address. This way, you will be able to use your router's administration tools to assign it with a fixed IP address. This is not critical, however it is good practice to give computers that offer services to the network (such as file, web, or printing services) a fixed/permanent IP address.

You can certainly skip this step and set your hostname after you boot your Raspberry Pi for the first time. But, because there is no monitor connected to it, you will need to use a network scanner application or information from your router to find out your Raspberry Pi's assigned IP address. Once you know the IP address, you will be able to connect to it via SSH

and edit the host name. This is something I will show you how to do very soon.

But here, it is worth looking into the option of setting the hostname before the first boot so that you sort out the hostname early in the process (and get the easy access benefits). At the same time, you will learn about the internal structure of your Raspberry Pi's SD card.

Let's start.

At the time I am writing this, there is no reliable way to set the hostname of a Raspberry Pi computer running Raspbian by setting a configuration file in the root partition. I am aware of efforts by members of the community to create a solution for this based on a script, however this solution is not reliable at the moment.

Ideally, the Raspberry Pi Foundation will provide a solution similar to the SSH and wpa_supplicant method that you learned about in the previous chapters.

But, it is still possible to set a hostname before we boot the Raspberry Pi for the first time.

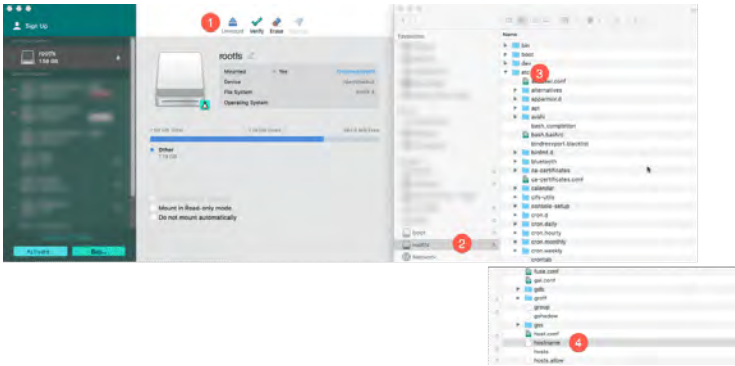
The Raspbian SD Card contains two partitions. You already know about the "boot" partition. You modified the contents of this partition in previous chapters to enable the SSH daemon and set the Wifi configuration.

The second partition is named "rootfs". This partition is formatted with the extFS4 filesystem. If you are working on a Linux computer, then you will be able to access this partition natively, like any other disk. However, if you are using a Mac OS computer (as I do), or a Windows computer, you will need to install an extFS4 driver first.

I use a utility called "[extFS for Mac](#)". There is a version for [Windows](#). This utility allows you to mount an extFS3 or extFS4 disk in your file system, and use it as you would with a native

disk. An alternative to the paid solution from Paragon is the [MacFUSE project](#) for the Mac OS, or the Ext2Fsd Project for [Windows](#). The Ext2Fsd Project seems to not be active though, with the latest release being in November of 2017.

Install one of those solutions to enable your computer to work with extFS4 drives, and then insert your Raspbian SD Card into your computer. Then, select the rootfs drive partition. In extFS For Mac, it looks like the example in the screenshot below.



To set the hostname, edit the “hostname” file under rootfs/etc.

Once you mount the rootfs partition on your computer, browse to rootfs/etc/.

Then, open the “hostname” file in a text editor, and type in the name that you’d like to refer to your Raspberry Pi by.

My Raspberry Pi hostname is:

raspberrypi-zero.local

Save this text file with the single line, close the text editor and eject the rootfs and boot partitions from your computer.

You are now ready to boot your Raspberry Pi for the first time, and access it with its new hostname. Let’s do that next.

8: Boot into Raspbian for the first time

RASPBERRY PI GETTING STARTED SERIES

Boot Into Raspbian For The First Time

In this lesson you will boot up your Raspberry Pi for the first time, and connect to it using SSH.



If you have set its hostname, you will use that hostname to access the Raspberry Pi.

If you have not, no problem: I will show you how to determine the IP address that your router has assigned to your Raspberry Pi, and use that IP address to access it (again, via SSH).

I will be using a Raspberry Pi Zero W.

To boot your Raspberry Pi Zero W and operate it, apart from the Raspbian SD Card, you will need a USB power supply and a USB cable with a Micro Type B plug at one end, and the regular USB Type A plug at the other end.



To operate your Raspberry Pi Zero W, apart from the Raspbian SD Card, you will need a USB power supply and a cable.

Insert the Raspbian SD card into the SD Card slot of the Raspberry Pi, and plug the USB power supply into the mains power socket. Then, plug the Micro USB connector into the USB port that is marked “PWR IN”.

In the photograph below, you can see my Raspberry Pi Zero W, powered by USB. The tiny green LED is showing disk activity by blinking irregularly as the computer is booting.



When you apply power to your Raspberry Pi, the green LED will blink irregularly to indicate disk activity.

Allow at least two minutes for your Raspberry Pi to boot and connect to your Wifi network. You will know that it is ready when the disk activity indicator (the green LED) stays on.

If you have set a hostname in the previous chapter, now is time to use it. Start your terminal emulator (such as iTerm or Terminal on Mac OS, and Putty or PowerShell on Windows), and issue this command:

```
$ ssh pi@raspberrypi-zero.local
```

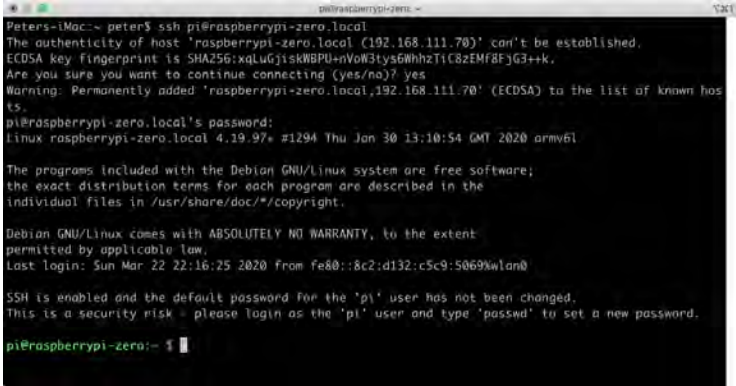
Of course, replace “raspberrypi-zero.local” with the host name of your Raspberry Pi. The default password for user “pi” is “raspberry”.

Let’s break down this instruction:

1. “ssh” is the protocol you are using for the communication session.
2. “pi” is the user name for the account you are logging into.
3. “raspberrypi-zero.local” is the hostname of

your Raspberry Pi.

In the screenshot below you can see an example of what to expect when you issue this command.



```
Peters-iMac:~ peter$ ssh pi@raspberrypi-zero.local
The authenticity of host 'raspberrypi-zero.local (192.168.111.70)' can't be established.
ECDSA key fingerprint is SHA256:xlLuGjiskWBPU+nVoW3tys6Whh2TlC8zEMF8FjG3++k.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'raspberrypi-zero.local,192.168.111.70' (ECDSA) to the list of known hosts.
pi@raspberrypi-zero.local's password:
Linux raspberrypi-zero.local 4.19.97* #1294 Thu Jan 30 13:10:54 GMT 2020 armv6l

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: Sun Mar 22 22:16:25 2020 from fe80::8c2:d132:c5c9:5069xwlan0

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

pi@raspberrypi-zero:~$
```

Connect to the Raspberry Pi using SSH and a terminal emulator.

Your SSH client will ask you to confirm that you want to connect to the new host because it has never seen its “fingerprint” before. Answer “yes”. Your computer will store your Raspberry Pi’s fingerprint and will not ask you again.

After this, SSH will ask you for the password for user “pi”. Type “raspberrypi”, and hit enter. You’re in!

Before doing anything else, I’d like to show you how to connect to your Raspberry Pi in case you have not set a hostname. In this case, you will need to find out the IP address that your router assigned to your Raspberry Pi.

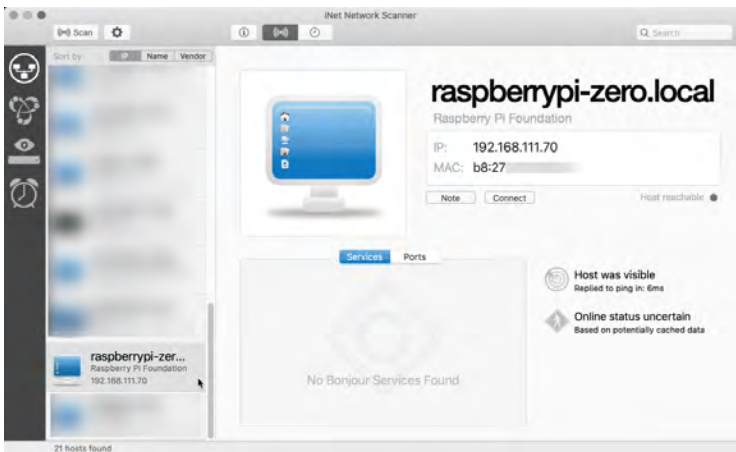
There are two ways to do this:

- With the help of a network scanner
- By looking into your routers DHCP page

Method 1: network scanner

You can use a network scanner like “[iNet Network Scanner](#)” for Mac OS and the [Advanced IP Scanner for Windows](#). Either utility will scan your network and make a list of all the hosts that it can find. One of those hosts will be your Raspberry Pi, and the scanner will give you its IP address.

In the screenshot below, you can see that my network scanner has found my Raspberry Pi, and is showing relevant information such as its hostname (which I set earlier), its IP address, and its MAC address.



My network scanner has found my Raspberry Pi.

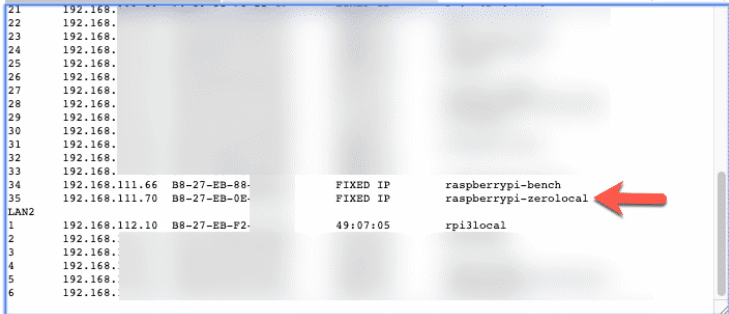
As you can see, the assigned IP address of my Raspberry Pi is 192.168.111.70. You can use the IP address instead of the hostname to connect via SSH:

```
$ ssh pi@192.168.111.70
```

Method 2: Router DHCP table

An alternative to the network scanner is to look for the same information in your router's DHCP table. You will need to login to your router's admin panel, and look for the DHCP page, usually available under a diagnostics menu item. In the screenshot below you can see the DHCP table of my router, which gives the IP address of my Raspberry Pi.

