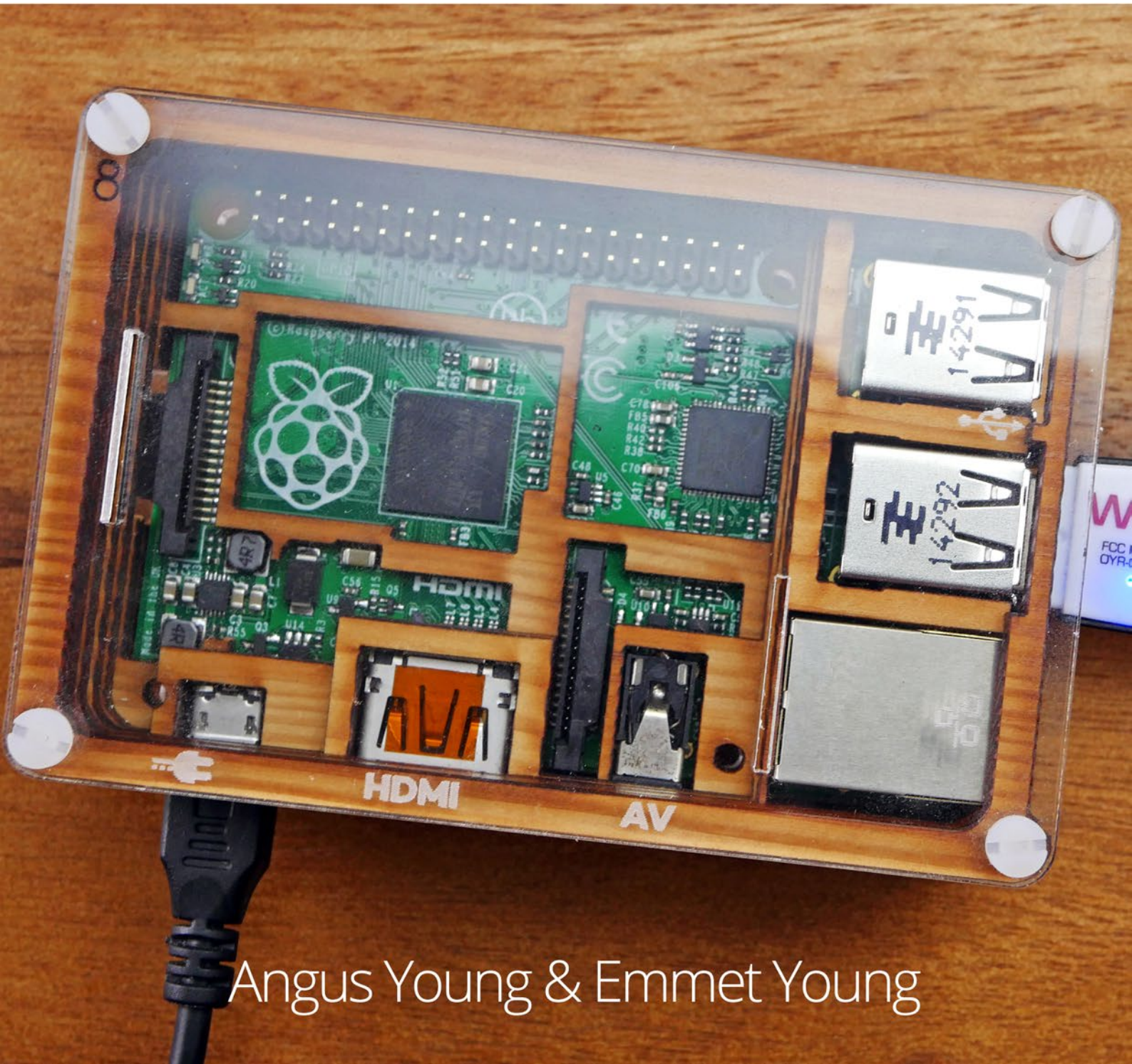
Pi My Life Up's **Getting Started** With the Raspberry Pi



Angus Young & Emmet Young

PiMyLifeUp

For permission contact Pi My Life Up at:

support@pimylifeup.com

The tutorials in this book are for general information purpose only.

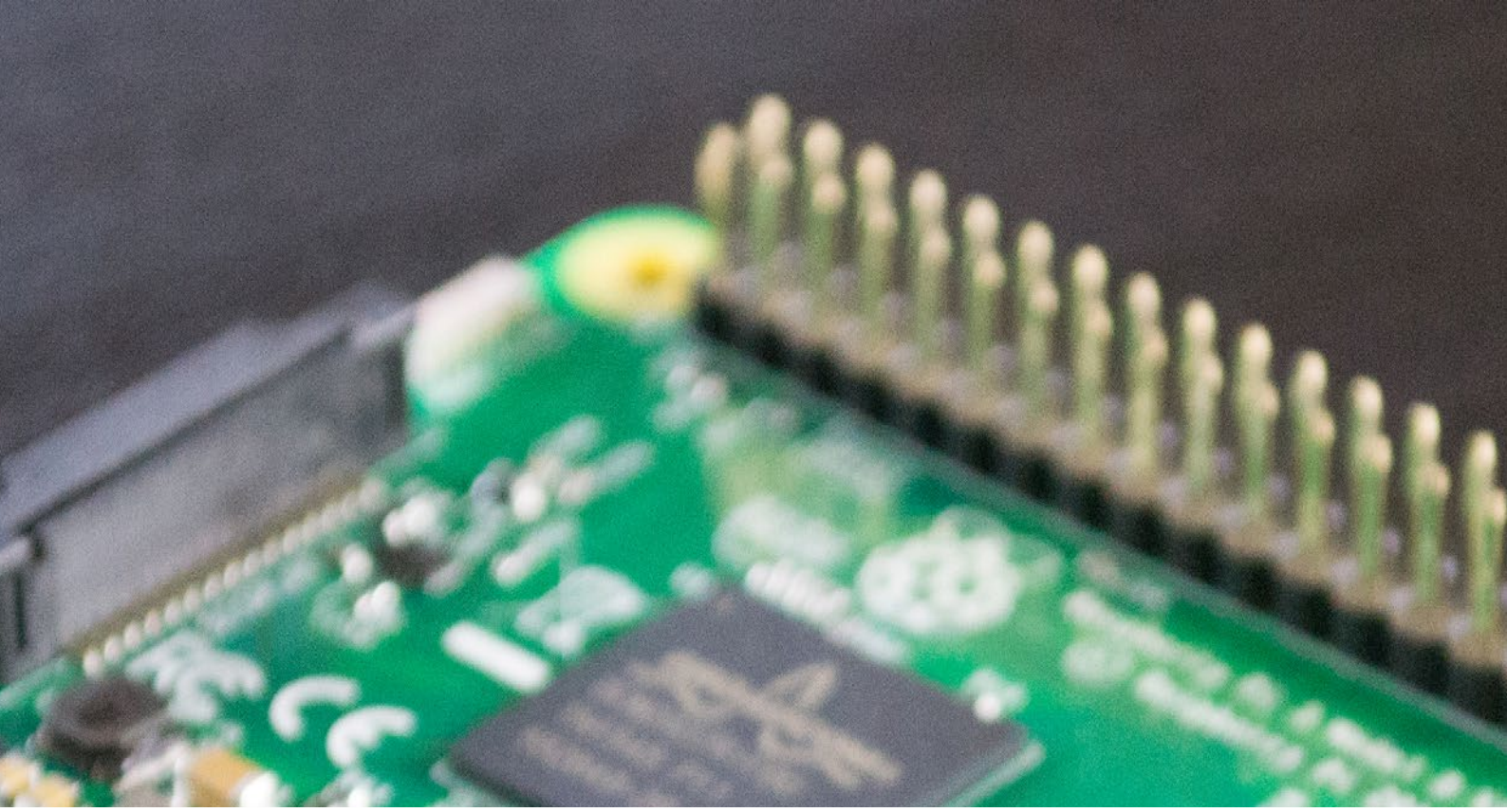
While we strive to keep our information up to date, we do not make any warranties about the completeness, reliability, and accuracy of this information.

Any action you take upon the information on within this book is strictly at your own risk.

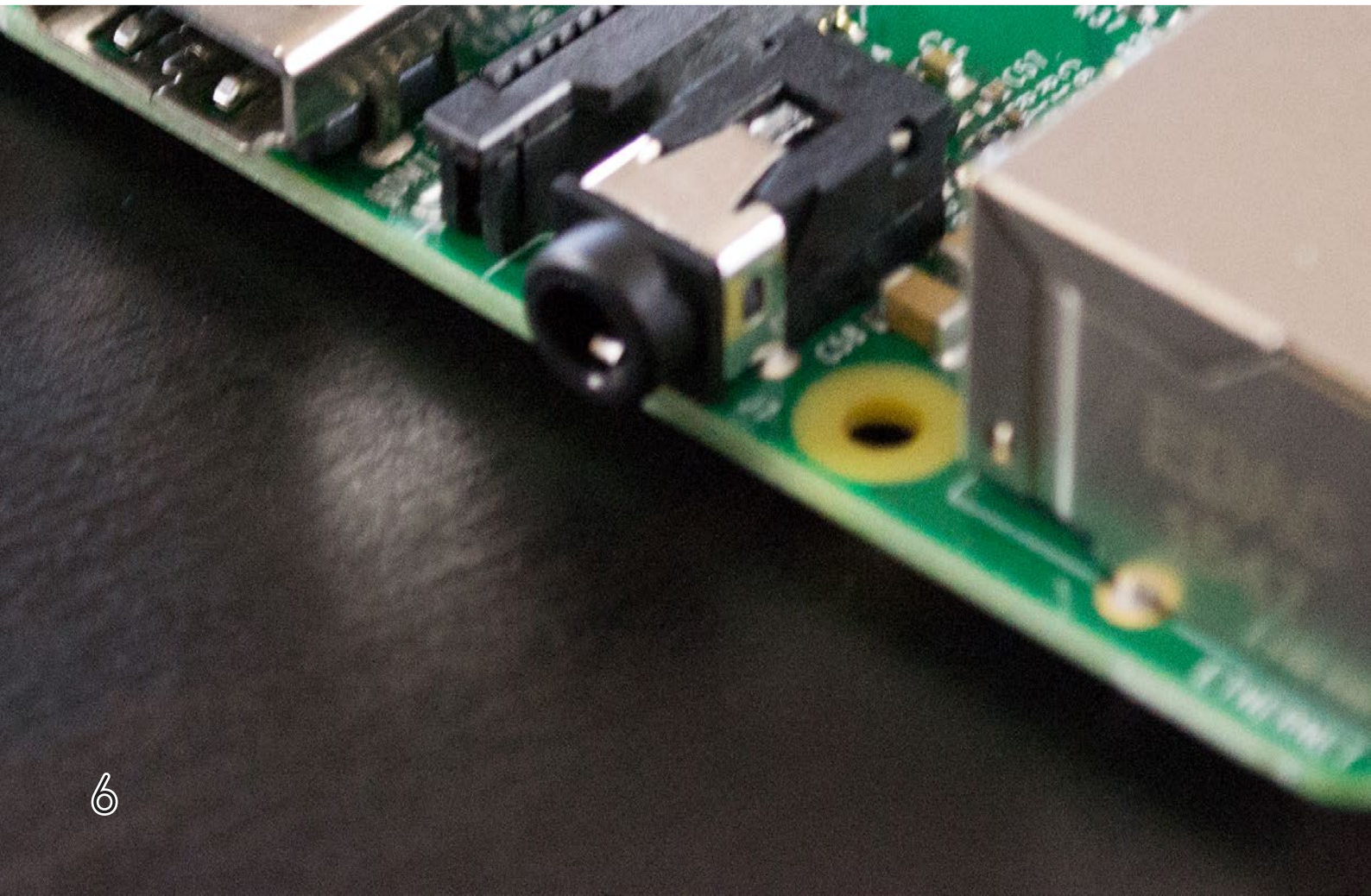
We will not be liable for any losses and damages in connection with the use of the information within this book

Contents


PiMyLifeUp.com



What is the Raspberry Pi



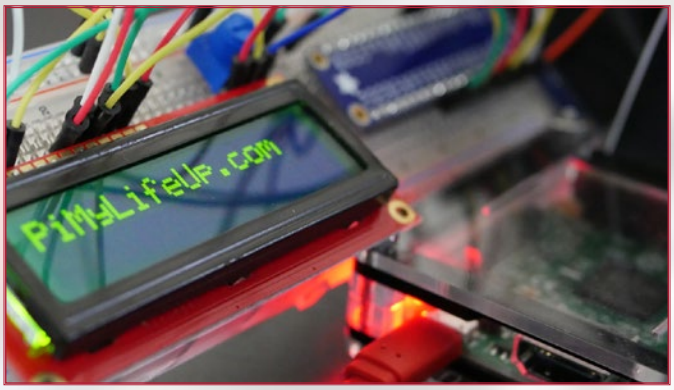
Quick Introduction
to the Raspberry Pi

The Raspberry Pi is a tiny computer about the size of a credit-card. The board features a processor, RAM and standard hardware ports you find with most modern computers, such as USB ports and an HDMI port.

The Raspberry Pi allows you to do most things a desktop computer can do such as document editing, playing HD videos, playing video games, coding and much more.

The Raspberry Pi obviously won't have as much power as a desktop PC but since it is a lot cheaper they make for great little computers you can play around with.

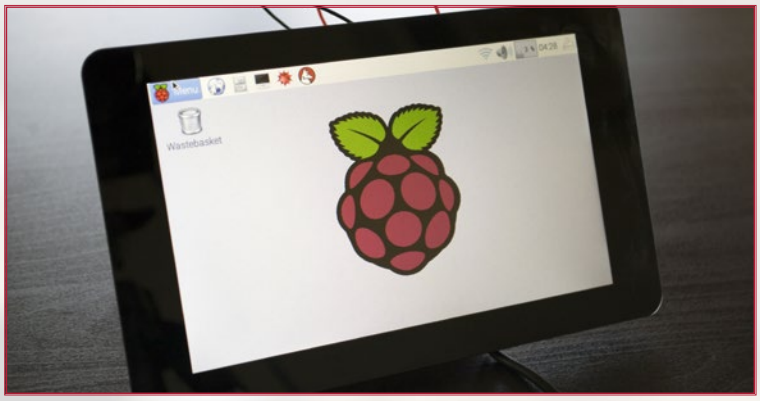
As a bonus, if you happen to break one, they're not going to cost you a fortune to replace.

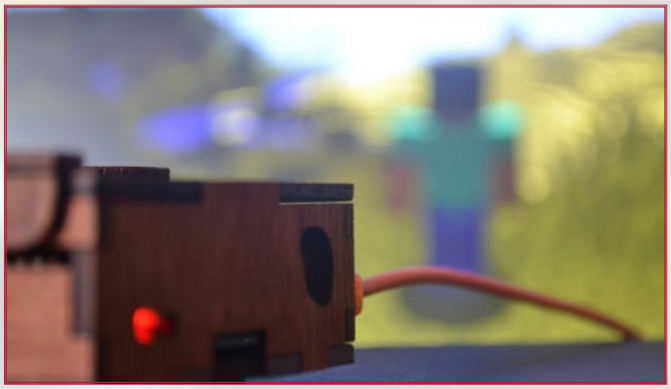


They are also great at doing a lot of things that you don't want to have a full tower computer running all the time, such as to run a home NAS (Network Attached Storage), web server, media center, TorrentBox and much more.

The primary operating system for the Raspberry Pi is Raspbian and which is based on the popular Debian operating system.

It is a distribution of Linux so you will probably find it a little different if you are used to a Windows or Mac-based computer.





Even though the primary supported operating system is Raspbian, you can install other operating systems such as Ubuntu Mate, Ubuntu Core, OSMC, RIS OS, Windows 10 IoT and much more.
For our purposes, though, we will be sticking to Raspbian as it offers the best support for the functionality of the Raspberry Pi.

There are four different models of the Raspberry Pi that you will currently find available from stores. We will go through the various features of each of the devices and why you might choose a particular model over another. We will quickly go through these over the next couple of pages.

Raspberry Pi A

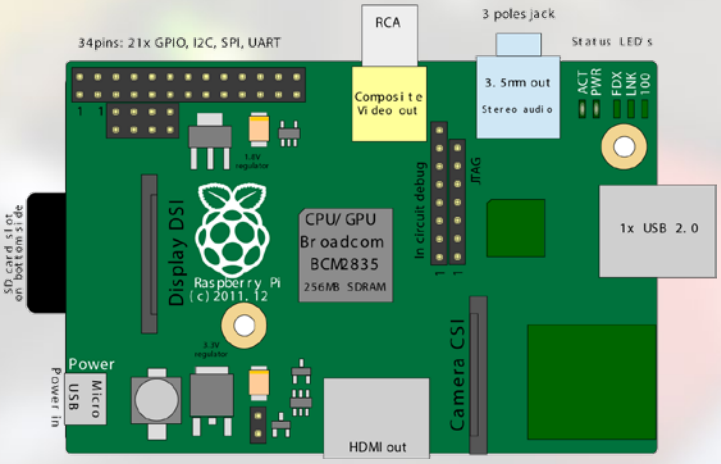
The Raspberry Pi A is the low spec and low-cost version of the original Raspberry Pi B. It has now primarily been replaced by the incredibly cheap Raspberry Pi Zero as the best choice for embedded projects.

You will find that this version of the Raspberry Pi only has one USB port, lower power consumption, no Ethernet port and just comes with 256mb of ram.

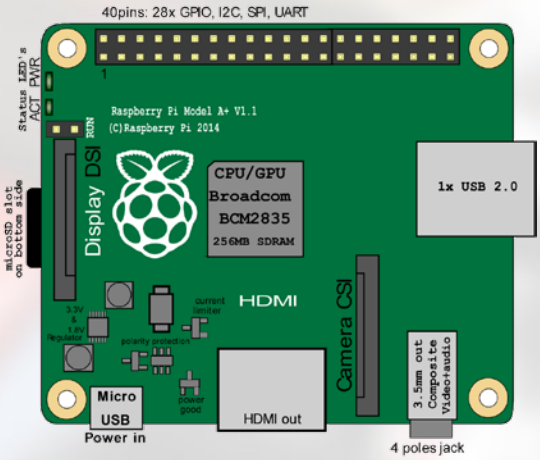
This version of the Raspberry Pi is better suited for projects that don't require any massive amount of processing power and will benefit from the lighter weight/smaller size.

Projects such as robotics, remote control planes/cars, and embedded applications are perfect for this Raspberry Pi. To just name a few examples it would make for a great robot brain, touchscreen car dashboard, motion-sensing camera and much more.

Raspberry Pi A Diagrams



Connectors and main ICs on Raspberry Pi 1 Model A



Connectors and main ICs on Raspberry Pi 1 Model A+ Revision 1.1

Raspberry Pi B+

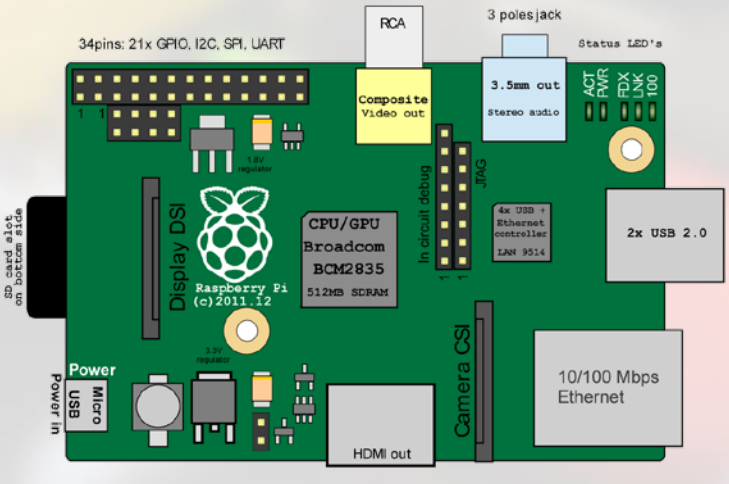
The Raspberry Pi B+ and Raspberry Pi B was the more powerful cousin of the Raspberry Pi A+, sporting 512mb of RAM, more USB Ports, and an Ethernet port.

The B+ version features a single core CPU, 4 USB ports, microSD card slot, lower power consumption and an expansion of its GPIO pins from only 20 pins to 40 pins, increasing the overall versatility of the device.

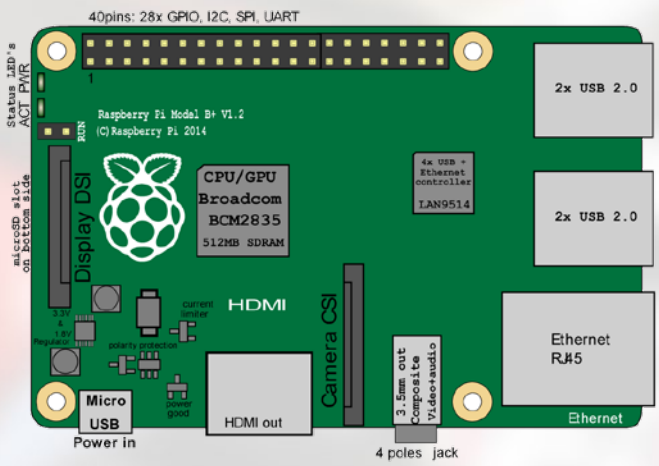
The Raspberry Pi B+ improves dramatically on the base model Raspberry Pi B that only had the 2 USB ports, higher power consumption and required a full-sized SD Card slot.

The Raspberry Pi B+ and Raspberry Pi B have now been replaced with the much-improved Raspberry Pi 2 and the far more feature inclusive Raspberry Pi 3.

Raspberry Pi B+ Diagrams



Connectors and main ICs on Raspberry Pi 1 Model B



Connectors and main ICs on Raspberry Pi 1 Model B+

Raspberry Pi 2

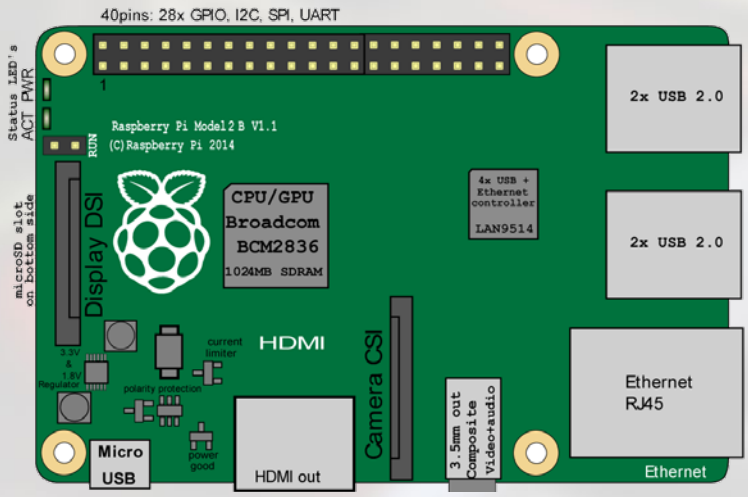
The Raspberry Pi 2 was the successor to the Raspberry Pi B and B+. It is quite like the Raspberry Pi B+, sporting similar base functionality, and the same general hardware layout.

It did, however, provide a giant step forward in the processing power of the Raspberry Pi. The Raspberry Pi team achieved this by swapping out the weak single-core ARMv6 processor with a much more powerful quad-core ARMv7 processor that is clocked at 900 MHz

The Raspberry Pi 2 also features double the amount of RAM than its predecessor. Ultimately this allows the Raspberry Pi to handle much more resource intensive tasks that the original one would struggle with.

However, this came at the cost of increased power usage, making it less useful for portable embedded projects. This problem was eventually solved by the Raspberry Pi Zero.

Raspberry Pi 2 Diagram



Connectors and main ICs on Raspberry Pi 2

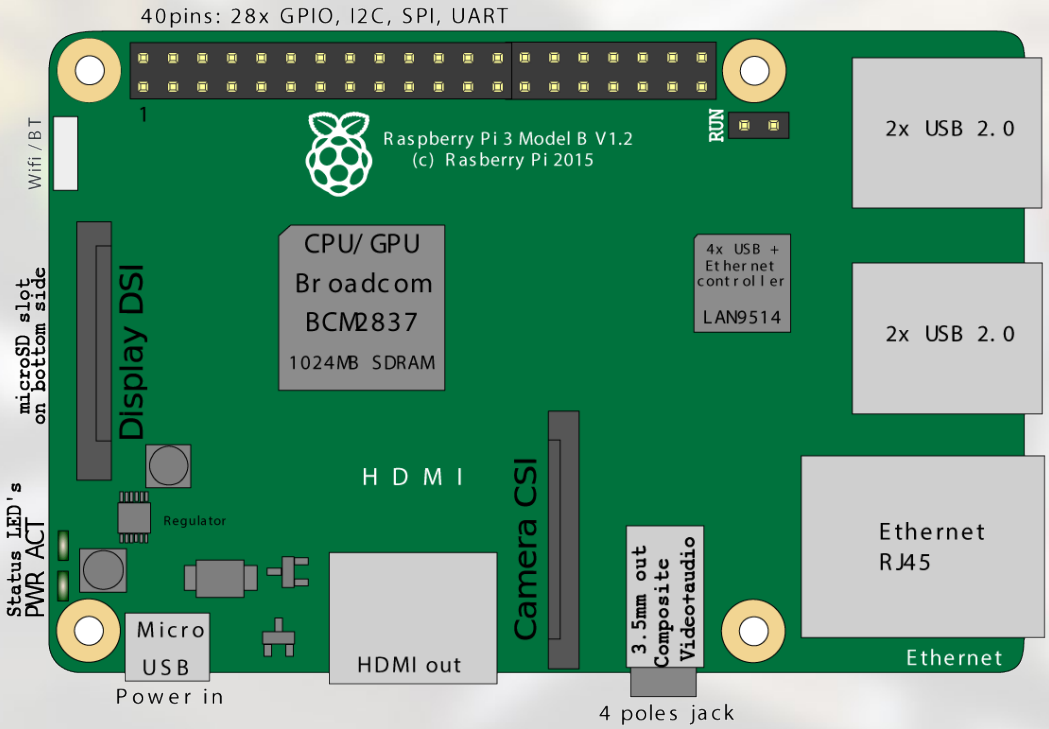
Raspberry Pi 3B

The Raspberry Pi 3 is an improved version of the Raspberry Pi 2 besting it in both performance and features. This release offers a few extras that make using a Pi so much easier to use.

This version of the Raspberry Pi brings with it a new CPU that clocks in at 1.2 GHz and technically has a 64bit capability. However due to the way it is being utilized in the Raspberry Pi it will only correctly function as a 32bit processor.

The Pi also has onboard Wi-Fi (802.11n) and Bluetooth 4.1. The addition of the built-in WiFi module will enable you to do some cool stuff while freeing up more of your USB ports.

Raspberry Pi 3B Diagram



Connectors and main ICs on Raspberry Pi 3B

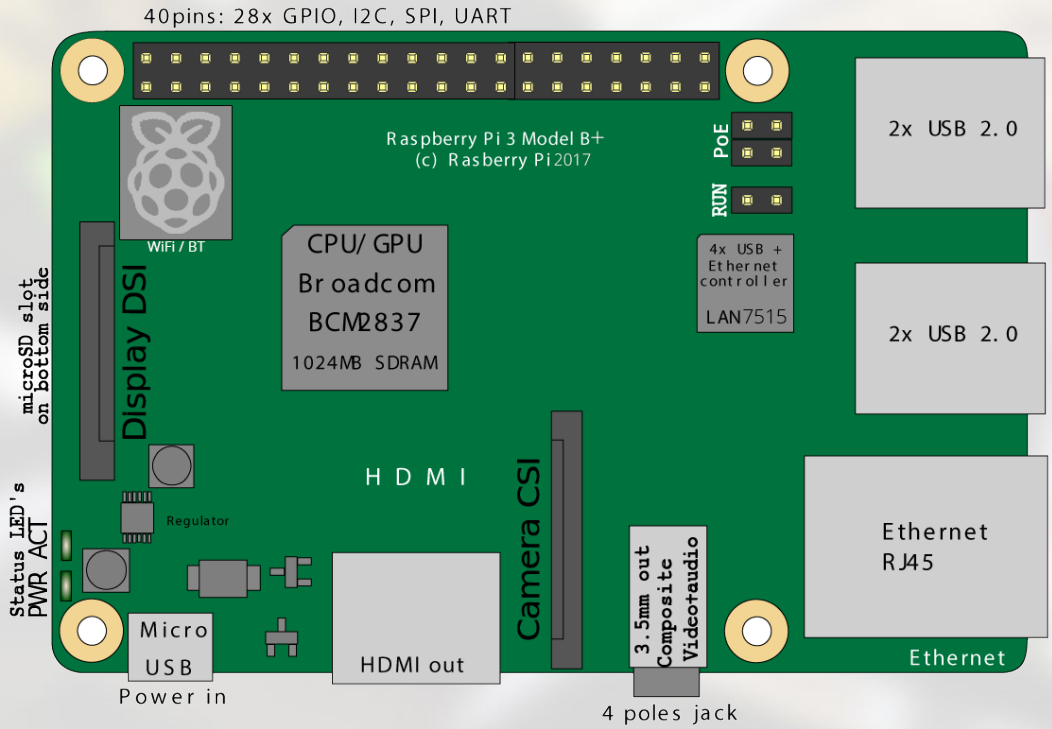
Raspberry Pi 3B+

The Raspberry Pi 3B+ is a refined version of the Raspberry Pi 3. While this is not a massive jump in power or capabilities, it does offer numerous improvements over its predecessor.

The most note worthy improvement is the replacement of the WiFi chip with one that is capable of dual-band wireless LAN and also features compatiblitiy with the 802.11ac standard.

In addition to the improvements to the WiFi, there ethernet connection is now capable of 300 Mbps, with the added ability to also use PoE, however you require an additional HAT to actually power your Raspberry Pi with it.