Diagnostics >> View DHCP Assigned IP Addresses



The screenshot shows a router's DHCP table with the following data:

DHCP IP Assignment Table		Other IP Assignment Table		Refresh
21	192.168.			
22	192.168.			
23	192.168.			
24	192.168.			
25	192.168.			
26	192.168.			
27	192.168.			
28	192.168.			
29	192.168.			
30	192.168.			
31	192.168.			
32	192.168.			
33	192.168.			
34	192.168.111.66	B8-27-EB-88-	FIXED IP	raspberrypi-bench
35	192.168.111.70	B8-27-EB-0E-	FIXED IP	raspberrypi-zero
LAN2				
1	192.168.112.10	B8-27-EB-F2-	49:07:05	rpi3local
2	192.168.			
3	192.168.			
4	192.168.			
5	192.168.			
6	192.168.			

My router's DHCP table show the IP address of my Raspberry Pi.

Use one of these methods to find our your Raspberry Pi IP address. Without this, or the hostname, you will not be able to connect to your Raspberry Pi, and therefore you will not be able to do the following work.

In the next lesson, I will show you how to set a fixed IP address for your Raspberry Pi, which is a good practice for any computer that offers services (such as a web server) to the network.

9: How to set a fixed IP address

RASPBERRY PI GETTING STARTED SERIES

How To Set A Fixed IP Address

In this lesson you will learn how to set a fixed IP address for your Raspberry Pi.

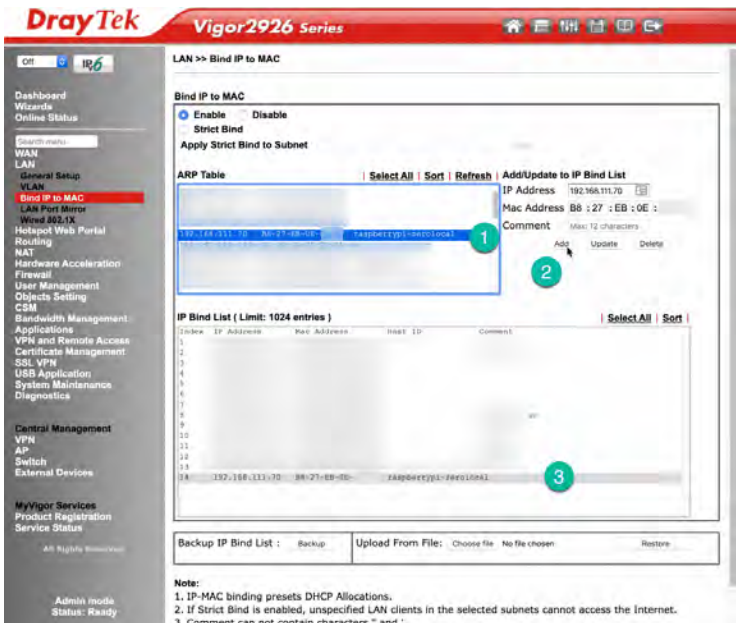


In addition to setting a hostname, it is good practice to set a fixed IP address to network hosts that provide services to other

hosts. This way, a client will be able to use the same IP address for all its requests to the server, instead of first doing a lookup request to the DHCP server.

To set a fixed IP address for your Raspberry Pi, you will need to login to your router's admin panel. Every router has an admin panel with its own "branded" design elements, but in general there is a page titled "DHCP" or "Bind IP" that allows you to edit the configuration of a host's IP address.

In the screenshot below you can see the relevant page in my router's administration panel. This page is available under the LAN menu item.



My router allows me to bind an IP address to a MAC address.

To bind an IP address to a MAC address, first select the Raspberry Pi by identifying its hostname from the ARP Table ("1"), then click on the "Add" button ("2"), you may change the IP address to something else, or accept the one that DHCP

has already assigned).

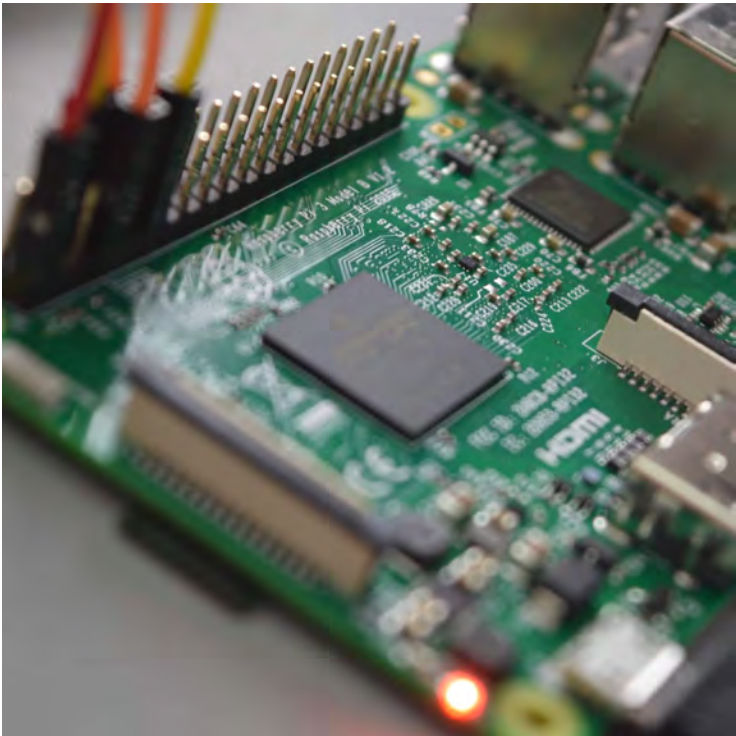
In the IP Bind List box, you can now see the fixed IP address for your Raspberry Pi ("3").

10: Basic configuration

RASPBERRY PI GETTING STARTED SERIES

Basic Configuration

In this lesson you will learn how to setup basic configurations on the Raspberry Pi.



All the network-related configurations are complete, so now you can go ahead with some basic and essential configurations on the Raspberry Pi itself.

Start by logging in:

```
$ ssh pi@raspberrypi-zero.local
```

```
$ ssh pi@192.168.111.70
```

```
$ ssh pi@192.168.111.70
```

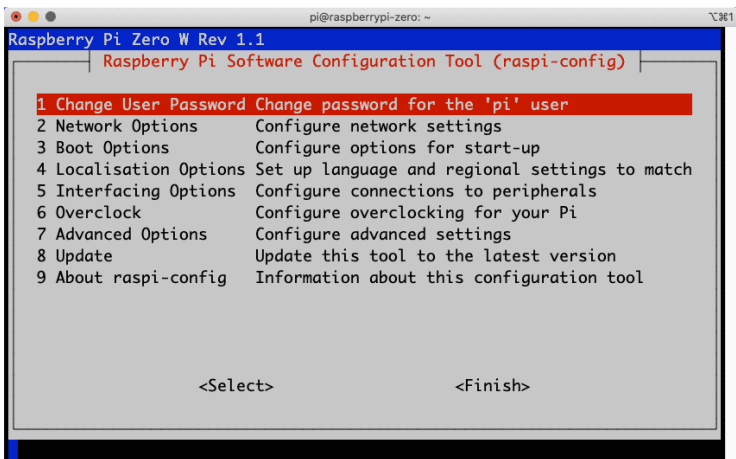
Raspbian comes with a configuration utility called “raspi-config”.

This utility allows you to easily turn on or off components like the SPI and I2C interfaces, set how much RAM to dedicate to video, and even to turn on overclocking so that your Raspberry Pi can operate faster (I do not recommend doing this).

Start raspi-config like this:

```
$ sudo raspi-config
```

You will see the utility home page, from where you can access the subcategories:



The home screen of the raspi-config utility.

You can navigate the utility pages with the arrow keys (up, down) and the TAB (change to next button) and Return (accept)

selection) keys.

These are the changes that you want to make (remember that some options, such as SSH and the hostname, are already enabled):

- Under Interfacing Options:
 - SPI: Enable
- Under Advanced Options:
 - Expand Filesystem: this will ensure that you are using the full available space on the SD card.
 - Memory Split: Make this 8MB (perhaps 0MB is possible, however I prefer allow some memory to the GPU to avoid potential instability).

Select “Finish” and allow your Raspberry Pi to reboot.

Give it a couple of minutes to reboot, and try to login again to ensure we can continue:

```
$ ssh pi@raspberrypi-zero.local
```

Great, we are almost done with the operating system setup. In the next lesson, I'll show you how to setup the root user.

By default, the root user is disabled, but we need to use it with our SFTP client later in the project.

11: Working as the 'root' user

RASPBERRY PI GETTING STARTED SERIES

Working As The 'Root' User

In this lesson you will learn how to work on your Raspberry Pi as the "root" user.



By default, the root user is disabled, but we need to use it with our SFTP client later in the project. An SFTP client is a program

that allows us to transfer files between two computers using the same secure protocol used in SSH.

When we are working on the Raspberry Pi using the terminal emulator and SSH, we can simply switch to the root user with the “sudo” command:

```
$ sudo su
```

This means that the root (or “super user”, hence “su”) user is available, but is only accessible via the “sudo” command. It is not enabled for remote access, such as SSH or SFTP.

Once you become “super user” with “sudo”, you can work as if you were logged in as “root”, and access parts of the file system that are only permitted to the root user, or edit files owned by root.

To exit the super user and go back to the normal “pi” user, type:

```
$ exit
```

Let’s enable external login to the root account to make our life a bit easier.

First, login to your Raspberry Pi as “pi”:

```
$ ssh pi@raspberrypi-zero.local
```

Then, change into the super user:

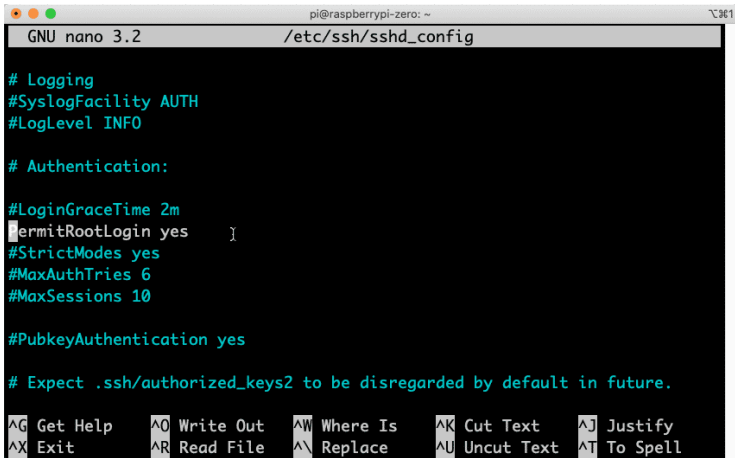
```
pi@raspberrypi-zero:~ $ sudo su
```

And finally, use the nano text editor to edit the sshd_config file:

```
root@raspberrypi-zero:/home/pi# nano /etc/ssh/sshd_config
```

Scroll down the sshd_config file to find the PermitRootLogin

directive. Its argument should be “yes”. Remove the “#” to un-comment it.



```
pi@raspberrypi-zero: ~
GNU nano 3.2 /etc/ssh/sshd_config

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.

^G Get Help      ^O Write Out    ^W Where Is    ^K Cut Text     ^J Justify
^X Exit          ^R Read File    ^N Replace     ^U Uncut Text  ^T To Spell
```

The sshd_config file.

Type Control-X to exit and save the changes in the file.

Next, restart the SSH daemon so that the changes become effective:

```
root@raspberrypi-zero:/home/pi# /etc/init.d/ssh restart
```

You are now almost ready to be able to login to your Raspberry Pi using the root user. The last thing to do is to set a password for the root user.

Do this with the “passwd” command:

```
root@raspberrypi-zero:/home/pi# passwd rootNew
password:Retype new password:passwd: password updated
successfully
```

The utility will ask you for the new password for the root user. I use “raspberry”, since I have no security concerns. If I was working to create a security-sensitive application, I would only

enable the root user during development, and disable it for production.

Time to test your “new” root user. Exit super user, then exit the pi user, and then login to your Raspberry Pi using “root”:

```
root@raspberrypi-zero:/home/pi# exit  
exitpi@raspberrypi-zero:~$  
exit  
logout  
Connection to raspberrypi-zero.local closed.  
Peters-iMac:~ peter$ ssh root@raspberrypi-zero.local  
root@raspberrypi-zero.local's password:Linux  
raspberrypi-zero.local 4.19.97+ #1294 Thu Jan 30 13:10:54  
GMT 2020 armv6l  
The programs included with the Debian GNU/Linux system are free software;the exact distribution terms for each program are described in theindividual files in /usr/share/doc/*/copyright.  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extentpermitted 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.  
root@raspberrypi-zero:~#
```

In the command line session above, I have marked in bold the “root” user in my new SSH command. As you can see, I was able to login to my Raspberry Pi using “root”.

Good work so far.

You can log out from root.

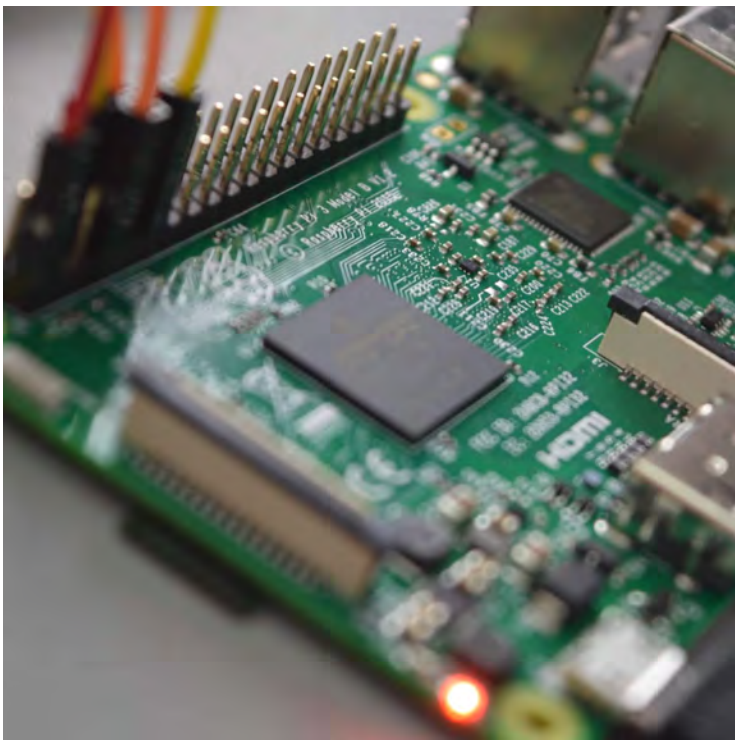
Your Raspberry Pi is now ready for development, but there’s one more critical set of operations I’d like to show you: backing up and restoring your Raspberry Pi SD Card. This will save you countless hours and frustration when things go wrong, and give you a reliable way to recover from disaster.

12: Raspberry Pi pins, roles, and numbers

RASPBERRY PI GETTING STARTED SERIES

Raspberry Pi Pins, Roles, And Numbers

In this lesson you will learn about the Raspberry Pi's pins, their roles, and how to address them.



It is time to dig into your first hands-on experiments. In the next few chapters you will learn how to work with an LED and

a button using Python. This is the first step towards doing much more interesting things.

On the Raspberry Pi, you can connect external devices, like buttons and LEDs, to the various pins that are exposed through a 40-pin header. Each of the pins perform specific functions.

Most of the pins can perform multiple functions, as you will see later.

There are two attributes that are most important when it comes to the Raspberry Pi pins:

- The pin number, which allows you to refer to a pin inside your Python scripts
- The pin capabilities, so that you know what it is that you can do with a pin.

Let's have a look at some examples:

- Pins 1 and 17 provide 3.3V power.
- Pins 2 and 4 provide 5V power
- Pins 6, 9, 14, 25, 30, 34 and 39 are GND.
- Pin 22 is a general-purpose input/output (GPIO) pin. You can use it to drive an LED or sense the state of a button.
- Pin 14 is a GPIO, but also the transmit pin of the UART (serial interface).
- Pin 32 is a GPIO, but also a PWM pin.

In the diagram below you can see a full map of the Raspberry Pi 40-pin header. The map shows the primary and secondary function of each pin.


```
pi@raspberrypi-zero: ~  
Revision      : 9000c1  
SoC           : BCM2835  
RAM          : 512Mb  
Storage      : MicroSD  
USB ports    : 1 (excluding power)  
Ethernet ports : 0  
Wi-fi       : True  
Bluetooth   : True  
Camera ports (CSI) : 1  
Display ports (DSI) : 0  
  
J8:  
  3V3 (1) (2) 5V  
  GPIO2 (3) (4) 5V  
  GPIO3 (5) (6) GND  
  GPIO4 (7) (8) GPIO14  
  GND (9) (10) GPIO15  
  GPIO17 (11) (12) GPIO18  
  GPIO27 (13) (14) GND  
  GPIO22 (15) (16) GPIO23  
  3V3 (17) (18) GPIO24  
  GPIO10 (19) (20) GND  
  GPIO9 (21) (22) GPIO25  
  GPIO11 (23) (24) GPIO8  
  GND (25) (26) GPIO7  
  GPIO0 (27) (28) GPIO1  
  GPIO5 (29) (30) GND  
  GPIO6 (31) (32) GPIO12  
  GPIO13 (33) (34) GND  
  GPIO19 (35) (36) GPIO16  
  GPIO26 (37) (38) GPIO20  
  GND (39) (40) GPIO21  
  
For further information, please refer to https://pinout.xyz/  
pi@raspberrypi-zero:~ $
```

Pinout is a handy utility that shows basic information about your Raspberry Pi and a map of its header.

Pinout is already installed in the full version of Raspbian, but not in Lite.

To install pinout in Raspbian Lite, issue this command:

```
$ sudo apt install python3-gpiozero
```

To show the pin map, type this:

\$ pinout

As you can see in these two maps, each pin has two numbers. For example, with reference to screenshot from the Pinout output above, the second pin from the top in the left column is "GPIO2 (3)".

This means that you can refer to this pin in two ways in your Python scripts:

- With its GPIO reference number, which is "2".
- With its board reference number, which is "3".

Similarly, for "GPIO11 (23)", the GPIO number is 11, and the board number is 23.

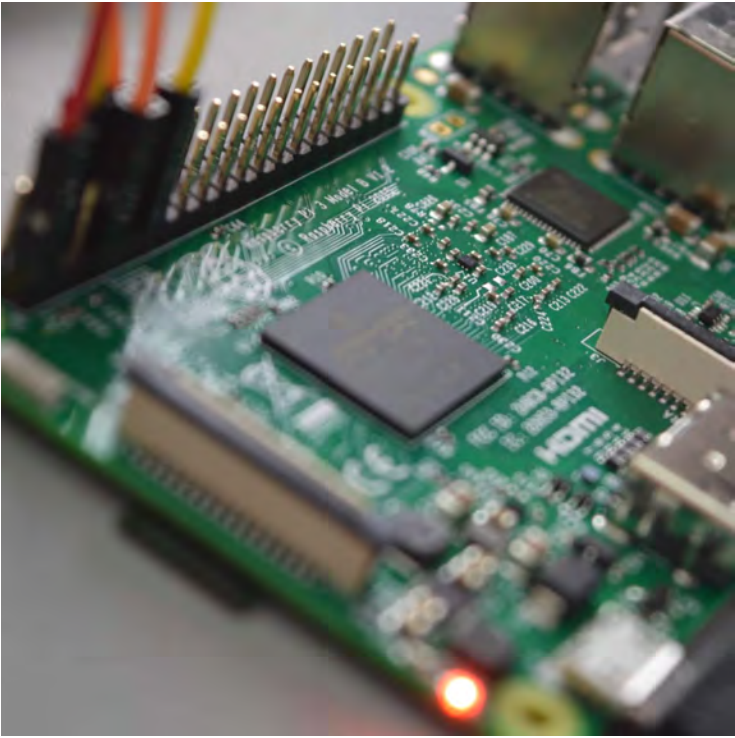
In the scripts in this project we will use the GPIO numbering system unless noted otherwise.

13: A taste of Python on the Command Line Interpreter

RASPBERRY PI GETTING STARTED SERIES

A Taste Of Python On The Command Line Interpreter

In this lesson you will practice with the use of the Python language on the Raspberry Pi command like interpreter (CLI).



Let's play with Python!

The Raspberry Pi is a full Linux computer, so you can use virtually any programming language with it. However, Python is the one language that stands out. Python is the language with the widest adoption among Raspberry Pi users. This means that it has (by far) the best and most plentiful documentation, widest range of available libraries that you can use in your programs, and drivers for all sorts of external hardware.

For this reason, we will be using Python in this project.

In this chapter you will try out a few simple experiments with Python that do not require external hardware. This is an opportunity to get used to using Python on the command line.

If you have never worked with Python before, do not worry. We'll take this one step at a time and gradually build the skills you will need for the project.

Let's begin.