Raspberry Pi 3B+ Diagram



Connectors and main ICs on Raspberry Pi 3B+

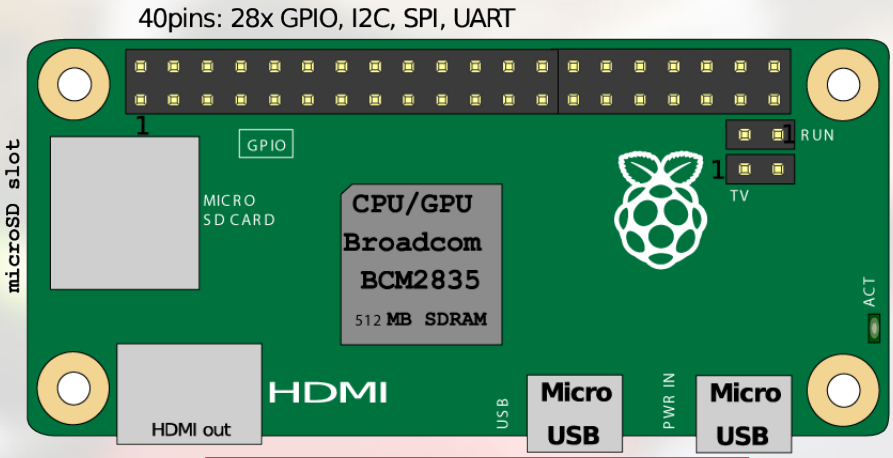
Raspberry Pi Zero

The Raspberry Pi Zero is the smallest and cheapest of all the Raspberry Pi's currently available. It is designed as a more powerful and more energy efficient successor to the small Raspberry Pi A+ and an alternative to the more feature complete Raspberry Pi 3.

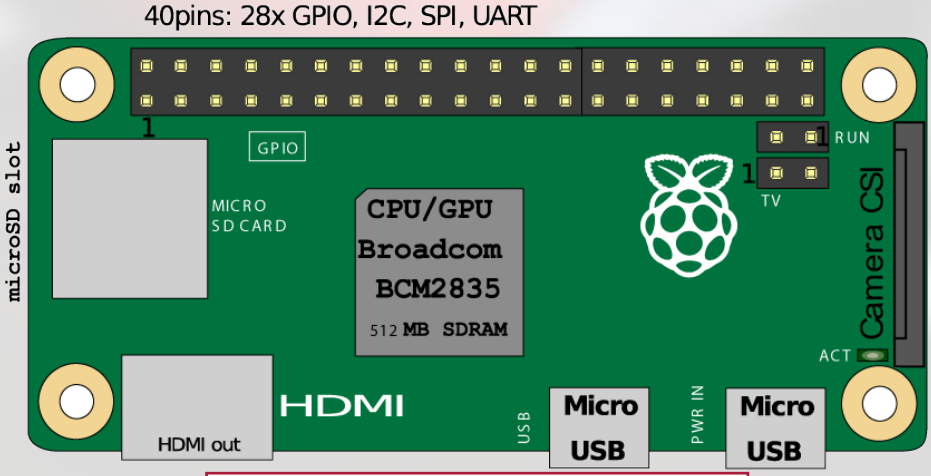
This board is priced at $5 USD and features the barebones of the Pi on a super small board.

There is currently two revisions of the Raspberry Pi Zero board, revision 1.0 and revision 1.3. The 1.3 change brought with it the addition of a CSI connector for the Camera module.

Raspberry Pi Zero Diagram



Connectors and main ICs on Raspberry Pi Zero (Rev 1)



Connectors and main ICs on Raspberry Pi Zero (Rev 1.3)

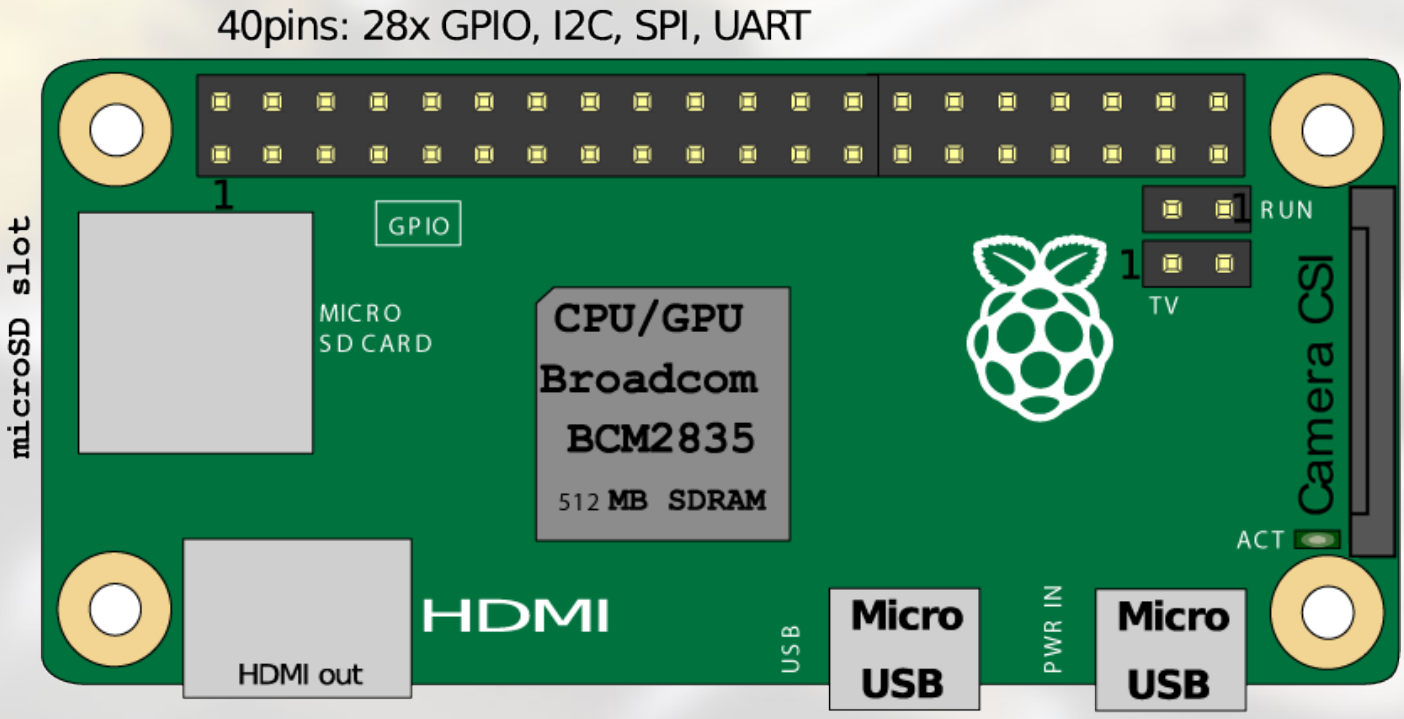
Raspberry Pi Zero W

The Raspberry Pi Zero W is an upgraded version of the Raspberry Pi Zero. There are no real drastic changes in the Raspberry Pi Zero W except for the addition of Wi-Fi and Bluetooth support on the board.

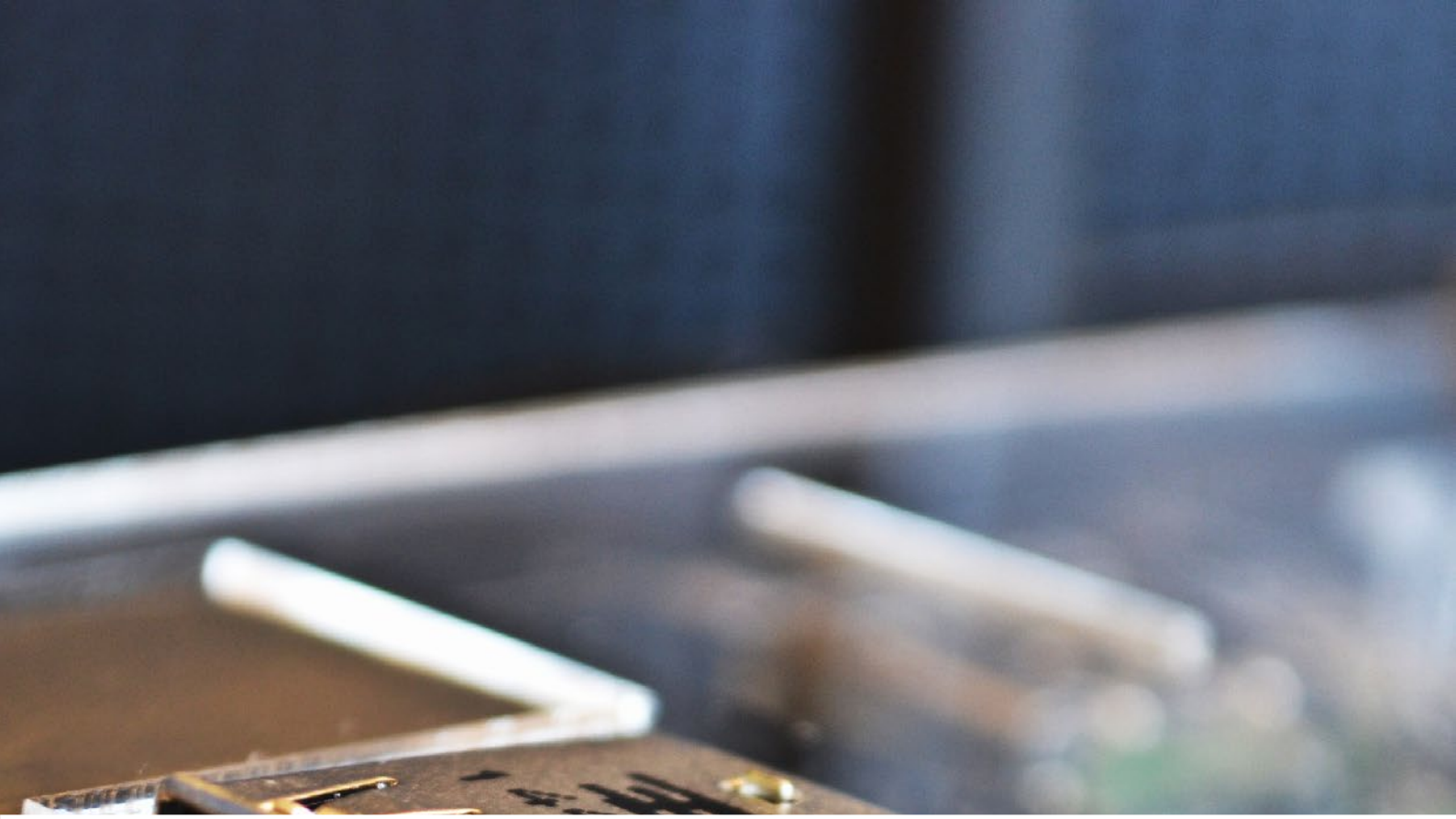
This board is priced at $10 USD which is a small price increase when compared to the original Zero board. However, the addition of both Wi-Fi and Bluetooth support is a massive improvement given the lack of external connections available for the board.

There is no performance difference between the Raspberry Pi Zero and the Raspberry Pi Zero W.

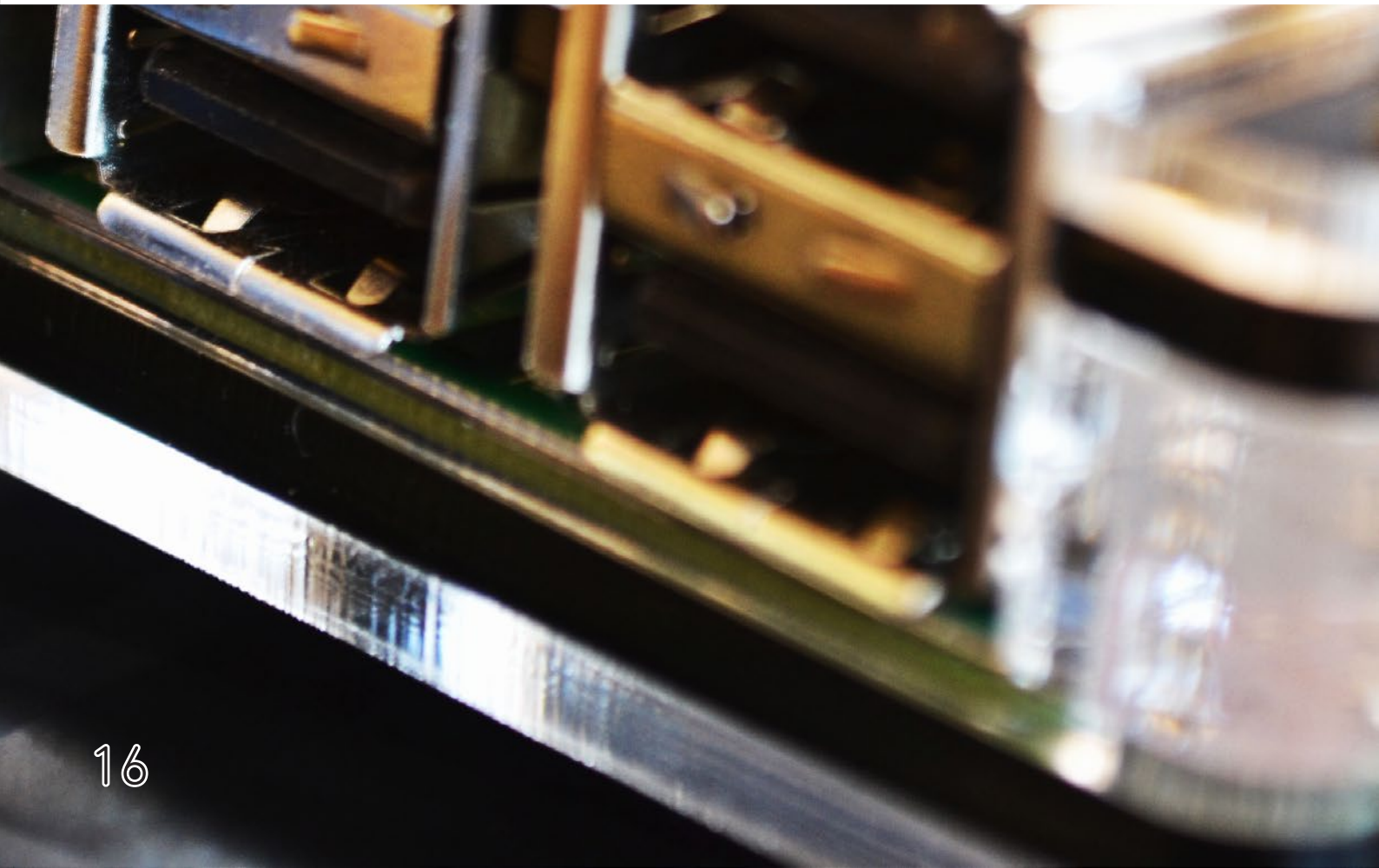
Raspberry Pi Zero W Diagram



Connectors and main ICs on Raspberry Pi Zero



Getting Started
With The Raspberry Pi



This section is for all those who are just starting out with the Raspberry Pi. This guide will take you through all the initial steps to get your Raspberry Pi up and going.

Along with giving you an idea of how to use the basic functionality of the Raspberry Pi so you can better grasp the following tutorials and projects.

I will first quickly go through all the equipment that you will need if you're just starting out. You might have some of these parts already so you can save a bit of money and reuse the parts for the Raspberry Pi.

Essential Equipment

Below is all the gear that you will need to make the most out of your Raspberry Pi.

Raspberry Pi

You can't start without one of these! There are quite a few variations of the Raspberry Pi so be sure to choose whichever best suits your needs.

SD Card

Necessary for the Pi to operate, it is recommended that the SD card is an 8 GB SD Card (Especially if you're installing the latest version of Raspbian) If you have a Pi B+, 2 or 3, then you will need a micro SD card.

Keyboard & Mouse

Required if you're controlling the Pi directly. Can also be incredibly handy when there is a problem that causes your Raspberry Pi to lose internet access, or the SSH package gets corrupted.

HDMI Cord

You will need one of these to be able to hook the Pi up to a screen. An HDMI cord is necessary if you will be controlling and using the Pi directly rather than setting it up to be headless.

Power Supply

A power supply is essential if you would like the Pi to turn on. A 5V, 2A micro USB power supply is recommended for the Raspberry Pi.