Log into your Raspberry Pi with the pi user. For me, the login looks like this (in bold is the command that I entered):

```
Peters-iMac:~ peter$ ssh pi@raspberrypi-zero.local
pi@raspberrypi-zero.local's password:Linux
raspberrypi-zero.local 4.19.97+ #1294 Thu Jan 30 13:10:54
GMT 2020 armv6lThe programs included with the Debian
GNU/Linux system are free software;the exact distribution
terms for each program are described in theindividual files in
/usr/share/doc/*/copyright.Debian GNU/Linux comes with
ABSOLUTELY NO WARRANTY, to the extentpermitted by
applicable law.Last login: Tue Mar 24 00:12:09 2020 from
192.168.112.13SSH is enabled and the default password for
the 'pi' user has not been changed.This is a security risk -
please login as the 'pi' user and type 'passwd' to set a new
password.pi@raspberrypi-zero:~ $
```

At the time I am writing this, Python is available in two versions. Python 2, and Python 3. Python 2 is a legacy version for which support is ending. So, we'll be working with Python 3.

You can use Python in two "modes".

First, you can write traditional Python programs and then execute them. We'll do that very soon.

Second, you can write interactive Python programs in the Command Line Interpreter (CLI). The CLI is a utility that evaluates a Python instruction as you issue it. It is a very good way to quickly try out an idea, or learn how to use a function.

Let's have a look at using Python on CLI now.

At the command line, type this:

```
pi@raspberrypi-zero:~ $ python3Python 3.7.3 (default, Dec 20
2019, 18:57:59)[GCC 8.3.0] on linuxType "help", "copyright",
"credits" or "license" for more information.>>>
```

This will invoke the CLI, and present you with the ">>>" prompt. You can type a Python command at the prompt, and the the CLI will evaluate it immediately.

Try out these commands (lines that start with ">>>" are where I have entered a command, lines without the prompt is where Python prints out the result of the command, and the numbers before the ">>>" are there for reference):

```
1- >>> 1+1 22- >>> 17/3 5.6666666666666673- >>> 17/3
# This is a division 5.6666666666666674- >>> division_result
= 17/35- >>> division_result 5.6666666666666676- >>>
print(division_result) 5.6666666666666677- >>>
round(division_result,2) 5.678- >>> 'Tech Exporations' 'Tech
Exporations'9- >>> str = 'Tech Exporations' 10- >>> str
'Tech Exporations'11- >>> print(str) Tech Exporations12-
>>> print(str + ' ' + 'RPi FS') Tech Exporations RPi FS13- >>>
3 * str 'Tech ExporationsTech ExporationsTech Exporations'14-
>>> str[5] 'E'15- >>> str[5:7] 'Ex'16- >>> str[0] 'T' >>>
```

Let's take some time to dissect some of these examples.

- In the first two prompts ("1+1" and "17/3"), you did simple arithmetic. You used the "+" and "/" operators to do addition and division. So, you can use Python as a calculators.
- In the third prompt I have added a comment using the "#" symbol. Any text after the "#" is ignored by Python.
- In the fourth prompt, you created a variable with the name "division_result" and store the result of the division in it. In Python, variables can store numbers, strings, arrays,

and all kinds of objects.

- In the fifth prompt, you typed in the name of the variable only, and what came back from Python is the value stored in it.
- In the sixth prompt you used the “print” function to show the content of the variable. The result is the same as if you have just typed in the name of the variable, and this is specific to how the CLI behaves in these cases.
- In the seventh prompt, you used the “round” function to round the result of the division to two decimal points.
- In prompt 8, you have created a string. Notice that you used single quotes to enclose the string.
- In prompt 9, you stored the string in a variable titled “str”.
- In prompt 10, you retrieved the value stored in “str”.
- In prompt 11, you did the same by using the “print” function.
- In prompt 12, you used the “+” operator to concatenate three smaller strings into a large one. The “+” operator concatenates strings when it is used with strings, and adds numbers when it is used with numbers.
- In prompt 13, you used the “*” operator to multiply a string by a number. The result is the the same string came out multiple times

(3 times, in this example).

- In prompts 14, 15, and 16, you use the string as an array of characters. Inside the square brackets, you make reference to the cell of this array from where you want to extract the character. Inside the “str” variable, you have stored the string “Tech Explorations”. The letter in cell “0” is “T”, and in cell “5” is “E”. You can also specify cell ranges, so the letters in the range from 5 (inclusive) to 7 (not inclusive) are “Ex”.

As you can see, Python, while very powerful, is very intuitive for beginners. You can be productive with it very quickly, without any formal study of the language.

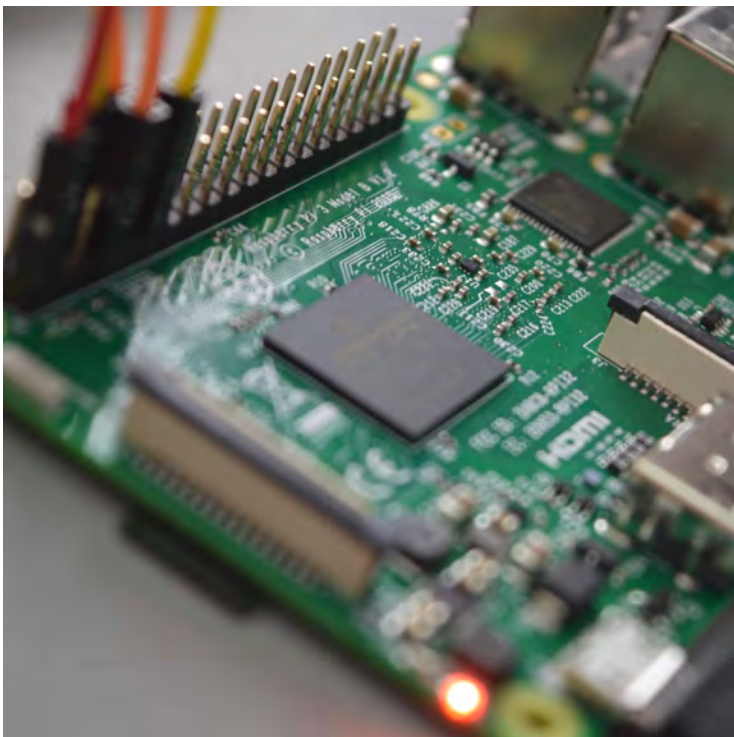
In the next chapter, I will show you the basics of functions. Functions is a programming construct that allows us to bundle functions together, and call them with a single name.

14: Python Functions

RASPBERRY PI GETTING STARTED SERIES

Python Functions

A function is a block of code that does something useful. Think of functions as mini programs that are part of a larger program. They are an extremely useful programming building block, and, of course, you will use it a lot in this project.



So, after the function definition, you have a line of code that multiplies the value stored in the “strn” variable by 3 and prints it to the console. Notice that the third line also begins

with "...". Python is not evaluating anything yet, it is waiting for you to give it the next instruction.

It is important to remember that white space is very important in Python. This is how Python can figure out which instruction belongs to a particular block. If you had not included the 3 spaces in the second line of this example, Python would assume that the new instruction does not belong to the function. This dependency on white space is a known pain point for Python programmers, and something that you will have to endure.

To keep this example super-simple, let's end the function here, with only a single line of code. Your cursor is in line three. Hit Enter one more time. You should see this:

Noticed the "def" keyword, followed by the function name, the parameter in parentheses, and the ":" in the end that indicates the end of the function definition.

In the following lines you can write the code that makes up the function block. Let's do this on the CLI:

Unless you are working on a trivial program, you will need a way to bundle multiple lines of code together. Modern programming languages have a handy way to do this: functions (also known as "methods").

A function is a block of code that does something useful. Think of functions as mini programs that are part of a larger program. They are an extremely useful programming building block, and, of course, you will use it a lot in this project.

Since we are working with Python and the Command Line Interface, I'd like to show you how to create a simple function in this chapter.

Login to your Raspberry Pi as "pi", and start the Python CLI:

```
$ python3
```

Now, let's create a function. In Python, we use the keyword "def" to define a new function. A function definition needs a name (so we can use this name to call the function from somewhere else in our program), and an optional one or more attributes. An attribute can be a variety of things, like strings, numbers, variables, and arrays.

Here's an example Python function, named "str_operation", that has a single argument "strn":

```
def str_operation(strn):
```

```
>>> def str_operation(strn):...
```

The "..." in the CLI show that Python is aware that we are creating a function. It is waiting for us to write the function code, and is not evaluating anything until we "tell" it that our function is complete. So remember, when you see "...", Python is waiting for you to type in an instruction that is part of the function.

Let's continue to the second line. Press the space bar three times to move the start of the next instruction towards the right by three spaces, and type this in:

```
print(3 * strn)
```

Your CLI should look like this:

```
>>> def str_operation(strn):... print(3 * strn)...
```

```
>>> def str_operation(strn):... print(3 * strn)...>>>
```

After the blank line, the CLI shows the familiar ">>>" prompt. Whatever you type now will be evaluated by python. Your new function exists in memory, and it will produce a result on the screen when you call it by name and pass an argument to it.

Now, try this:


```
>>> str_operation("Hello ")
```

You should get this back:

```
>>> str_operation("Hello ")Hello Hello Hello>>>
```

The string "Hello" has been copied three times in the console.

Nice!

You can modify this little function like this:

```
>>> def str_operation(strn):... a = 3 * strn... print(a)...>>>
```

This will evaluate the expression "3 * strn" and store the result in variable "a", then print it to the console.

When you call it, the result is exactly the same as in the first version:

```
>>> str_operation("Hello ")Hello Hello Hello>>>
```

Yet another version of this example is the following:

```
>>> def str_operation(strn, num):... a = num * strn...  
print(a)...>>>
```

In this version, I have added a new parameter, called "num". In the second line, I have replaced the original "3" with "num", so that I can repeat the string stored in "strn" as many times as I like without having to modify the program.

I can call this function like this:

```
>>> str_operation("Hello ",4)Hello Hello Hello Hello>>>
```

Notice that inside the parentheses I have provided the string argument, and the number argument. In this call, the arguments appear in the same order as in the function definition; this is very important to remember as Python

excepts the arguments to be in the order defined in the definition of the function.

As you can see, we use functions to bundle together instructions. We can call a function by its name, and pass any required parameters to it in order to have its instructions executed. But functions can also return values to the caller. Have a look at this variation of the "str_operation" function:

```
def str_operation(strn, num): return num * strn
```

In this example, the function definition is the same as in the original example. But, instead of the function doing the calculation and printing the result to the console, it simply returns the result of the calculation using the "return" keyword. The caller of the function can then decide what to do with the returned value.

Here is the example on the CLI:

```
>>> def str_operation(strn, num):... return num * strn...>>>
a = str_operation("Hello ", 3)>>> print(a)Hello Hello Hello
```

In this example, I have defined the new version of str_operation to return a value. Then, I call the function with its parameters, and store the returned value in a variable. Finally, I print the value stored in the "a" variable in the console.