It is essential that it outputs at least 5V, 2A otherwise the Pi may not turn on or behave bizarrely.

If you get a rainbow colored square in the top right corner, then you do not have a sufficient enough power supply connected and should be replaced if you want your Pi to be stable.

Optional Equipment

Below we will be detailing various pieces of equipment that make using your Raspberry Pi and developing different projects much more accessible. Some of these recommendations also have the added benefit of reducing the difficulty of some of the more complicated projects in this book.

An Internet Connection

A WiFi adapter or Ethernet cord will be required to connect to the internet. If you have the Raspberry Pi 3, then you won't need a Wi-Fi dongle as this is onboard.

The clear majority of the tutorials in this book will require an internet connection. This internet connection is so that you can install packages that are not found in the default Raspbian installation.

A case for your Raspberry Pi

While not essential it is recommended that you use a case as it will help protect your Raspberry Pi from various outside interference. It also has the advantage of giving your Pi a sleeker and more custom appearance.

You will notice some cases even add some extra functionality to the Raspberry Pi.

GPIO Breakout Kit

A GPIO Breakout kit is a beneficial tool for the Raspberry Pi. While it's not a necessity, it helps significantly in quickly and efficiently accessing the GPIO pins on the Raspberry Pi.

It will make your life a lot easier when it comes to tackling some of the projects that require wiring modules to the GPIO pins.

Heatsinks

Heatsinks are probably the most optional component in this list. Heatsinks are designed to improve the amount of heat that can be removed from an individual object. For the Raspberry Pi, the critical place to try and reduce heat from is the processor.

The Raspberry Pi 3 and Raspberry Pi 2 while offering a more significant amount of processing power, also generate a lot more heat. A heatsink allows the Raspberry Pi to dissipate that heat better away, ensuring that your Raspberry Pi will remain stable under increased load.

Powered USB Hub

Powered USB Hubs are a useful component to have with your Raspberry Pis. Especially if you intend to connect several high-powered USB devices such as portable hard drives.

While the newer Raspberry Pi's all feature 4 USB ports, they still suffer a problem, and that is the low power draw from the Raspberry Pi itself.

If you connect too many high-powered devices, the Pi will not have enough power left to run itself, and you run the risk of damaging the equipment due to the higher power drain.

A powered USB hub solves this problem by utilizing mains electricity to power all the connected devices, leaving your Pi to just to handle the data connections.

Assembly

If you have purchased a case for your Raspberry Pi, then this is a good time to assemble the case.

Since all cases are different simply assemble according to the instructions that hopefully were supplied with the case. Most cases tend just to clip together.

Once you have assembled the case, you are then ready to plug everything in and get your Pi working.

Plugging in your Raspberry Pi

Before you go ahead and turn on your Raspberry Pi make sure you have all the required equipment that we listed on page 17 of this book.

If you don't have some of the equipment, you may find the Pi will not function or will be impossible to control.

1. Start by inserting your SD card into the slot on the Raspberry Pi, this is located on the bottom of the Pi, and the card will only fit one way.

2. Next plug in the keyboard & mouse into the USB slots. These will allow you to interact with the Raspberry Pi.

3. Next, connect the HDMI cord to the HDMI port on the Pi and then to the TV or monitor. It's important to make sure you have selected the right input on your TV or monitor (E.g., HDMI 1).

4. If you want to connect your Pi to the internet you will need to insert an Ethernet cord or a WiFi adapter. If you have the Raspberry Pi 3, you can just utilize its inbuilt WiFi module.

5. When you are happy that you have plugged in all the cables and the SD card required, finally plug in the micro USB power supply. With the power supply connected your Raspberry Pi will turn on and automatically begin booting.

6. If this is your first Raspberry Pi and you are using a NOOBS SD card, then you must select an operating system to install and configure. If you need to install Raspbian manually, then don't worry as we will be going over how to do that next.

Installing NOOBS on the Raspberry Pi

What is NOOBS?

NOOBS is one of the easiest ways of installing an operating system onto the Raspberry Pi. NOOBS act as an installer for a variety of different operating systems and is designed to ensure that the setup is done right and is easy to understand for the average user.

If you have purchased a NOOBS (New Out Of the Box Software) SD card then installing the OS is a very straightforward process and you can just skip forward to the "**First time booting**" section on **page 20**.

However, if you didn't purchase a NOOBS SD card then don't stress because it is simple to download and install onto a blank SD Card / Micro SD Card, we will go through the steps to setup NOOBS on an empty SD Card now.

Step 1 - Downloading NOOBS

1. You will need to make sure you're using a computer with an SD card reader. If you don't have one, then you can buy a USB SD Card reader off the internet cheaply, and they should also be readily available at most electronic stores.

2. You will need to download the NOOBS installer (Offline & network install) You can find the NOOBS downloads at: **https://www.raspberrypi.org/downloads/noobs/**.

You will notice there is two options here, NOOBS and NOOBS Lite. We recommend grabbing the NOOBS version as it contains Raspbian within the zip archive.

Raspbian Lite just contains NOOBS, this means you will need to have an active internet connection on your Raspberry Pi to download an operating system over the network.

3. Once NOOBS has finished downloading, use an archive extracting tool such as 7Zip to extract the files from the zip to a folder. We will need these extracted files in the next step.

Step 1 - Formatting your SD Card

First thing you should do when installing a new image to an SD card is format it correctly.

The following steps will show you the best way to format your SD Card for the Raspberry Pi:

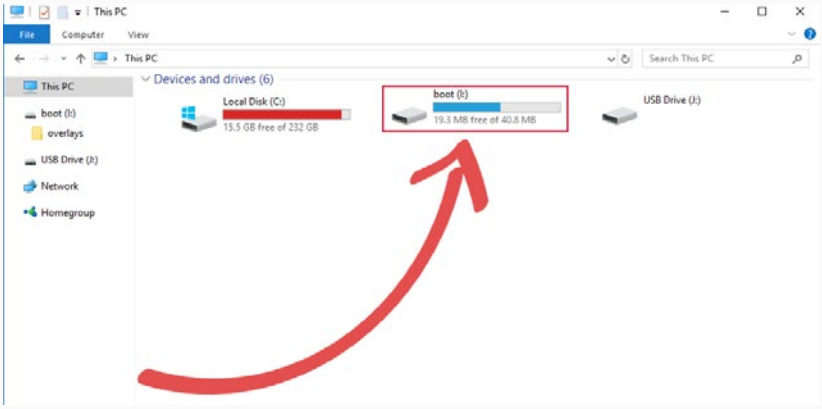
1. To begin you need a tool to format your SD Card. You can obtain one of the most robust ones from here: **https://www.sdcard.org/downloads/formatter_4/**

Once you are on the website will find a download link for the SD Formatter.

2. Download and install the SD Card Formatter from the SD Card association.

3. With the SD Formatter installed to your computer, insert your SD card and check the drive letter that is allocated to it, e.g., **G:/**.

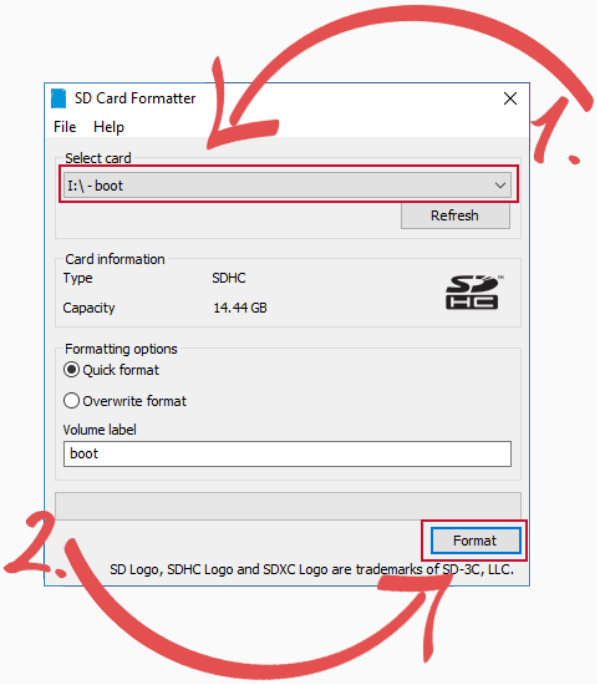
Make sure that you have the correct drive letter otherwise you could accidentally format the wrong device.



4. With your SD Card now plugged into your computer, launch the SD Formatter software that we installed earlier.

First set the drive letter to the correct one **(e.g., G:/)**.

Once you have everything set correctly, press the "**FORMAT**" button.



Step 3 - Copying files to SD Card

1. Once the SD card has been formatted, locate the place where you extracted the NOOBS installer files and copy and paste them onto the freshly formatted SD Card.

2. All the required files will then be transferred to your SD card. Pay attention to the transfer progress to make sure all the files are successfully copied over.

3. When the transfer has finished, safely remove the SD card, and insert it into your Raspberry Pi. We can then proceed with finally setting up the operating system itself.

Step 4 - First time booting your Raspberry Pi

1. Make sure that your keyboard, mouse, and monitor cables are all plugged into your Raspberry Pi. You will need them to correctly setup your device initially, afterward you will be able to connect using SSH.

2. Now once you are sure that you have connected the keyboard, mouse, and HDMI cable, plug in the USB power cable.

It's important to make sure the wires are connected correctly before plugging in the USB power cable as the Raspberry Pi does not handle hotplugging well.

It will also boot as soon as power is delivered to it, as there is no on and off button on the Raspberry Pi.

3. Your Raspberry Pi will boot, and a window will appear with a list of different operating systems that you can install.

Throughout the Raspberry Pi projects in this book, we will mostly be using **Raspbian** unless it specifies otherwise, ticks the box next to the OS you wish to install and then click install.

4. The installation process will now begin, this may take awhile.

5. Once the install process has completed, the Raspberry Pi configuration menu (**raspi-config**) will load.

Here you can set the time and date for your region and enable a Raspberry Pi camera board, or even create users.

We will be going into depth on the various features of the Raspberry Pi configuration menu on the next page

You can exit this menu by using Tab on your keyboard to move to Finish.

Step 5 - Logging into your Raspberry Pi

The Raspberry Pi default login for Raspbian is username pi with the password raspberry. Please note that as you type your password, nothing will be displayed. Not showing the password is a security feature within Linux based operating systems so someone watching cannot make out your password.

The default logins for the Raspbian operating system on the Raspberry Pi is as followed.
The default **username** is **pi** and the default **password** is **raspberry**

If Raspbian initially loads without any graphical interface, then to bring up the graphical user interface simply type **startx**.

Updating the Raspberry Pi

Updating packages on Raspbian

Updating packages on the Raspbian operating system is incredibly easy. It involves typing two straightforward commands into the terminal. It is good to remember the two commands that are shown below as they will become some of your most commonly used commands.

```
sudo apt-get update
sudo apt-get upgrade
```

The first of these commands (**sudo apt-get update**) makes a call to the **Advanced Packaging Tool** (**apt**) to update the package list, this is highly important as the **install** and **upgrade** commands only search the pre-grabbed package list and don't make any attempts to update it themselves.

The **update** command works this by searching the **/etc/apt/sources.list** file, and then polling all the websites in the list for all available packages creating a list of their download location and their current version.

The default location that Raspbian uses for its packages is http://mirrordirector.raspbian.org/raspbian/. This mirror director is designed to direct you to the closest download provider automatically.

Failing to run the **update** command before running **install** or **upgrade** could also cause you to run into download errors especially when packages are moved around on the mirror server.

The second command (**sudo apt-get upgrade**) again utilizes the **Advanced Packaging Tool** (**apt**). But this time it uses it to check all currently installed packages against the package list.

If there is a version miss-match for any, it will attempt to update it by downloading the new version from the link in the list. The upgrade tool will never remove a package.

Sometimes packages will be held back during an upgrade due to a change in the package's dependencies. To update the packages that are held back, we must use a different command. That command is shown below:

```
sudo apt-get dist-upgrade
```

This command (**sudo apt-get dist-upgrade**) again utilizes the **Advanced Packaging Tool** (**apt**) it starts off by performing an upgrade as usual, but when it comes to a package that has changed dependencies it intelligently handles them, this means it could also potentially remove packages in favor of others.

How to update Raspbian Wheezy to Jessie

In this segment, we will be showing you how to update Raspbian Wheezy to Jessie. There are a fair few steps to this process as a fair bit changed from Raspbian Wheezy to Jessie.

Two things to note before starting the upgrade process **backup your SD Card** just in case this fails for some reason.

You should also do this using a keyboard and mouse with **direct access to your Raspberry Pi** and **not** over SSH

1. To begin updating Raspbian, we need to first edit the **/etc/apt/sources.list** file to point to Raspbian Jessie instead of Raspbian Wheezy.

We can begin doing this by running the following command in terminal:

```
sudo nano /etc/apt/sources.list
```

2. Within the **sources.list** file find and replace all occurrences of **Wheezy** with **Jessie**, as we have shown below.

Find

```
deb http://mirrordirector.raspbian.org/raspbian/ wheezy main contrib non-free rpi
```

Replace with

```
deb http://mirrordirector.raspbian.org/raspbian/ jessie main contrib non-free rpi
```

Once you have finished replacing Wheezy with Jessie, you can save and quit by pressing **Ctrl + X** then **Y** and finally pressing **Enter**.

3. Next we have to also modify the **/etc/apt/sources.list.d/raspi.list** file, this file acts as another way of adding entries to the **sources.list** file we modified in the previous step.

Run the following command to begin editing the file:

```
sudo nano /etc/apt/sources.list.d/raspi.list
```

4. Within the **raspi.list** file we need to modify the first entry changing **Wheezy** to **Jessie**.

We also need to add "**ui**" to the end of the line, just like we have shown in our example below.

Find

```
deb http://archive.raspberrypi.org/debian wheezy main
```

Replace with

```
deb http://archive.raspberrypi.org/debian jessie main ui
```

Once you have finished modifying the file you can save and quit by pressing **Ctrl + X** then **Y** and finally pressing **Enter**.

How to update Raspbian Wheezy to Jessie

5. Now before we begin the update process itself, we need to create a directory, this is a requirement for some of the new and updates packages that are provided in Raspbian Jessie.

Type the following command into the Raspberry Pi's terminal to create this directory.

```
mkdir /home/pi/.config/autostart
```

6. Now we can finally begin the update process. Please be prepared for this to take a couple of hours as it is quite an extensive process.

Run the following two commands on your Raspberry Pi to begin the update process:

```
sudo apt-get update
sudo apt-get dist-upgrade -y
```

Finals steps after upgrading to Raspbian Jessie

1. Once the update process has completed, you need to reboot your Raspberry Pi.

The easiest way to do this is to type the following command into the terminal.

```
sudo reboot
```

2. Upon rebooting you will see several messages about "**Calling CRDA to update world regulatory domain**".

There is no need to worry about these messages, just wait until they stop appearing before proceeding.

3. Once the reboot has completed, you can now log in to your Raspberry Pi using your Pi user.

If the GUI fails to automatically start up you can type the following command into terminal:

```
startx
```

Be prepared to have to wait several minutes for the desktop to launch. The wait is due to it requiring having to update files as it loads.

The screen will go black during this time, but just be patient and wait for it to finish.

4. Now we can move on to adding the new packages that were introduced with a fresh installation of Raspbian Jessie. You can just run the following command to install these packages.

```
sudo apt-get install rc-gui libreoffice libreoffice-gtk alacarte bluej greenfoot claws-mail
```

The packages that you are installing, in order of their name, are as follows. The new GUI version of the raspi-config command line tool, LibreOffice and its GTK extension, the Alacarte menu editor, both the BlueJ and Greenfoot Java IDE's, and finally we also install the ClawsMail email client.