There is a lot of flexibility in writing functions, using the return function, and in general how you structure your Python programs. For example, if you prefer a multi-step approach, you can write the function like this:

```
def str_operation(strn, num): new_string = num * strn
return new_string
```

Here, your function does the calculation in one step, and the return of the value in another step.

Or, you can choose a more compact way of coding, like this:

```
def str_operation(strn, num): return num *  
strnprint(str_operation("hello ", 3))
```

In this example, you have condensed four lines of code into 2 (the return line, and the print line), skipping the intermediate steps that involve the variables.

My personal preference is to code for clarity. I imagine other people reading my code, or myself reading my code sometime in the future. Will I (or my readers) be able to understand what my code does? Will they be able to improve it, or add new features? This means that usually it is best to be more verbose, with sensible function and variable names, and a logical program structure.

From here, functions and programs get more elaborate and complicated, but you are already familiar with the basic principles.

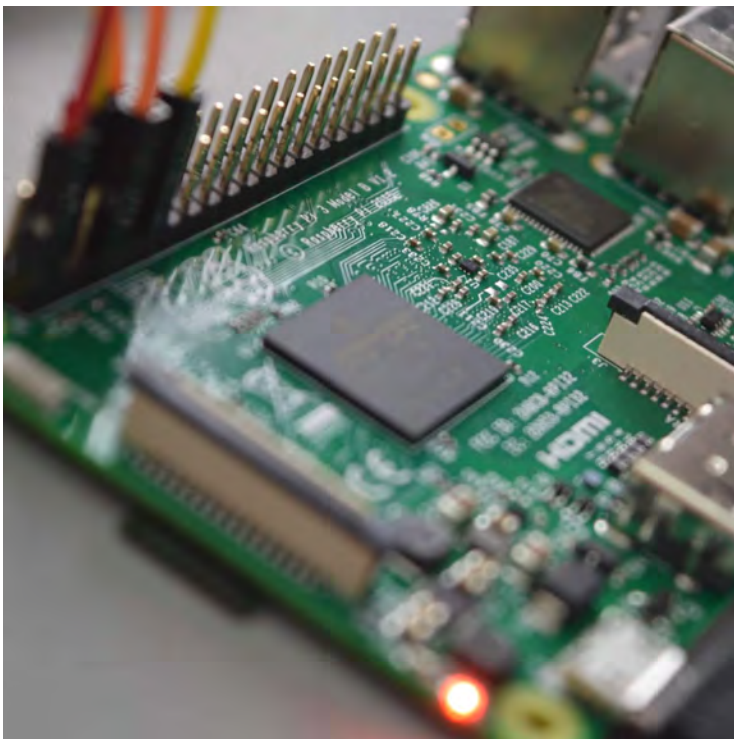
Ready for something new? In the next chapter I will show you how to write a Python script that you can call from the command line, without using the CLI. You will build on the knowledge you gained in this chapter.

15: A simple Python program

RASPBERRY PI GETTING STARTED SERIES

A Simple Python Program

Let's convert the little program you wrote in the CLI in the last chapter into a proper Python program. At the same time, you will learn the basics of the Vim editor, which you will use extensively in this project.



You will write this program over multiple iterations. Let's start with the first version. To write it, you will use a powerful editor called "Vim". This editor has existed since the beginning of time (at least the beginning of computer time), and you will find it on every Linux computer.

Login to your Raspberry Pi as "pi". Before you write your first program, you must install the Vim text editor. You can install Vim using the Debian "apt-get" utility, like this:

```
pi@raspberrypi-zero:~ $ sudo apt-get install vim
```

The "sudo" command will temporarily raise your user level to "root" so that you can install Vim on the system.

Once Vim is installed, use it to create a Python program:

```
pi@raspberrypi-zero:~ $ vim example.py
```

You are creating a new file called "example.py" with the vim editor. You will see a blank editor window, like the one below:



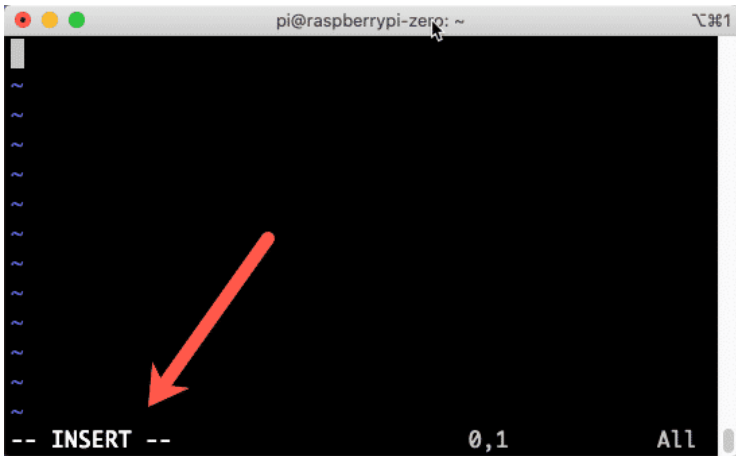
A blank Vim editor waiting for your program.

Vim is extremely powerful once you memorise a few hundreds

of shortcuts. There is no mouse support, and no obvious way of doing anything with it. But you can do everything you need with just two or three keyboard shortcuts. Even my dog can remember two or three commands.

The first thing you will want to do is, of course, to enter some text.

To do this type “i”. The “i” command will instruct Vim to enter “insert mode” so that you can type text in its buffer. You know that its Ok to type text because as soon as you type “i”, Vim will update its status to “- - INSERT - -”, as you can see below.



Use “i” to change Vim mode to “insert”.

Now that you are in Insert mode, go ahead and copy this text into Vim:

```
def str_operation(strn, num): a = num * strn
print(a)str_operation('Hello ', 3)
```

Be careful with the white space, and remember to leave a blank line after the third line of the function. In Vim, while you are in Insert mode, you can use the space bar to leave a white space, Enter key to create a new line, and the

Backspace/Delete key to reduce indentation.

This program first defines the function, and then calls it. This is exactly what you did earlier in the CLI.

When you finish typing, you will have your Python program in the Vim buffer, like in the example below.



```
pi@raspberrypi-zero: ~  
import sys  
def str_operation(strn, num):  
    a = num * strn  
    print(a)  
  
str_operation('Hello ', 3)  
~  
~  
~  
~  
~  
~  
-- INSERT --          7,1          All
```

Your first Python program in Vim.

The program looks ready to execute. But first you must save it to disk and exist Vim. As you are still in Insert mode, you need to hit the Esc key once, to go into command mode. Your editor should look like the example below.

You now know enough vim to actually complete this project, but remember that there's much more you can learn to become a Vim Ninja.

You should now be back in the command line. Let's run the program. To run a Python program, you must pass it as an argument to the Python interpreter. For us, the interpreter is Python3, so execute your program like this:

```
pi@raspberrypi-zero:~ $ python3 example.pyHello Hello Hello
```

Easy!

What I'd like to do next is to parametrise this program. Instead of the program outputting the exact same thing every time you run it (=boring), how about you make a few small changes so that you can control what it outputs when you invoke it? For example, you could write something like this:

```
pi@raspberrypi-zero:~ $ python3 example.py 'This is AWESOME!' 5
```

And your program would output "This is AWESOME!", five times.

The way you did this is by provide input to your program from the command line. Your program needs a way to take your input from the command line and process it.

Luckily, Python has a simple way to do that. There is a built-in library called "sys" which contains an attribute called "argv". Argv is really an array, and you can retrieve command line arguments from it like this:

- To grab the first command line argument, use `sys.argv[1]`.
- To grab the second command line argument, use `sys.argv[2]`.
- Etc.

Ok, let's modify your program. Again, use Vim to open it and edit it:

```
pi@raspberrypi-zero:~ $ vim example.py
```

This time you will not see a blank buffer, but your program. Remember how to go into Insert mode? That's right, type "i". Use the keyboard cursors to navigate up/down, the Enter key to add a new line, and the Delete key to delete text.

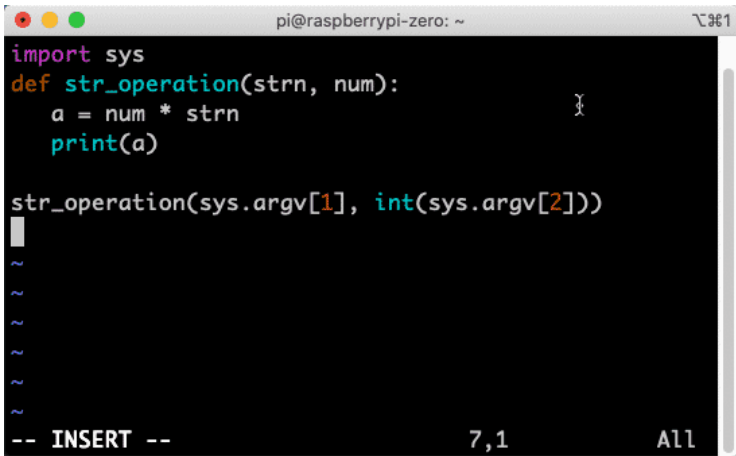
Add a new line above the function definition and type "import sys". Replace the arguments in the function call of the last line with the calls to sys.argv.

There is one more detail to note: sys.argv will return a string for each command line argument. For our multiplication purposes, we want the "num" variable to contain a number. So, you must explicitly instruct Python to convert the string from the second argument into a number. Python has a keyword for this "int". Enclose the second sys.argv as a parameter for "int".

Here is the completed program:

```
import sys
def str_operation(strn, num): a = num * strn
print(a)str_operation(sys.argv[1], int(sys.argv[2]))
```

Your program should now look like this in the editor:



```
pi@raspberrypi-zero: ~  
import sys  
def str_operation(strn, num):  
    a = num * strn  
    print(a)  
  
str_operation(sys.argv[1], int(sys.argv[2]))  
~  
~  
~  
~  
~  
~  
-- INSERT --          7,1          All
```

To modify an existing program, go into Insert mode with “i”.

Ready to try it out? Save the buffer and quit Vim (“:wq”). Then invoke your program:

```
pi@raspberrypi-zero:~ $ python3 example.py 'This is  
AWESOME! ' 5This is AWESOME! This is AWESOME! This is  
AWESOME! This is AWESOME! This is  
AWESOME!pi@raspberrypi-zero:~ $
```

Try different combinations of strings and multipliers, and each time your program will produce something different. Python is an extremely flexible multiple purpose language, while at the same time being friendly to beginners.