If you are aiming for a slimmer installation, you can just install **rc-gui** and **alacarte**, or skip the installation altogether if you have no intention of using the GUI.

Finals steps after upgrading to Raspbian Jessie

5. Finally, we need to fix various issues with the upgrade to the newer Raspbian Jessie UI. To do this, we will be user numerous commands. We will explain what each command does underneath it.

To proceed type in each of the commands shown below:

```
cp –ax /usr/share/themes/PiX ~/.themes
```

This makes Raspbian load in the new version of the PiX GTK theme.

```
sudo rm /etc/xdg/autostart/clipit-startup.desktop
```

This command stops the ClipIt application from automatically starting up on boot.

```
sudo rm /etc/xdg/autostart/wicd-tray.desktop
```

This command will stop the Wicd network manager from starting up on boot automatically.

```
sudo rm –rf /var/lib/menu-xdg
```

This removes a massive list of shortcuts that are automatically placed in the "**Other**" menu, users don't need to deal with these.

```
sudo raspi-config nonint do_boot_behaviour_new B4
```

This tells your Raspberry Pi to boot to desktop and to automatically login as the default pi user.

```
sudo rm /usr/share/applications/obconf.desktop
```

This hides the menu shortcut for the Openbox Window Manager. We no longer need to link to this as its functionality has been replaced by the brand new Appearance Settings application.

You should now have a Raspberry Pi that has been successfully updated from Raspbian Wheezy to Jessie. With any luck, this would have happened with no issues.

How to update Raspbian Jessie to Stretch

Before we start the whole upgrade process make sure you **backup your SD Card**.

Use a keyboard and mouse with **direct access to your Raspberry Pi** and **not** SSH.

1. Before we begin switching our sources over to Stretch, we will first perform a full system upgrade to ensure we are running the latest available packages in Jessie.

To do this run the following two commands in the terminal:

```
sudo apt-get update
sudo apt-get upgrade -y
```

2. In some cases, packages may be held back, and this is the last thing we want when upgrading the system to Stretch as it may cause some serious issues down the track.

Let's enforce upgrading to the held back packages by running the following command in the terminal on the Raspberry Pi.

```
sudo apt-get dist-upgrade -y
```

3. Now the final update we need to do before beginning the process of updating to Raspbian Stretch is to do a firmware update.

To ensure we have the latest firmware, we can run the following command:

```
sudo rpi-update
```

4. Now to get Raspbian to update to stretch we need to modify the **/etc/apt/sources.list** file so it points to Raspbian Stretch instead of Jessie.

Begin editing this file by running the following command in terminal:

```
sudo nano /etc/apt/sources.list
```

5. Now that we have done the initial updates we can now modify the **sources.list** file and replace all occurrences of **Jessie** with **Stretch**, as we have shown below.

This command makes Raspbian search the new Stretch repository for packages.

Find

```
deb http://mirrordirector.raspbian.org/raspbian/ jessie main contrib non-free rpi
```

Replace with

```
deb http://mirrordirector.raspbian.org/raspbian/ stretch main contrib non-free rpi
```

Once finished you can save and quit by pressing **Ctrl + X** then **Y** and finally pressing **Enter**.

6. With that done we now need to modify the **/etc/apt/sources.list.d/raspi.list** file, this file adds additional sources to our **sources.list** file that we made modfiied to in the previous step.

Run the following command to begin editing the file:

```
sudo nano /etc/apt/sources.list.d/raspi.list
```

How to update Raspbian Jessie to Stretch

7. Once we begin editing the **raspi.list** file we need to change the first entry by replacing **Jessie** with **Stretch**.

Find

```
deb http://archive.raspberrypi.org/debian jessie main ui
```

Replace with

```
deb http://archive.raspberrypi.org/debian stretch main ui
```

Once you have finished switching **Jessie** out for **Stretch**, you can save and quit by pressing **Ctrl + X** then **Y** and finally pressing **Enter**.

8. To ensure we have a fast and smooth updating process, we will also need to remove the package called **apt-listchanges**.

The reason for removing this package is to stop it from trying to load the changelog for the massive number of packages that are going to be upgraded.

To remove this package, we need to run the following command in the terminal on the Raspberry Pi.

```
sudo apt-get remove apt-listchanges
```

9. Now that we have finished updating both the **raspi.list** file and the **sources.list** file we can proceed with updating Raspbian to Stretch.

Run the following two commands on your Raspberry Pi to begin the update process.

Please be prepared for this to take several hours as it is a rather extensive process.

```
sudo apt-get update
sudo apt-get dist-upgrade -y
```

You will have to pay attention during this installation process as you will be required to input '**Y**' and press **Enter** for the upgrade process to continue onwards.

10. With our Raspberry Pi now successfully updated to Raspbian Stretch there are a few more things we will want to do.

During installation, many packages will be marked as no longer needed due to changed dependencies.

To remove these, we can just type in the following command into the terminal on the Raspberry Pi:

```
sudo apt-get autoremove -y
```

How to update Raspbian Jessie to Stretch

11. After running the autoremove command, we should also clean out the package cache. The command that we will be using is autoclean.

It will automatically remove any packages files that are no longer able to be downloaded and are mostly useless.

Run the following command in your Raspberry Pi's Terminal to automatically clean these packages up.

```
sudo apt-get autoclean
```

12. Finally, the last thing we should do is reboot the Raspberry Pi to ensure it loads in all the new packages and services correctly.

Just run the following command in terminal to restart your Raspberry Pi.

```
sudo reboot
```

You should now hopefully have a Raspberry Pi that has been successfully updated from Raspbian Jessie to Stretch.


HDMI
AV

Backing Up Your Raspberry Pi

Backing up your Raspberry Pi

In this guide, we will be showing you various ways of backing up your Raspberry Pi and restoring it. Backing up your Raspberry Pi is a crucial task that you should often be doing, especially if you make a lot of changes or are storing data on it.

One thing to note is that some backup methods are going to be way more thorough the others, for instance backing up your Raspberry Pi SD Card image is going to be more reliable than just backing up all the files to a USB device since the image is a replication of all partitions on the SD Card.

We will be exploring two different methods of backing up your Raspberry Pi in this guide, the two different methods that we will be showing you how to do is the following:

- Backing up your Raspberry Pi to a computer using an SD Card Reader
- Backing up the Raspberry Pi to a USB device

Personally, we recommend backing up your Raspberry Pi to a computer as being the most robust method.

Mainly since the Raspberry Pi doesn't have the best USB speeds or network speeds so creating, backups can take considerable time and put a fair bit of stress on the Raspberry Pi.

Backing up your Raspberry Pi SD Card

To start off, we are first going to show you how to backup your Raspberry Pi SD Card as an image. To do this tutorial, you will need to have an SD Card reader handy.

To begin this tutorial, please first turn off your Raspberry Pi by running the following command on your Pi's terminal:

```
sudo shutdown now
```

Once the Raspberry Pi has powered down, disconnect the power and remove your SD Card. Place the SD Card into your SD Card reader and proceed onwards with this section of the guide.

This section of the guide will be split into three parts, one for Windows, one for Mac OS X and one for Linux based systems.

Backing up your Raspberry Pi to a SD Card - On Windows

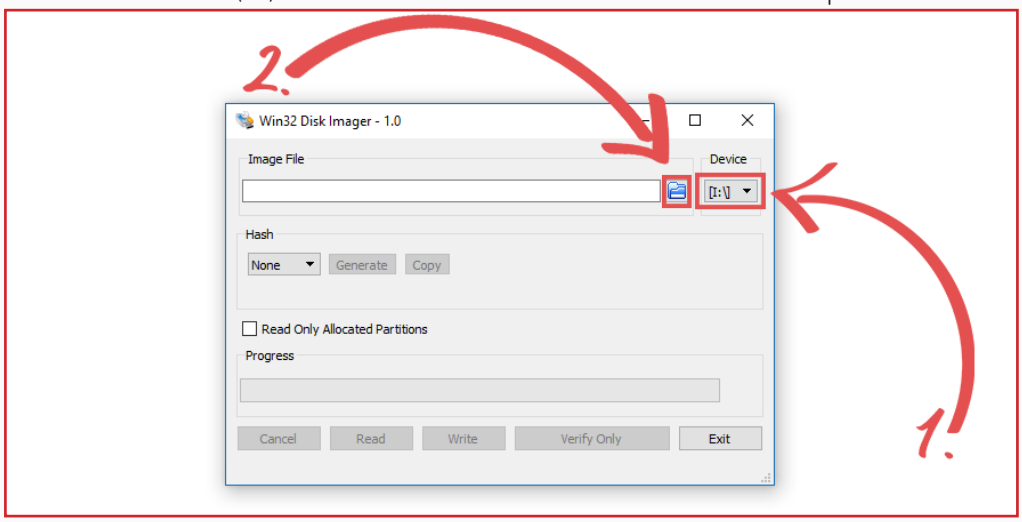
1. To backup our Raspberry Pi's SD Card on Windows we will be utilizing the imaging tool that is called **win32diskimager**. Win32diskimager is an incredibly useful tool that can read and write images to USB Sticks or SD/CF Cards.

We will start off this section of the guide by downloading and installing this tool by going to their web-site here: https://sourceforge.net/projects/win32diskimager/

2. With win32diskimager now installed, open it up. You should be greeted with a screen as we have shown below.

For now, we need to make sure the right drive has been selected by clicking the **drop-down box** under "**Device:**" (**1.**) and ensuring the correct drive letter of our SD Card is selected.

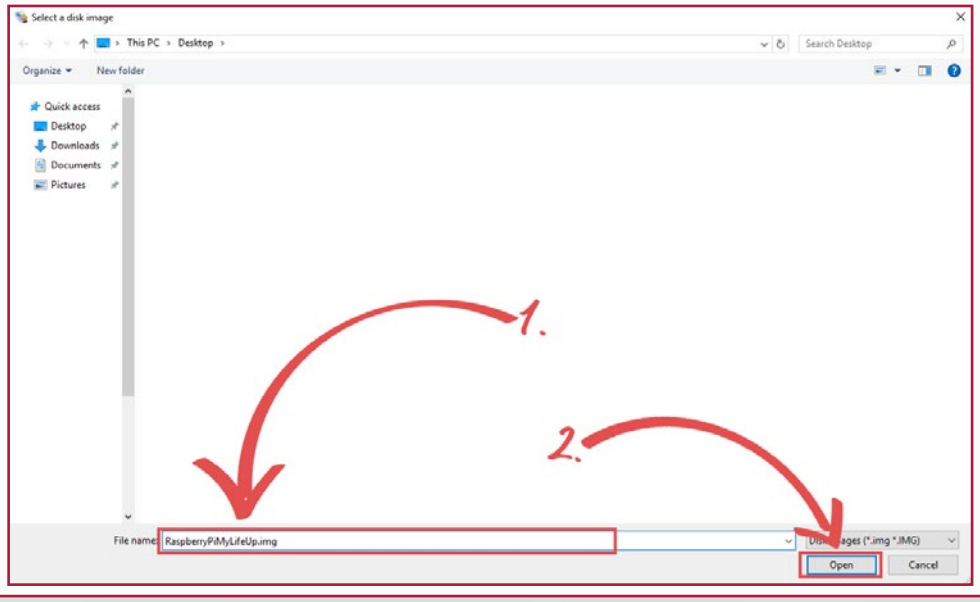
Afterward, **click** the **folder icon** (**2.**) that is direct to the left of the device drop-down box.



3. You will now be shown a file selection screen. However, we won't be selecting any file. Instead, navigate to the folder you want to keep your backup file. Once within this folder, type in the name you want to give your backup file in the **textfield** (**1.**) at the bottom as shown in the screenshot below.

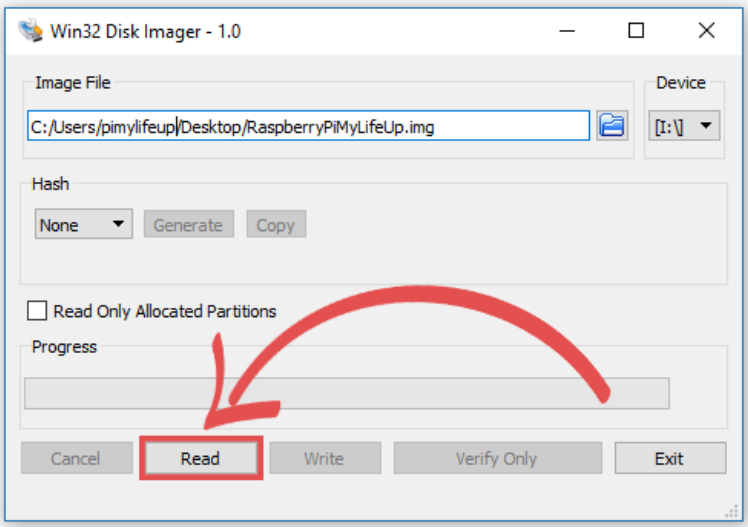
Make sure you have **.img** at the end of the name you choose to use. For example, we named our back-up file **RaspberryPiMyLifeUp.img**

Once you have typed in your file name, you want to utilize, **click** the **Open** button.



Backing up your Raspberry Pi to a SD Card On Windows

4. Finally with our device set and our new file name set we can begin the backup process. To do this, we just just need to **click** the "**Read**" button. Please note that this can take some time as it is a complete backup of your SD Card, meaning every single byte is replicated.

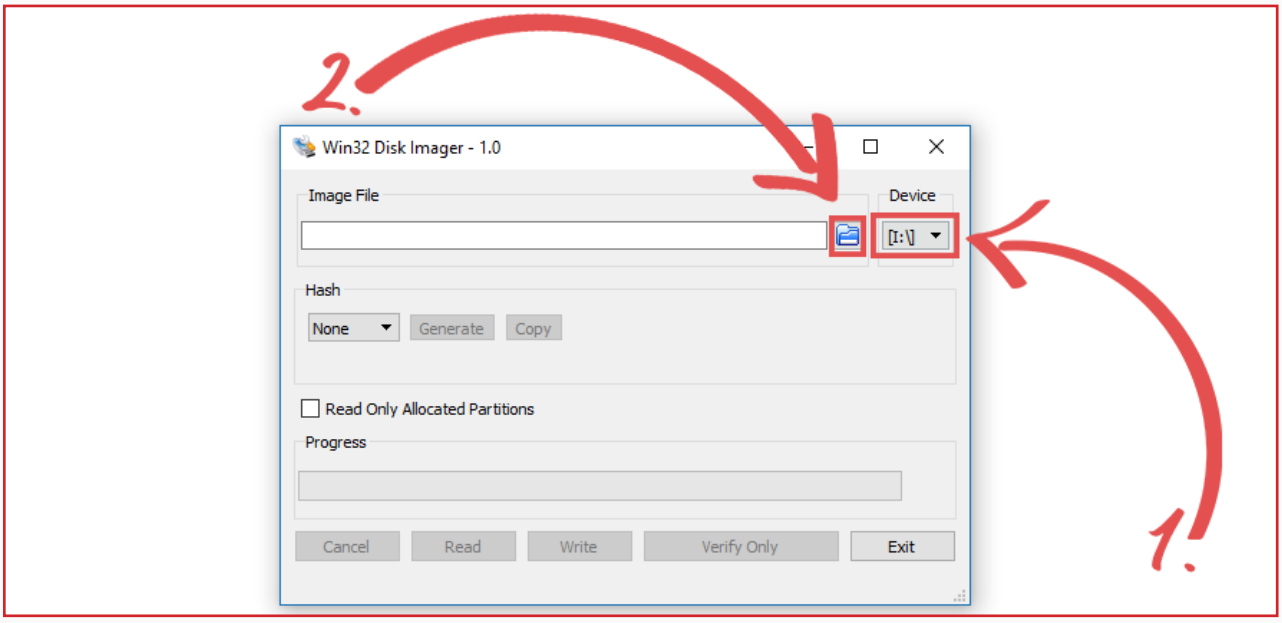


Restoring your Raspberry Pi Backup on Windows

1. Now when it comes to the time that you need to make use of your full SD Card backup, we will need to make use of **win32diskimager** once again.