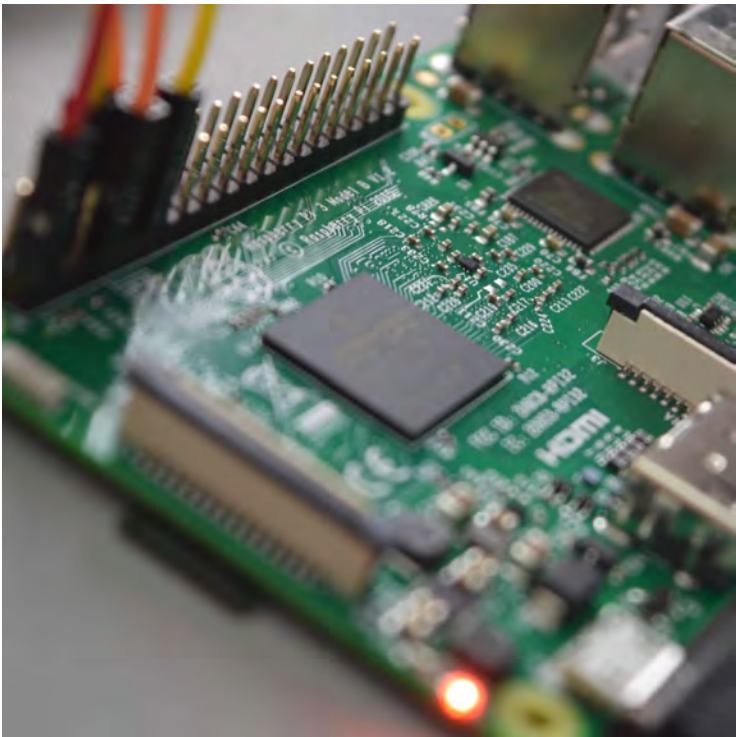
Are you ready for the next step? How about you learn how to use Python to control an LED? Time to construct a simple circuit, in the next chapter.

16: Wire a simple circuit

RASPBERRY PI GETTING STARTED SERIES

Wire A Simple Circuit

In this lesson, you will assemble a simple circuit that contains an LED and a button. Once you have your circuit ready, you'll learn how to work with it through several simple Python scripts.



You may not feel like it, but you know enough Python to be able to manipulate simple hardware connected to your Raspberry Pi. In this lesson, you will assemble a simple circuit

that contains an LED and a button. Once you have your circuit ready, you'll learn how to work with it through several simple Python scripts.

Before you plug anything to your Raspberry Pi's header pins, you must shut it down and remove the USB power cable. Unlike the Arduino, the Raspberry Pi is much more sensitive to things like static electricity and short circuits. It is easy to damage it. By turning off power, you protect your Raspberry Pi

At the command line, type this:

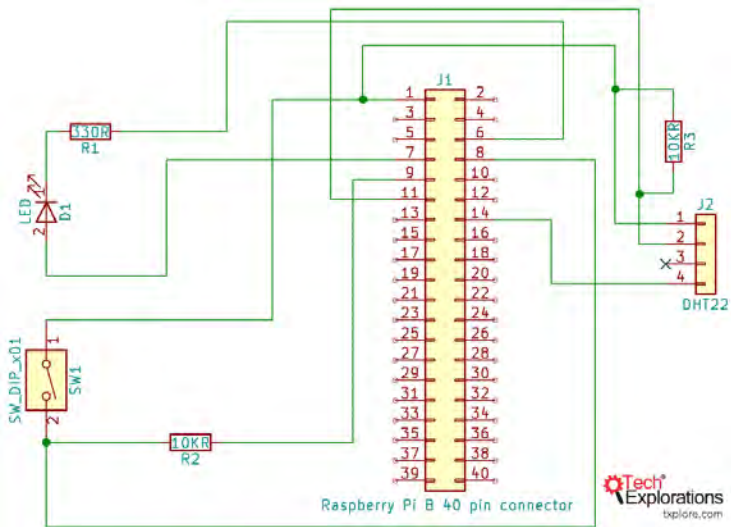
```
pi@raspberrypi-zero:~ $ sudo shutdown -h now
Connection to raspberrypi-zero.local closed by remote host.
Connection to raspberrypi-zero.local closed.
Peters-iMac:~ peter$
```

The shutdown command must be given as a super user, hence the "sudo" in front of it. The "-h" switch instructs Raspbian to "halt", ie. not restart. And "now" will cause the shutdown to take place immediately.

You can also schedule the shutdown for sometime in the future.

Once the green activity LED shows no activity, remove the power USB cable. Now you can safely plug external wires into the header pins.

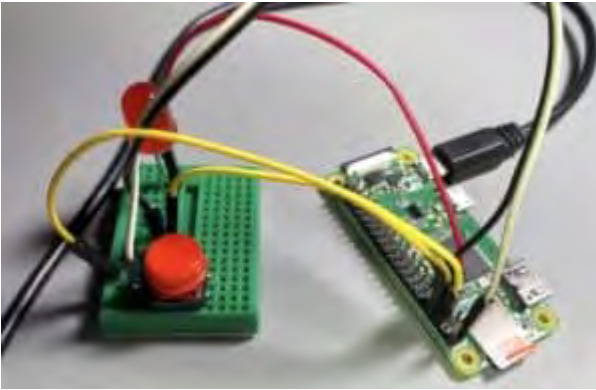
Use the circuit depicted in image below to assemble a simple circuit that contains a button, an LED, and two resistors. Don't worry about the two components on the right side of the schematic (marked J2 and R3); you will implement this part in a later chapter.



This schematic will guide you in the assembly of a simple circuit

Start with the LED and its 330 current-limiting resistor. The anode (long pin) of the LED is connected to GPIO 4, which is physical pin 7.

For the button, be careful to connect one side to physical pin 1 (which provides 3.3V power), and the other side to physical pin 9 (GND) via a 10 k pull-down resistor and to GPIO14 (physical pin 8). You can see my assembled LED-button circuit in the photograph below. It isn't easy to copy the circuit from this photograph, so I advise you to work from the schematic above.



This is my assembled LED-button circuit.

If, in the testing you will do later, you find that the button presses are not reliably detected by the Raspberry Pi, try switching the 10K pull-up resistor for a larger one, such as a 50K resistor. In my testing, the larger pull-up makes the button press detection more reliable.

When you are ready, do a final check to ensure that the wiring is correct, and connect the power USB cable. Give your Raspberry Pi a couple of minutes to boot and connect to your Wifi network, and continue in the next chapter.

17: Control an LED with GPIOZERO

RASPBERRY PI GETTING STARTED SERIES

Control An LED With GPIOZERO

You have an LED connected to GPIO4. How can you make it blink? I'll show you in this lesson.

There are many ways to achieve the same result when it comes to programming and electronics. I will show you two ways to manipulate the LED.



You have an LED connected to GPIO4. How can you make it blink? I'll show you in this lesson. As you can probably guess, there are many ways to achieve the same result when it comes to programming and electronics. I will show you two ways to manipulate the LED.

The first one involves using a feature of the [GPIO Zero library](#) that you installed in an [earlier lesson](#). Back then, you only used this library to print out a pin map on the console. But the library contains a lot of functionality, including simple functions to manipulate an LED.

Let's try it out.

Login to your Raspberry Pi as the "pi" user. Then, invoke the CLI for Python 3:

```
pi@raspberrypi-zero:~ $ python3Python 3.7.3 (default, Dec 20
2019, 18:57:59)[GCC 8.3.0] on linuxType "help", "copyright",
"credits" or "license" for more information.>>>
```

Next, import the LED module from the gpiozero library:

```
>>> from gpiozero import LED
```

This gives you access to the specific functions that allow you to manipulate the state of an LED. Before you can turn the LED on, you must define the GPIO to which it is connected. Remember you have connected the anode of the LED to GPIO4.

So, you will issue this command:

```
>>> led = LED(4)
```

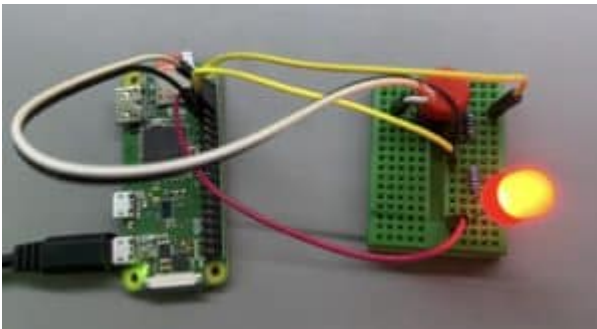
What you just did is to create an LED object that is connected to GPIO4 and assigned it to the "led" variable.

Now, you can call the "on()" and "off()" functions on this object to turn it on and off.

Try this:

```
>>> led.on()
```

Did your LED turn on? It should look like this:



My LED, on.

Can you turn it off?

Try:

```
>>> led.off()
```

Here's the complete CLI session:

```
pi@raspberrypi-zero:~ $ python3Python 3.7.3 (default, Dec 20
2019, 18:57:59)[GCC 8.3.0] on linuxType "help", "copyright",
"credits" or "license" for more information.>>> from gpiozero
import LED>>> led = LED(4)>>> led.on()>>> led.off()>>>
exit()
```

You can also achieve the same thing by writing a small Python program and executing it with the Python3 interpreter.

Create a new text file with Vim, and call it "led_blink_gpiozero.py". Type this code in the buffer:

```
from gpiozero import LEDfrom time import sleepled =
LED(4)while True: led.on() sleep(1) led.off() sleep(1)
```

Your Vim should look like this:

```
pi@raspberrypi-zero: ~
from gpiozero import LED
from time import sleep

led = LED(4)

while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
~
~
~
~
~
-- INSERT --          11,1          All
```

This program will blink an LED on and off in 1 second intervals.

Save your program and exit Vim (":wq").

Now, execute it like this:

```
$ python3 led_blink_gpiozero.py
```

Your LED will blink on and off.

In the code for this program, we imported the “time” Python module because it contains the “sleep” function. The “sleep” function is useful when you want to delay the execution of a program for a while. In our case, we used it to keep the LED on or off for 1 second. Feel free to replace “1” with another number if you want to create a different delay.

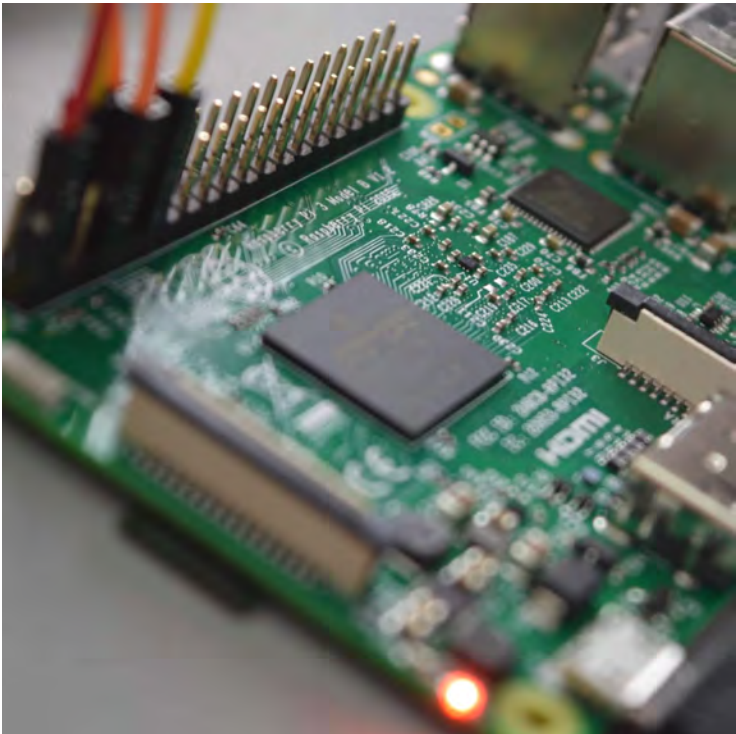
To end the program, use this keyboard shortcut: Ctr-C.

18: Read a button with GPIOZERO

RASPBERRY PI GETTING STARTED SERIES

Read A Button With GPIOZERO

A momentary button is an input device. It is essentially a sensor with only two possible states: “pressed” and “not pressed”. In your circuit, you have connected the output of the button to GPIO14. Your Python program will check the voltage at GPIO14 to determine if the button is pressed or not.



You now know how to “do” output. It’s time to learn “input”.

A momentary button is an input device. It is essentially a sensor with only two possible states: “pressed” and “not pressed”. In your circuit, you have connected the output of the button to GPIO14. Your Python program will check the voltage at GPIO14 to determine if the button is pressed or not.

Unlike with the LED example, this time we’ll go straight to write a regular Python program. This is because we want to get Python to read the state of GPIO14 many times per second, so that as soon as we press the button, Python can print a message on the screen to let us know if has detected the press.

We can also do this on the CLI, but it will be a clunky implementation which I prefer to avoid.

Use Vim to create a new Python program:

```
$ vim button_gpiozero.py
```

Copy this code into the Vim buffer:

```
from gpiozero import Buttonimport timebutton =  
Button(14,None,True)while True: if button.is_pressed:  
print(“Button is pressed”) else: print(“Button is not pressed”)  
time.sleep(0.1)
```

Save the buffer to disk, and quit Vim (“:wq”).

There’s a few new elements in this code, so lets take a moment to learn.

In the first line, you are importing the Button module, another member of the gpiozero library.

In the second line you import the “time” module, so that you can insert a tiny delay later in your program.

This is important because without this delay, your program will sample the button GPIO as fast as your Raspberry Pi can, leaving little time for anything else. We only want to read the state of a button, not to totally dominate the CPU.

Line three is a little challenging because it contains three parameters. You can find detailed information about these parameters in the [gpiozero documentation](#) for the Button object. The first one is the GPIO number. Since the button is connected to GPIO14, you'll enter "14" in the first parameter.

The second parameter controls the type of pull-up/down resistor we are using. The Raspberry Pi can use internal pull-up resistors resistor, but in our circuit we have provided an external pull-down resistor. This resistor ensures that voltage at GPIO14 is equal to GND when the button is not pressed.

For an article on pull-up/down resistors, please have a look at [this article](#).

Because of the presence of the external pull-down, we use "None" as the value of the second parameter.

In the third parameter, we include the active state value. Because when the button is pressed the voltage at GPIO14 is "HIGH", and when the button is not pressed the voltage is "LOW", we use the "True" value for this parameter. If the voltages were reversed (i.e. pressed buttons created a "LOW" voltage, and not pressed button created "HIGH"), then for the third parameter we would write "False".

You have saved the configured Button object in the "button" variable, and then get into an infinite loop.

In the look, your program simply reads the state of the button and if it is pressed, it prints "Button is pressed" to the console. If it isn't pressed, it prints "Button is not pressed".

Run the program like this:

```
$ python3 button_gpiozero.py
```

Now press the button as see how the message on the console changes accordingly. It should look like this:

```
Button is not pressed
Button is not pressed
Button is not pressed
Button is not pressed
Button is not pressed
Button is not pressed
Button is not pressed
Button is not pressed
Button is not pressed
Button is not pressed
Button is not pressed
Button is not pressed
Button is not pressed
Button is not pressed
Button is pressed
Button is pressed
Button is pressed
Button is pressed
Button is pressed
Button is pressed
```

A button was just pressed.

It works!

19: Setup the DHT22 sensor with Git

RASPBERRY PI GETTING STARTED SERIES

Setup The DHT22 Sensor With Git

In this and the next lesson I'll show you how to setup and use a common environment sensor, the DHT22. With the DHT22 you can measure temperature and humidity. This is an integrated digital sensor that uses a communication protocol to "talk" to its host.



You should now be more comfortable working on the command line and writing small Python programs.

You know how to control an LED and read the status of a button. Both a simple devices with only two possible states each: either on or off, or pressed/not-pressed.

Sensors are sophisticated integrated devices that you can use to measure a range of environment factors, such as temperature and humidity, the strength of the Earth's magnetic field (or a small magnet), various forces and acceleration, even the concentration of various chemicals in the air.

In this and the next lesson I'll show you how to setup and use a common environment sensor, the DHT22. With the DHT22 you can measure temperature and humidity. This is an integrated digital sensor that uses a communication protocol to "talk" to its host. The host can be a microcontroller, like the Arduino, or a computer, like the Raspberry Pi.

The protocol is a language that the two devices use to communicate. The DHT22 will respond to a request for a reading from the Raspberry Pi with the raw numbers that represent the temperature and humidity. Unlike the button, which only has two states, a sensor like the DHT22 is a real "chat box" that sends several bytes of data, encoded according to its protocol, that the host has to decipher.

As you can see from this description, using a sensor is a bit more involved then using a button. As the programmer, you will need to understand the intricacies of the communications protocol, which include things like register addresses, specific timings for requests and events, and a deeper understanding of computer concepts like "Big-endian" and "Little-endian" and checksums.

That's too much if you just want to measure a temperature.

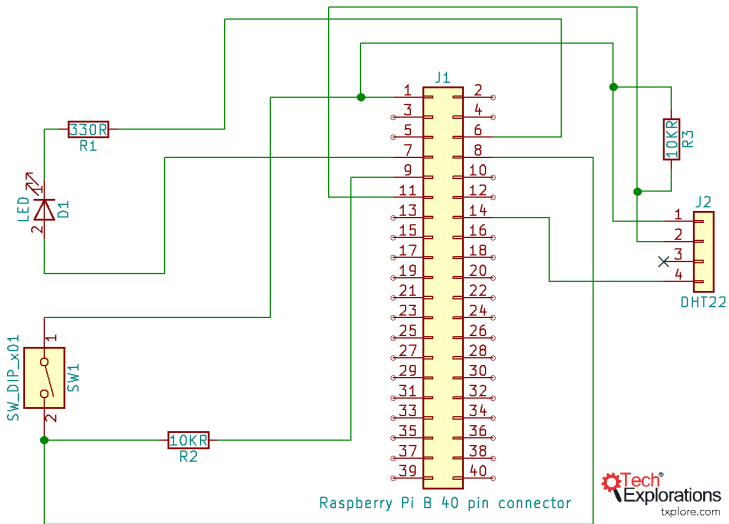
Luckily, all this low-level functionality is packaged nicely inside

libraries that we can easily integrate with our programs. We can then extract the information we need from the sensor with a simple command.

This is what we'll do with the DHT22. The process involves to install one of the many DHT22 libraries for Python and the Raspberry Pi, and then to run one of the example programs.

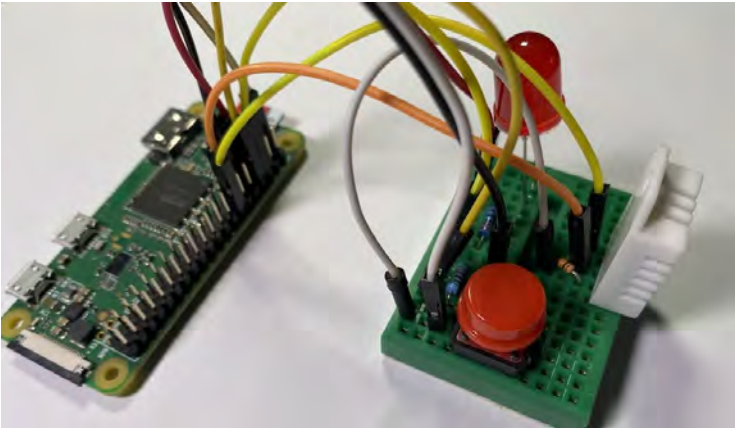
First, let's add the DHT22 sensor on the breadboard and connect it to the Raspberry Pi.

Go ahead and wire the right part of this schematic, that includes the DHT22 sensor and the 10 k resistor (see schematic below).



This schematic will guide you in the assembly of a simple circuit

In the photograph below you can see my assembly of the circuit on a mini breadboard.



The circuit, assembled on a mini breadboard.

Don't forget to turn off your Raspberry Pi before you connect the sensor. Do that like this:

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

Wait for the shutdown to complete, and then remove the power cable.

Once you have assembled and confirmed the circuit, apply power and wait for a couple of minutes, then login to your Raspberry Pi as "pi".

You will install the Adafruit DHT library for the Raspberry Pi. This library is hosted on [Github](https://github.com/adafruit/Adafruit_DHT). There are a few ways by which you can transfer the library (or any file) to your Raspberry Pi, but for now, and because this library is hosted on Github, the easiest is to use Git. With Git, you can clone (copy) a repository from Github to your Raspberry Pi.

But first, you need to install Git on your Raspberry Pi.

That is easy. Assuming you are logged in as Pi, copy this command:

```
$ sudo apt-get install git-core
```

This will install the Git tool on your Raspberry Pi. Wait for the installation to complete, and then configure the tool for use:

```
pi@raspberrypi-zero:~ $ git config --global user.email  
peter@txplore.compi@raspberrypi-zero:~ $ git config --global  
user.name "Peter"
```

Of course, use your own email address and name.

Now you can go ahead and clone the repository into your pi user home folder. Copy this into the command line:

```
pi@raspberrypi-zero:~ $ git clone  
https://github.com/adafruit/Adafruit_Python_DHT.gitCloning  
into 'Adafruit_Python_DHT'...remote: Enumerating objects:  
325, done.remote: Total 325 (delta 0), reused 0 (delta 0),  
pack-reused 325Receiving objects: 100% (325/325), 98.35 KiB  
| 234.00 KiB/s, done.Resolving deltas: 100% (176/176), done.
```

That's it, you now have a copy of the DHT22 library repository in your local folder.