With the **win32diskimager** software now opened, let's click the **drop-down box** that is located under the "**Device**" (**1.**) header and select the correct drive letter of our SD Card.

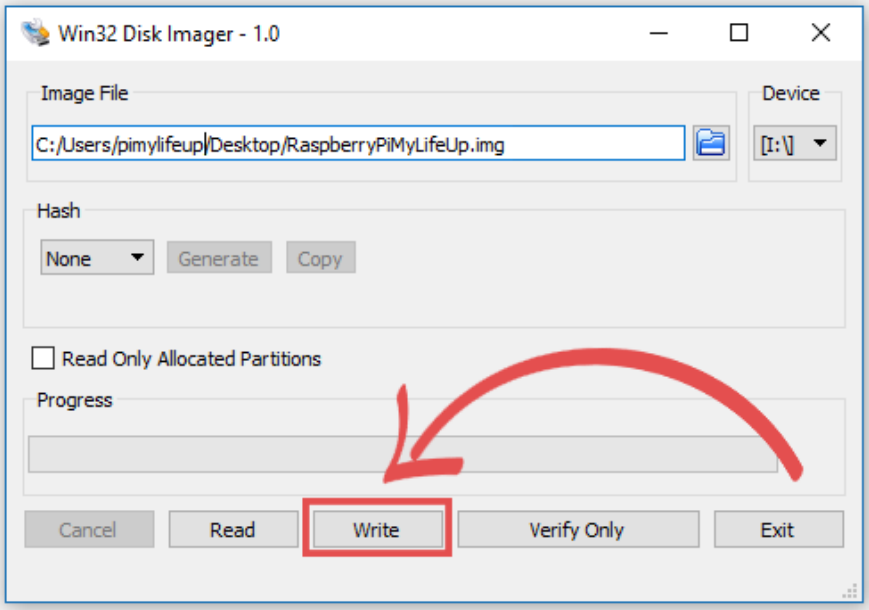
After you have set the device, we can now, **click** the **folder icon** (**2.**) that is located to the left of the drop-down box.



2. Now that we have the file browser loaded up let's find the backup file that we created earlier. Once you have found it either, **double-click** the file, or **single click** it and **click** the "**Open**" button.

Restoring your Raspberry Pi Backup on Windows

3. With our backup image now selected, we can now proceed with writing the backup file to the SD Card. When you are happy you have everything right, **click** the "**Write**" button.



4. Once the writing process has completed, you should now have an SD Card that is in the exact state of when you initially made the backup. Thanks to the full image backup it means all partitions on the card are restored.

Backing up your Raspberry Pi to a SD Card - On OSX

1. With your SD Card inserted into a card reader on your Mac, we can begin the process of making a full image backup of your Raspberry Pi.

To proceed with this tutorial, start off by opening the Terminal application.

2. With the Terminal application now open on your Mac device, we need to utilize the following command. This command will display all available disks on your device.

```
diskutil list
```

Within this list, look for your SD Card by looking for a disk that is about the size of your SD Card.

For example, with a **16gb** SD Card, you should be looking at a "**_partition_scheme**" of about **16gb**. You will also notice that there is likely a partition called "**boot**."

Once you have found your SD Card in this list, take notice of the mount location. In our case, the SD Card was listed under **/dev/disk1**.

3. Now still within the terminal on your Mac, we need to utilize the following command.

This command will create a copy of your SD Cards image, and save it to your home directory as **PiSD-Backup.dmg** (The file format being a disc image)

Make sure you swap out **/dev/disk1** with whatever you found when using the **diskutil list** command.

```
sudo dd if=/dev/disk1 of=~/PiSDBackup.dmg
```

This command can take some time to complete as it requires reading the entire SD Card to the disk.

The command also provides no feedback on how far along it is, so please be patient and wait for the prompt to enter another command to reappear before removing your SD Card from your Mac.

Restoring your Raspberry Pi Backup on OS X

1. Now that you need to restore your Raspberry Pi to its backup we need to use the **diskutil list** command again to find our SD Card.

Remember to as before, take note of partition sizes to find your SD Card.

```
diskutil list
```

2. Now before we can write to the SD Card, we will need to unmount it.

The reason for this is that OSX will attempt to write to the SD Card at the same time, unmounting the SD Card prevents this from happening.

Run the following command on your Mac device to unmount the card, again making sure that you replace **/dev/disk1** with the location that you found using the **diskutil list** command earlier.

```
diskutil unmountDisk /dev/disk1
```

Restoring your Raspberry Pi Backup on OS X

3. Finally, we can now write the image back to the SD Card. Please be prepared for this to take some time as it involves rewriting the entire SD Card.

Remember to change out **/dev/disk1** with the mount location you grabbed using the **diskutil list** command.

```
sudo dd if=~/PiSDBackup.dmg of=/dev/disk1
```

Like reading the SD Card to a disk image file, the process of writing the image also takes a long time.

As an added note, the dd tool doesn't show any writing progress so please be patient and wait till the enter command prompt reappears.

4. Once the writing process has completed we can now eject the SD Card from the Mac so we can continue using our Raspberry Pi Backup. To remove the SD Card, we will need to utilize the command below.

Your Raspberry Pi's SD Card should now be in the same state as when you made the original backup.

```
sudo diskutil eject /dev/rdisk3
```

Backing up your Raspberry Pi to a SD Card on Linux

1. Before starting backing up your Raspberry Pi's SD Card on Linux, we will first run a command with the reader **not plugged in**.

The reason for this is that it makes it much easier to see which device is which.

```
df -h
```

This command will return something like what is shown below.

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       15G  4.0G  9.6G  29% /
devtmpfs       458M     0  458M   0% /dev
/dev/mmcblk0p1  41M   21M   21M  52% /boot
```

2. Now insert your SD Card reader back into your Linux computer and again rerun the following command, but this time take note of the additional entries.

```
df -h
```

This command will return something like what you got above, but with an additional entry, in our case **/dev/sda1** is that additional entry, with your entry, remove the partition number.

For instance, **/dev/sda1** will become **/dev/sda**. We need to do this as we want to write to the entire drive and not just a single partition on it.

In our example below we have highlighted the new entry that appeared on ours.

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       15G  4.0G  9.6G  29% /
devtmpfs       458M     0  458M   0% /dev
/dev/mmcblk0p1  41M   21M   21M  52% /boot
/dev/sda1      3.7G   75M  3.6G   3% /media/boot
```

3. We can now utilize dd, the same tool we make use of on Mac OS X since it is also built on Linux.

We can utilize the following command to begin dumping the SD Cards image to our home directory.

Make sure you replace **/dev/sda** with the filesystem that you grabbed using the **df -h** command.

```
sudo dd if=/dev/sda of=~/PiSDBackup.img
```

The process of backing up your Raspberry Pi can take some serious time, so be patient and wait.

The **dd** tool provides no feedback, so you will have to wait until the input command returns to your terminal.

Once it reappears, you will have successfully backed up your Raspberry Pi.

Restoring your Raspberry Pi Backup on Linux

1. Now that you have made a backup of your Raspberry Pi you will want to at some stage make use of this.

To do this, we will need to again go through the process of finding out the location of our filesystems.

Use the **df -h** command like we did in the first segment of this guide, though this time you might have more than one partition pop up for your SD Card such as **/dev/sda1** and **/dev/sda2**.

Take note of all new entries as you will need to unmount all of them.

```
df -h
```

2. Now that we have all our partition locations ready, we can unmount each of them by running the following command for each one.

Switch out **/dev/sda1** for the locations that you got in the previous step.

```
sudo unmount /dev/sda1
```

3. With all the partitions now unmounted let's write our backup image to the SD Card. We can do that by running the following command.

Remember to swap out **/dev/sda** with your devices mount location.

```
sudo dd bs=4M if=~/PiSDBackup.img of=/dev/sda
```

Backing up your Raspberry Pi to a USB

1. Before plugging in your USB device that you want to keep the backups on let's first run the following command to find out the current filesystems that are available to us.

```
df -h
```

This command will return something like what is shown below.

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       15G  4.0G  9.6G  29% /
devtmpfs       458M     0  458M   0% /dev
/dev/mmcblk0p1  41M   21M   21M  52% /boot
```

2. Now insert your USB Device into your Raspberry Pi and run the following command.

Take note of any new entries that pop up.

```
df -h
```

This command will return something like what is shown below.

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       15G  4.0G  9.6G  29% /
devtmpfs       458M     0  458M   0% /dev
/dev/mmcblk0p1  41M   21M   21M  52% /boot
/dev/sda1      3.7G   75M  3.6G   3% /media/pi/MYBACKUPDRIVE
```

3. With our USB device now showing on the list, we need to look at the "**Mounted on**" location. In our case, this is **/media/pi/MYBACKUPDRIVE**.

This path will be the location that we will write our backup images to. So, make a note of it for later in the tutorial.

Now with our backup location now handy we can download the backup script that we are going to use, this script was written by a user on the Raspberry Pi forums called Jinx.

Let's clone the script to our Raspberry Pi by running the following two commands.

```
cd ~
git clone https://github.com/lzkelley/bkup_rpimage.git
```

4. With the script now saved to the Raspberry Pi, we can start to make use of it. We can do an initial backup by running the command below on our Raspberry Pi.

Make sure you replace **/media/pi/MYBACKUPDRIVE** with the mount location that you grabbed in the previous step.

```
sudo sh bkup_rpimage.sh start -c /media/pi/MYBACKUPDRIVE/rpi_backup.img
```

This script will create a dummy image then launch a rsync process to copy all the files from the system to the dummy image.

Please note that the initial backup can take up to an hour to complete.

Backing up your Raspberry Pi to a USB

5. Now that we have created our initial backup file and know that the script is working as intended we can move onto automating the backup. To do this, we will be making use of cronjobs.

One thing to decide on how is whether you want an incremental backup or multiple backups. An incremental backup just updates the original backup and doesn't generate a new file.

Run the following command on your Raspberry Pi to begin editing the crontab.

```
sudo crontab -e
```

6. In the crontab editor, add one of the following lines to the bottom of the file. These additions make a backup at the start of every day.

If you want to edit the cron timings, you can use this website to help work out the correct values you need https://crontab.guru/.

Incremental Backup

```
0 0 * * * sudo sh ./home/pi/bkup_rpimage/bkup_rpimage.sh start -c /media/pi/MYBACKUPDRIVE/rpi_backup.img
```

Multiple Backup

```
0 0 * * * sudo sh ./home/pi/bkup_rpimage/bkup_rpimage.sh start -c /media/pi/MYBACKUPDRIVE/rpi_$(date +%Y-%m-%d).img.img
```

7. Now save the file by pressing **Ctrl + X** then pressing **Y** and then hitting **Enter**.

8. You should now have an automated backup system up and running that will continually backup your Raspberry Pi to your USB device.

To restore these images, follow our Restoring guides located in the SD Card section of this guide

Guide to the Raspberry Pi Configuration Tool

The Raspberry Pi configuration tool (**raspi-config**) can be incredibly daunting at first with its abundance of options and settings that it allows you to change a whole variety of the Raspberry Pi's functionality.

While for most people you will not need to change the default settings, some of the projects in here do require you to make changes.

In this guide, we are going to explain some of these options so that you will be able to understand better what each one does and then be able to make the appropriate changes when you are looking at a specific setting.

Before following this guide make sure that you have your copy of Raspbian up to date. If you are unsure on how to update your Raspberry Pi you can follow our guide that is featured earlier in this book.

Opening the Raspi Config tool

Opening the Raspi config tool is super easy and is the most important step. Opening the tool can be done by doing the following. (Raspi Config is launched automatically on first boot of Raspbian)

Double click on the LX terminal located on your Pi desktop (You don't have to do this if you're already in a terminal session)

Type the following command into the terminal:

```
sudo raspi-config
```

You will now be asked you for your password before it will proceed, if you haven't changed the password then it will be the default Raspbian **password**: **raspberry**

Once you have logged in as a root user, you will now be given access to the Raspi Config tool. To help you understand the abundance of options we will now go into the function of each item in the config tool over the next few pages.

You can navigate around the Raspi-config tool by using the **up and down arrow keys** along with making use of the **Enter key to go into the menu's** and the **escape key to exit out of them.**

Raspi Config Tool Settings rundown

1. Change User Password

The Raspbian default login details are openly published across the internet and are featured in the Raspbian. For this reason. it is essential that you change the password to keep your Raspberry Pi secure from unwanted access.

You can change the password for the default user **pi** by using this tool.
The default password on the Raspbian operating system is **raspberry**.

2. Network Options

This set of options allows you to control several different network related options such as the Wi-Fi settings and the Raspberry Pi's hostname.

N1. Hostname

This option allows you to change your Raspberry Pi's Hostname, the hostname is the visible identifier for your device on a network. By default this is just "**Raspberry Pi**".

N2. Wi-fi

This option allows you to easily setup WiFi on your Raspberry Pi by just specifying the SSID and the passkey to your WiFi.

N3. Network interfaces names

This option allows you to enable or disable the new predictable network interfaces names. This new system is designed so you can specify a specific network interface.

Previously there was no guarentee that wlan0 for instance would be the same device upon rebooting.

3. Boot Options

B1. Desktop / CLI

B1. Console

Boots the Raspberry Pi into console mode, this means there will be no GUI. This mode is useful for anyone who doesn't want the desktop GUI and would like to stick to terminal commands.

Console mode is not recommended for users who aren't familiar with navigating the system with the terminal commands. If you want to attempt this, check out our Linux command cheat sheet for a ton of commands that will make your life easier when purely using the terminal.

B2. Console Autologin

Console Autologin is the same as above, but the default **pi user** is automatically logged in at startup.

B3. Desktop

Tells the Raspberry Pi to auto boot into the desktop interface, this is the ideal option for most users who are not familiar with dealing with the terminal.

The desktop autologin mode is also the default option on a new installation.

You can still access the terminal from within the desktop GUI, there will be an icon on the GUI that will load up the terminal for easy to access.

B4. Desktop Autologin

Desktop autologin is the same as above, but the default **pi** is automatically logged in at startup.

B2. Wait for Network at Boot

This option causes the Raspberry Pi to wait for the network to initialize before it bothers continuing to boot. By default, Raspbian "Jessie" and newer boots as quickly as possible meaning that the network connection is likely not to be up and running by the time programs are loaded into memory.

Some programs can fail to launch correctly if they are opened before the network has started up. This option is a solution to that by making the system wait for the network to boot up and connect before it continues while delaying the boot it can be useful for solving problems with programs that start at boot.

B3. Splash Screen

The most recent versions of the Raspbian operating system includes a splash screen on boot up. The splash screen isn't necessarily something every user would like appearing.

While it shouldn't add any real extra load to the system and offers a prettier look on startup rather than the running command line, you can use this option to disable it.

4. Localisation Options

I1. Change Locale

In here you can change your locale, for example, en_au.UTF8 UTF8 or en_gb.UTF-8 UTF-8. Changing the local sets the Raspberry Pi to better support a language.

I2. Change Timezone

Use this option if you wish to set the Raspberry Pi time zone. Using this will allow you to update/set the time zone so that it is the correct time for your location.

To use this tool first select a region such as Australia or Europe for example. You will then need to select a city that is closest to your location.

I3. Change Keyboard Layout

This one can take a while to load all the keyboard layouts that are available. Once it has loaded, you're able to select the appropriate keyboard layout. Keep in mind UK is a different layout to a US layout keyboard.

I4. Change Wi-Fi Country

Changing the Wi-Fi country allows you to set the legal channels used in your Country for Wi-Fi. Setting the Wi-Fi country is useful for countries that have different legal ranges for Wi-Fi channels. Some countries allow people to connect to much higher channel ranges than others.

Raspi Config Tool Settings rundown

5. Interfacing Options

P1. Camera

You will need to enable this if you wish to use the Raspberry Pi camera module. To do this, you just need to select the option and then simply select the enable option. Enabling the camera option will also ensure that there is at least 128mb of RAM dedicated to the GPU.

Please note that this will reduce the amount of RAM that the CPU can access.

P2. SSH

This option allows you to disable/enable SSH access to your Raspberry Pi. (Command Line)

Enabling this will allow you to access your Pi from a remote location, if you don't plan on ever using your Raspberry Pi from a remote location, then it is best to keep it disabled as it can allow unwanted outside access to your Raspberry Pi if you leave it unsecured.

If you're also on planning on using it on a public network, ensure that you have changed all the passwords so that they're much stronger.

The last thing you will want is an outside person gaining control over the Raspberry Pi on your home network.

P3. VNC

This option allows you to disable/enable the VNC server on your Raspberry Pi.

VNC stands for Virtual Network Computing and is a graphical desktop sharing system that transmits mouse input and keyboard input over the network and then relaying the screen back to the connecting user.

It is useful if there are certain tools that you use on your Raspberry Pi that requires a graphical interface. If you only intend to utilize the terminal, you are better off sticking with SSH access.

P4. SPI

Allows you to disable or enable the SPI (Serial Peripheral Interface) kernel module which is needed for Raspberry Pi modules such as PiFace to work correctly.

It will allow you to connect a four-wire serial link so you can have sensors, memory and, various other peripherals. Enabling this option well enable them to talk directly with the Raspberry Pi.

P5. I2C

Allows you to enable the I2C kernel module so that you can connect I2C devices.

P6. Serial

Allows you to disable/enable shell and kernel messages from the serial connection.

P7. 1-Wire

This option allows the Raspberry Pi to support 1-Wire devices, such as the DS18B20 1-wire digital temperature sensor. This makes pin 7 (GPIO4) on the Pi to act as the 1-wire interface.

P8. Remote GPIO

This option allows you to decide whether the GPIO Server used in the Raspbian operating system to handle access to the GPIO pins should be accessible from the network or not.

Raspi Config Tool Settings rundown

6. Overclock

You're able to overclock your Raspberry Pi to get more power out of it. The default setting for this is to have the overclock disabled, meaning the CPU by default sits at its normal factory set clock rate, this varies between the different Raspberry Pi models.

This tool lets you override this default clock rate pushing the Raspberry Pi's processor harder then it was originally intended. However, this can provide an excellent performance boost but be prepared to deal with various issues that can pop up.

You should also note that overclocking may cause higher instability and shorten the lifespan of your Pi.

7. Advanced Options

These are the last lot of options in the Raspi Config tool and are a little more involved.

You probably don't need to alter these unless you're doing more outside the basics of the Raspberry Pi. Any tutorials that require changes in this part of the raspi-config tool will be mentioned in the tutorial.

A1. Expand Filesystem

If you installed Raspbian via the NOOBS setup, then you can safely ignore this option as NOOBS automatically extends the file system for you.

If you installed the Raspbian image straight to the Pi rather than using NOOBS, then you will be restricted to only the first 3GB of the SD Card.

Use this option to expand your installation to the entire SD card. This option will allow you to fit much more stuff onto the Pi/SD card.

A2. Overscan

If you're finding that your TV has a black border around it then you're suffering from an issue called underscan.

It's also possible that you are suffering from the opposite issue as well, this is where the visual output is overflowing off the screen. This issue is called overscan. It's a problem many older TV's suffer from due to the way they handle the incoming image.

Some TV's and screens do, however, let you change the way it handles the incoming image, so check that before enabling this option, as this option requires a fair bit of fine-tuning.

If you're suffering from underscan or overscan then enabling this option will help fix the display output.

You're able to edit this more in the config.txt located at the root of the device to get this, so the image is displaying correctly on your monitor/TV.

Raspi Config Tool Settings rundown

A3. Memory Split

This option allows you to alter the amount of memory that is made available to the GPU. If your tasks are just CPU intensive and make no real use of the GPU, feel free to increase the amount of RAM that is given to the CPU.

However, please note this can cause some side effects if you do use programs that utilize the GPU alot, so be careful with how you select the memory split.

In general, it is safer not to touch this option unless you know exactly what you are doing.

A4. Audio

This option allows you to force audio out of either the 3.5mm jack or the HDMI port. By default, this is set to auto, which means it will automatically determine which output to send sound through.

Unless you are having troubles with the automatic audio detection, there is no real need to modify this option.

A5. Resolution

This option allows you to set the screen resolution that the Raspberry Pi will start up at, this comes with a variety of different options. The default screen resolution is **720x480**

A6. Pixel Doubling

This option changes the way Raspbian handles pixels by mapping every pixel as a 2x2 block, EWffectively doubling the size of every pixel. This feature is useful for those who are using Raspbian alongside a monitor/screen that has very small pixels such as the MacBook Pro's retina display.

A7. GL Driver

This option allows you to switch on or off the experimental desktop GL driver for the Raspberry Pi. This option is kept off by default but can produce quite large speed improvements on the desktop and for applications that utilize OpenGL.

This GL driver allows the hardware on the Raspberry Pi to handle the rendering of OpenGL applications. The OpenGL driver is much faster than the software rendering that the Raspberry Pi relies on when this is disabled.

8. Update

Selecting this allows you to update the Raspi Config tool to the latest available version. They are continually updating this tool, so it is important to keep it updated as they often fix various issues.

9. About the Raspi Config

This option explains more about the Raspi Config tool. There isn't a huge amount to be said about this option besides that it gives you an idea of what version of the tool that you are running.

Setting up Wi-Fi

Project Description

In this guide we will be showing you all the steps to setting up Wi-Fi on your Raspberry Pi.

This guide works for both the inbuilt Wi-Fi module that comes with the Raspberry Pi 3 and the clear majority of the Wi-Fi dongles that you can get for the Raspberry Pis.

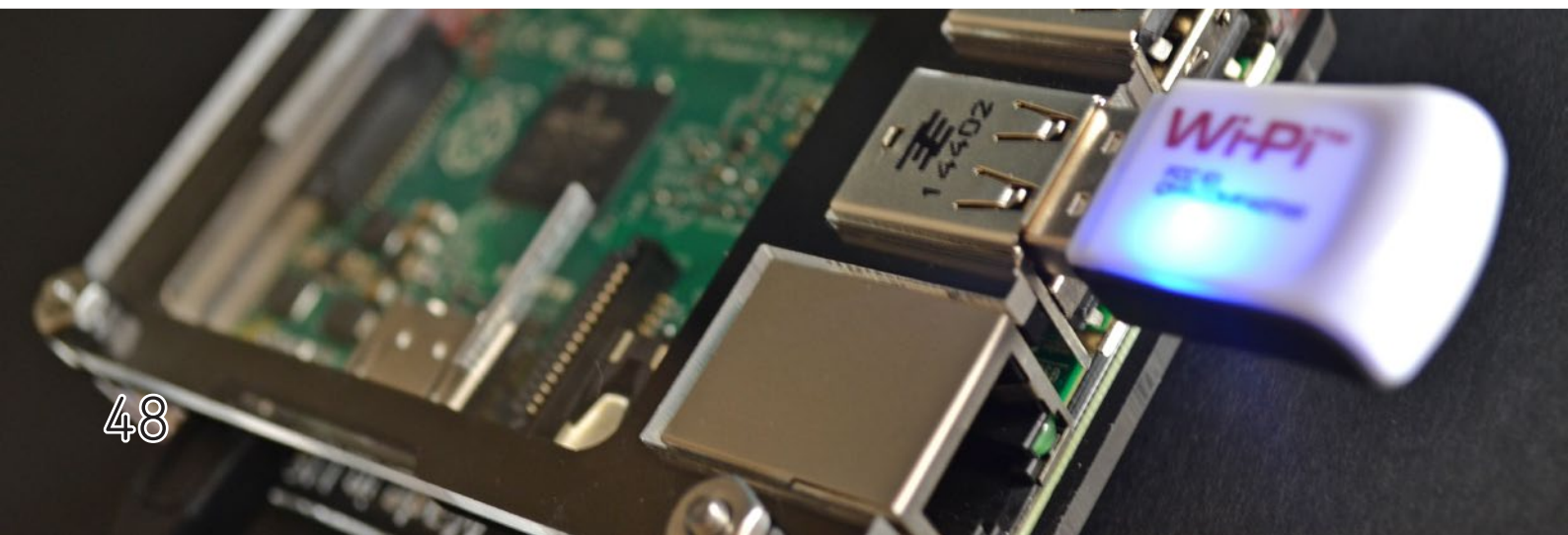
In this guide, we will show you how to set up the Raspberry Pi's Wi-Fi via both the command line for the Raspberry Pi and how to set up the Wi-Fi from the GUI configuration tool that is included within the desktop version of the Raspbian operating system.

One note we should make is that if you are using the Raspberry Pi 3, we highly recommend you get a case for it, as the Wi-Fi module located on the bottom of the board, is susceptible to damage.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Wifi dongle

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case



Setting up Wi-Fi via Command Line

If you don't have the option of setting up the Wi-Fi connection up through the GUI, then this is the next best thing.

It's a little harder but still relatively easy to setup.

1. While you are in the terminal on the Raspberry Pi enter the following command.

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

2. At the bottom of this file add the following lines, replacing the SSID and password with your own

```
network={
ssid="YOURSSIDHERE"
psk="WIFIPASSWORDHERE"
}
```

3. Once you have added those lines, make sure you save them.

In **nano** you achieve this by pressing the following key combination. **Ctrl+X** and then press **Y** to save file.

4. The Raspberry Pi should in most cases notice the changes to the **wpa_supplicant** file and automatically make a connection to the Raspberry Pi.

If it doesn't, you can force it to check by doing the following commands.

4a. This command takes the network interface down, meaning we can restart it to reload the **wpa_supplicant** file.

```
sudo ifconfig wlan0 down
```

4b. This command brings the network interface up, upon it loading up, it will load the **wpa_supplicant** file back in.

```
sudo ifconfig wlan0 up
```

5. You can check if the Raspberry Pi has successfully connected by using the terminal command below.

If the Raspberry Pi has successfully connected to the internet the inet addr field should have an IP address in it.

```
sudo ifconfig wlan0
```

If you're having trouble connecting to your home network be sure to make sure that the Wi-Fi dongle is supported by the Raspberry Pi. Some Wi-Fi dongles require drivers that aren't available on Raspbian or they consume more power than the Raspberry Pi can provide through its USB Ports.

It is also possible you have the SSID wrong, to scan for your home network use the command below and check the ssid field that is provided.

This field should be the same as what you entered in the ssid field.

```
sudo iwlist wlan0 scan
```

Setting a static IP address from the command line

1. Setting up a static IP address for your Wi-Fi connection is a simple task that involves just modifying a single file.

While your WLAN connection should be identified as **wlan0** you can verify this by utilizing the following command on your Raspberry Pi. Any Wi-Fi interfaces will be prefixed with **wl**.

```
ifconfig
```

2. Once you have verified your Wi-Fi connections interface name we can go ahead and modify the dhcpd config file. Begin modifying this file by running the command shown below.

DHCPD is the daemon that provides the Dynamic Host Configuration Protocol service to the network. Basically, it helps things like negotiating the IP address that the device should be using.

```
sudo nano /etc/dhcpcd.conf
```

3. Within this file, you need to add the following lines to the end of the file making changes where applicable. We will describe what each of these lines do below.

```
interface wlan0
static ip_address=192.168.1.115/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

interface wlan0 - This line defines the interface that we are wanting to modify the configuration for. If your wireless connection is not running on wlan0 make sure you change the interface name here.

static ip_address=192.168.1.115/24 - This is the IP address and size (/24) that you want DHPCD to acquire from the network. Make sure this is an unused address otherwise there will be conflict issues.

static routers=192.168.1.1 - This line defines the IP address of your router (or gateway). Make sure this matches the IP address of your router so DHPCD knows where to connect.

static domain_name_servers=192.168.1.1 - This line defines the DNS addresses that the DHCP Daemon will utilize for this interface. Typically this can just be set to the routers IP address.

4. You can now save the file by pressing **CTRL + X** then **Y** and finally pressing **ENTER**.

5. Now to make sure these changes are properly loaded in you should reboot your Raspberry Pi by running the following command in the terminal.