The repository contains a few example programs that you can play with, as well as the library itself.

Before you can play with the examples, you must install the library.

Follow this process:

1. Go in the repository directory:

```
$ cd Adafruit_Python_DHT/
```

2. Run the setup utility:

```
$ sudo python3 setup.py install
```

The setup utility will produce a lot of output in the console, and eventually show you something like “Finished processing dependencies for Adafruit-DHT==1.4.0”.

The library is now installed.

Let’s play with it in the next lecture.

20: Use the DHT22 sensor

RASPBERRY PI GETTING STARTED SERIES

Use The DHT22 Sensor

In this lesson you will take measurements from the DHT22 sensor that you wired in the previous lesson.



Let's try out the DHT22 sensor with the Adafruit library.

From the root of your "pi" directory, change into the Adafruit DHT directory, and then into the examples directory:

```
$ cd Adafruit_Python_DHT/$ cd examples/
```

Then, run the AdafruitDHT.py example, like this:

```
$ python3 AdafruitDHT.py 2302 17Temp=23.6*  
Humidity=60.8%
```

Notice that there are two arguments that this example needs: the type of sensor you are using ("2302") and the GPIO to which the sensor data pin is connected to ("17"). When the example program runs, you can an output with the temperature and humidity.

Now that you know that your sensor works, you can take a closer look at the example program to learn how it works. You can see inside any text file using the "cat" command, like this:

```
$ cat AdafruitDHT.py
```

I am copying the program here, but to economise on space I have removed most of the multi-line comments:

```
~/Adafruit_Python_DHT/examples $ cat  
AdafruitDHT.py#!/usr/bin/python# Copyright (c) 2014 Adafruit  
Industries# Author: Tony DiCola# ... other commentsimport  
sysimport Adafruit_DHT# Parse command line  
parameters.sensor_args = { '11': Adafruit_DHT.DHT11, '22':  
Adafruit_DHT.DHT22, '2302': Adafruit_DHT.AM2302 }if  
len(sys.argv) == 3 and sys.argv[1] in sensor_args: sensor =  
sensor_args[sys.argv[1]] pin = sys.argv[2]else: print('Usage:  
sudo ./Adafruit_DHT.py [11|22|2302] <GPIO pin number>')  
print('Example: sudo ./Adafruit_DHT.py 2302 4 - Read from an  
AM2302 connected to GPIO pin #4') sys.exit(1)# Try to grab a  
sensor reading. Use the read_retry method which will retry  
up# to 15 times to get a sensor reading (waiting 2 seconds  
between each retry).humidity, temperature =  
Adafruit_DHT.read_retry(sensor, pin)# Un-comment the line  
below to convert the temperature to Fahrenheit.# temperature  
= temperature * 9/5.0 + 32# Note that sometimes you won't  
get a reading and# the results will be null (because Linux
```



```
can't# guarantee the timing of calls to read the sensor).# If
this happens try again!if humidity is not None and temperature
is not None: print('Temp={0:0.1f}*
Humidity={1:0.1f}%'.format(temperature, humidity))else:
print('Failed to get reading. Try again!') sys.exit(1)
```

The two import statements at the start of the program import the library itself (“Adafruit_DHT”), and the “sys” module. The “sys” (or “system”) module contains functions like “sys.argv”, used to get input from the command line, that you have seen in previous examples.

The “sensor_args” list matches our input of the sensor model to the internal code for this model that the library understands. So, when we specify “2302”, the library internally will use “Adafruit_DHT.AM2302”.

Let’s jump to the interesting parts. This line:

```
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
```

... is where the program asks the sensor for humidity and temperature readings. We can get both readings in a single line of code. The result is a multiple assignment, where the humidity value is stored in the “humidity” variable, and the temperature value is stored in the “temperature” variable.

The “read_retry” function needs two parameters: the type of sensor you are using, and the GPIO to which it is connected. Both of those values we specify in the command line when we invoke the program.

If you prefer the temperature in Fahrenheit, you can uncomment the conversion calculation just after the measurement is taken.

Another interesting segment in this program is this:

```
if humidity is not None and temperature is not None:
print('Temp={0:0.1f}*
```

```
Humidity={1:0.1f}%'.format(temperature, humidity))else:  
print('Failed to get reading. Try again!') sys.exit(1)
```

This is where the program tests for the validity of the result from the sensor. If the result is a number (instead of “None”), then it will print out the results. Otherwise it will print out a “failed” message.

Behind the scenes, a fairly sophisticated choreography of actions took place. But from our perspective, and end users of the sensor module and the library, we accomplished what we wanted with a single line of code. This is an example of the power of Python and its library ecosystem.

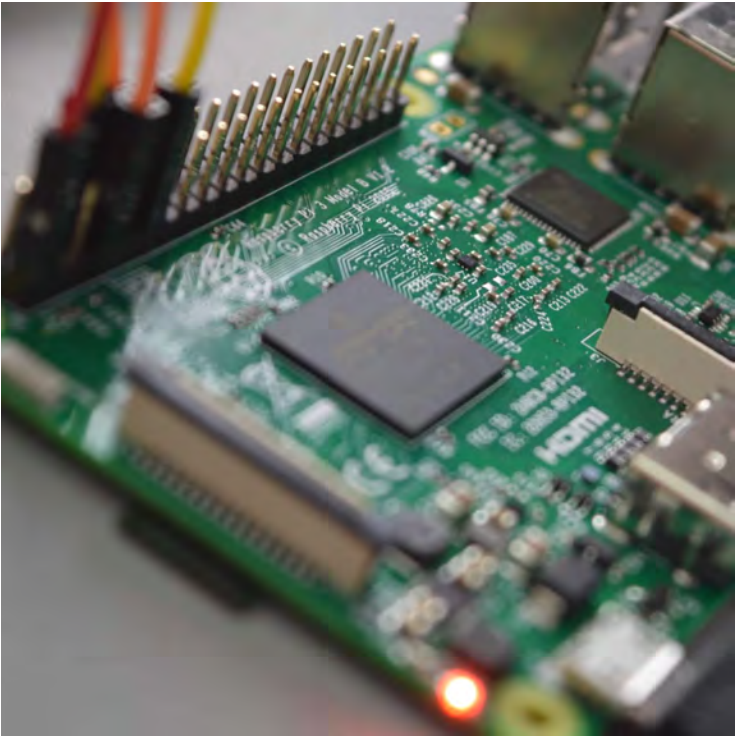
21. Raspberry Pi OS 32bit vs 64bit

RASPBERRY PI GETTING STARTED Guide SERIES

Raspberry Pi OS 64-bit vs 32-bit (Beginner's perspective)

Since the public release of the 64-bit Raspberry Pi OS a few years ago, there's still not a strong compelling reason for a beginner to use the 64-bit version over the 32-bit version.

In this article we explore the differences between the 32-bit and 64-bit versions of the Raspberry Pi OS, and when you should use the one over the other.



Which boards are compatible with the 64-bit Raspberry Pi OS?

If you happen to have a Raspberry Pi 1, 2, or Zero, then they won't support the 64-bit OS because their processors (BCM2835 and BCM2836) only have an architecture width of 32-bit. On the other hand, the newer boards – Zero 2 W, Pi 3, Pi 400 and Pi 4 (with the BCM2710 and BCM2711) are 64-bit so they will work with the 64-bit OS.

What are the benefits of 64-bit Raspberry Pi OS?

There are two main benefits to choosing the 64-bit Raspberry Pi OS - higher performance and better compatibility with 64-bit-only programs.

Higher performance

If you use your [Raspberry Pi](#) for CPU-intensive processes like file compression, and graphics manipulation, you will benefit from using the 64-bit OS.

Phoronix tested this on a Raspberry Pi 400 board which had 4GB of RAM.

Their first batch of tests [showed an average of 48% improvement](#),

In Phoronix's [second batch of tests](#), they saw the 64-bit OS finishing a process 86% of the time before the 32-bit OS did.

Access more RAM

Raspberry Pi 4 has an 8GB version which is the biggest beneficiary of the 64-bit OS.

If you used the 8GB Pi 4 with the 32-bit OS, you could still access all 8GB of RAM, but with the limitation that each process is limited to using 3GB each.

With the 64-bit OS, a process can access all 8GB at once.

Compatibility with 64-bit only software

Some software can only be run on a 64-bit OS, while others are not optimized for a 32-bit OS. By using a 64-bit OS, you could

potentially achieve better performance and run more software.

What are the benefits of 64-bit Raspberry Pi OS?

The downsides for a beginner can feel significant, given those incompatibilities can require some effort to correct them. In other cases, the software will not work at all.

This can make or break your project.

64-bit OS takes more RAM by default

If you have a more memory-starved Raspberry Pi, like the Raspberry Pi Zero 2 W (512MB of RAM) or the 1GB Raspberry Pi 3 or 4, you might be better off using the 32-bit OS.

Phoronix reports that, at idle/boot, the 64-bit OS takes up 196MB of RAM while the 32-bit uses 103MB.

Reports of incompatibility in software that runs fine on 32-bit

The 64-bit OS can sometimes have additional incompatibilities that the 32-bit software doesn't.

For example, if you want to use DRM-protected streaming services like Disney+ or Netflix, you can't do it on the 64-bit Chromium that comes pre-installed because the WidevineCDM library is not available. A side note: a GitHub post has shown [some progress on getting the Widevine library working on 64-bit Chromium.](#)

Other reports state that trying to run [snapt teams-for-linux](#), [X32Edit](#), Wolfram Mathematica and Remote Desktop doesn't work with the 64-bit OS.

You most likely have to do a clean install if you want to “upgrade”

There’s no easy upgrading from the 32-bit OS to the 64-bit OS.

While very technical people can do a process called crossgrading, it requires you to have a deep understanding of Linux.

For most people, you will have to do a clean install. If you already have a system based on the 32-bit OS, it would be a hassle to switch over.

Beginners might do better with the 32 bit OS for now

Since its official release, the 64-bit Raspberry Pi OS proves to be a challenge for beginners because it requires some technical knowledge and luck.

The 32-bit Raspberry Pi OS is more mature and therefore will provide a smoother experience.

If you are using your Raspberry Pi as a general-purpose computer such as running a VPN server, controlling components via the GPIO or using it as a desktop, then the 32-bit OS would perform more reliably.

Author information

Bruce Qin wrote this article. Bruce is currently the IoT gateway product manager from [Dusun IoT](https://dusun.io/). He specializes in hardware and software knowledge sharing for IoT devices, especially gateways, including modules, protocols, embedded development, and applications.