```
sudo reboot
```

6. You Raspberry Pi once rebooted should now stick to using the static IP address that you defined. You can verify your Raspberry Pi's local IP address by running the following command.

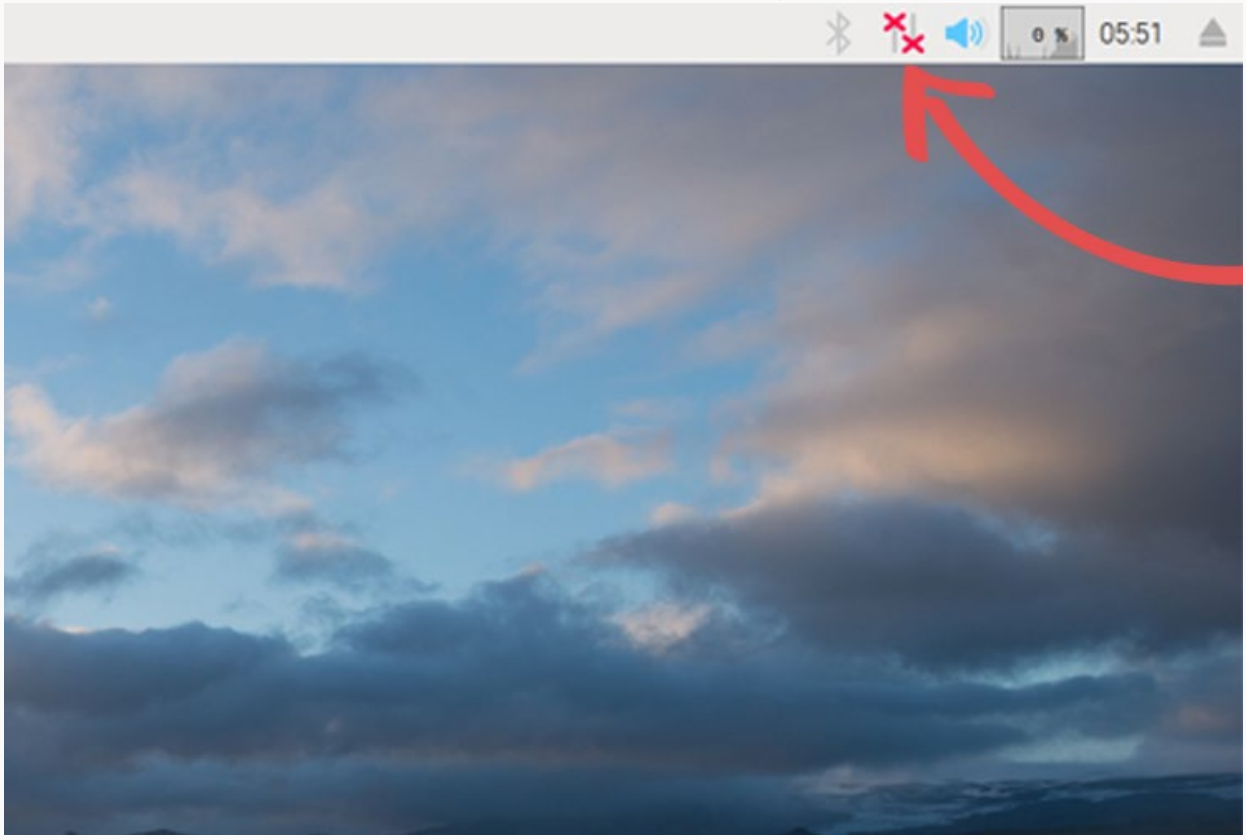
```
hostname -I
```

It is also recommended that you also set a static IP address in your router itself for the device. This will ensure that there is no chance that another device will be handed the local IP address.

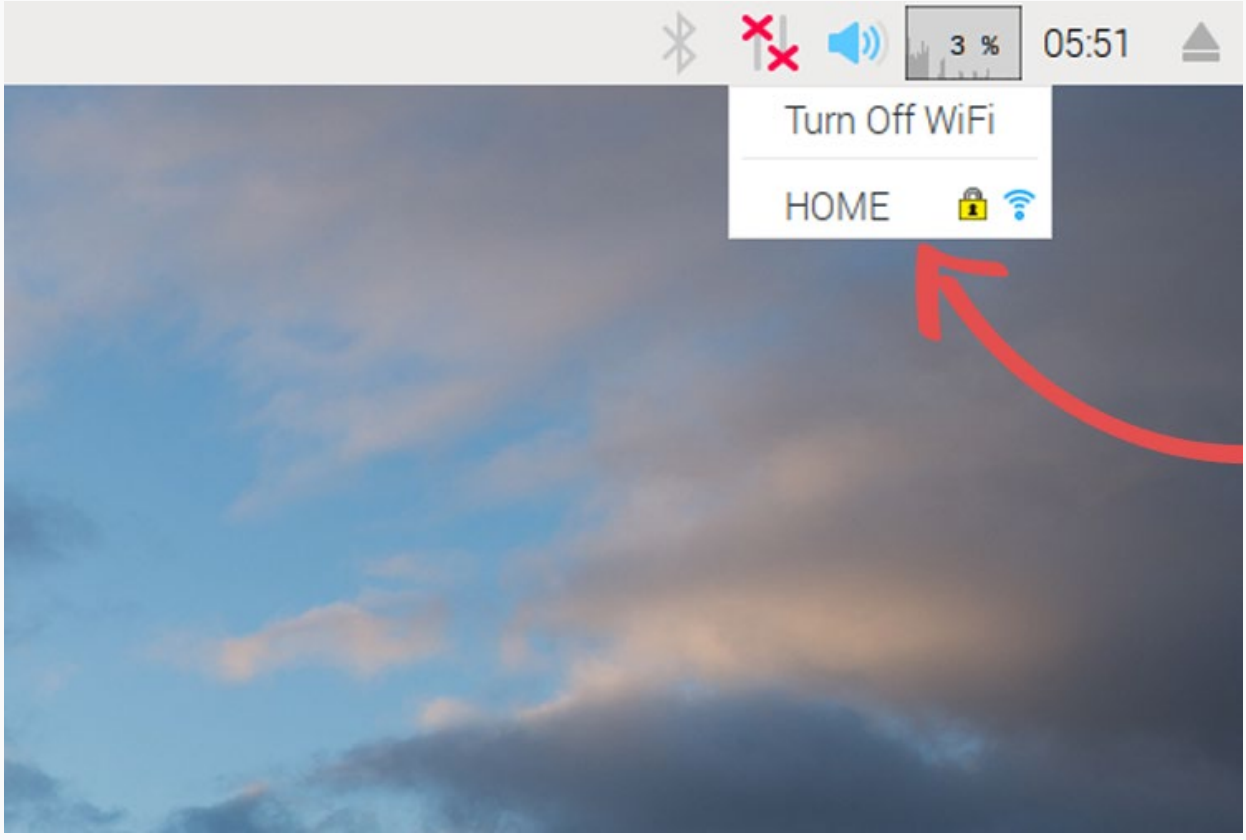
Setting up Wi-Fi via the GUI

1. You will need to have your Raspberry Pi connected to a monitor with a mouse and keyboard.

2. On the Raspbian desktop locate the WiFi icon in the upper right-hand corner of the desktop.

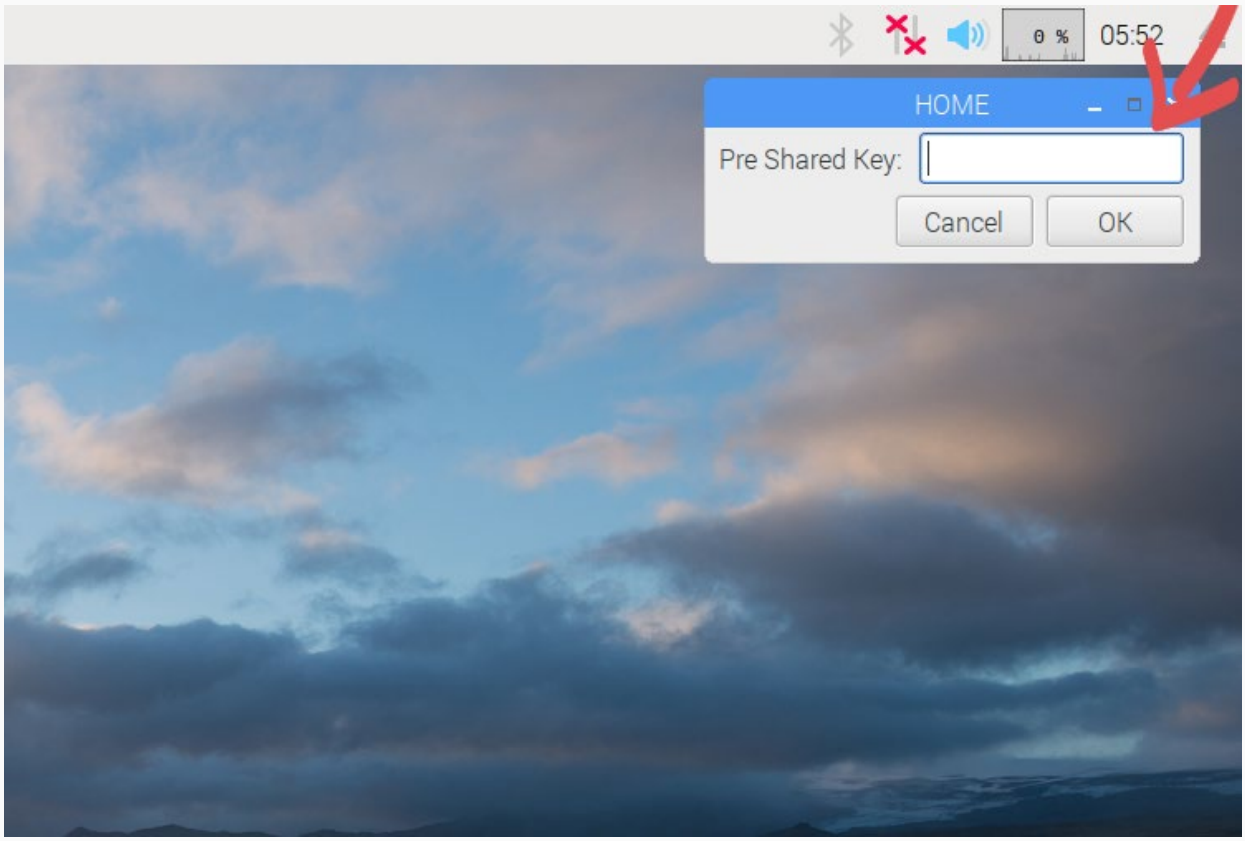


3. It will now scan for WiFi networks and display any that it can find. Click on the network that you wish to connect to.

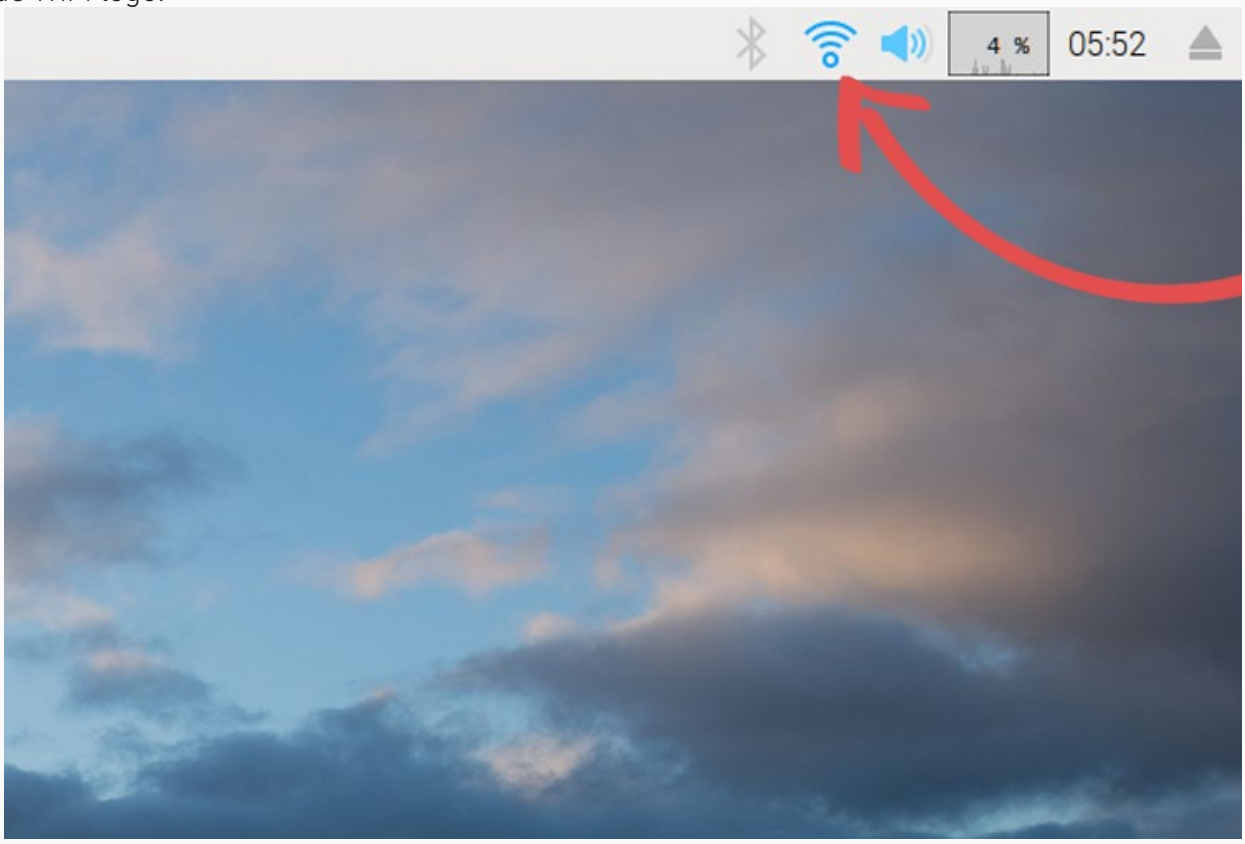


Setting up Wi-Fi via the GUI

4. If your network is password protected, then you will be prompted to enter a password. Enter the relevant password to connect to the WiFi network.



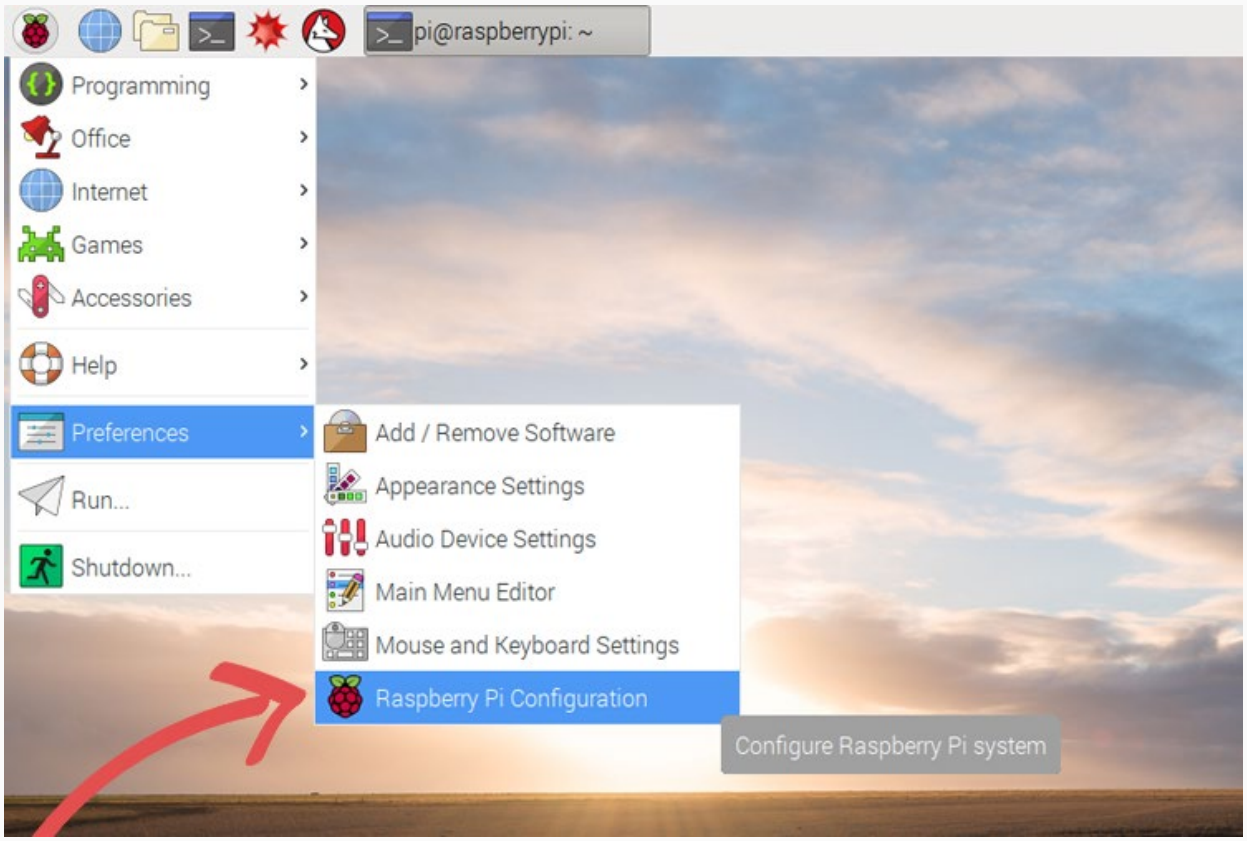
5. The Raspberry Pi should now connect to the WiFi network, and the icon should have now changed to a blue WiFi logo.



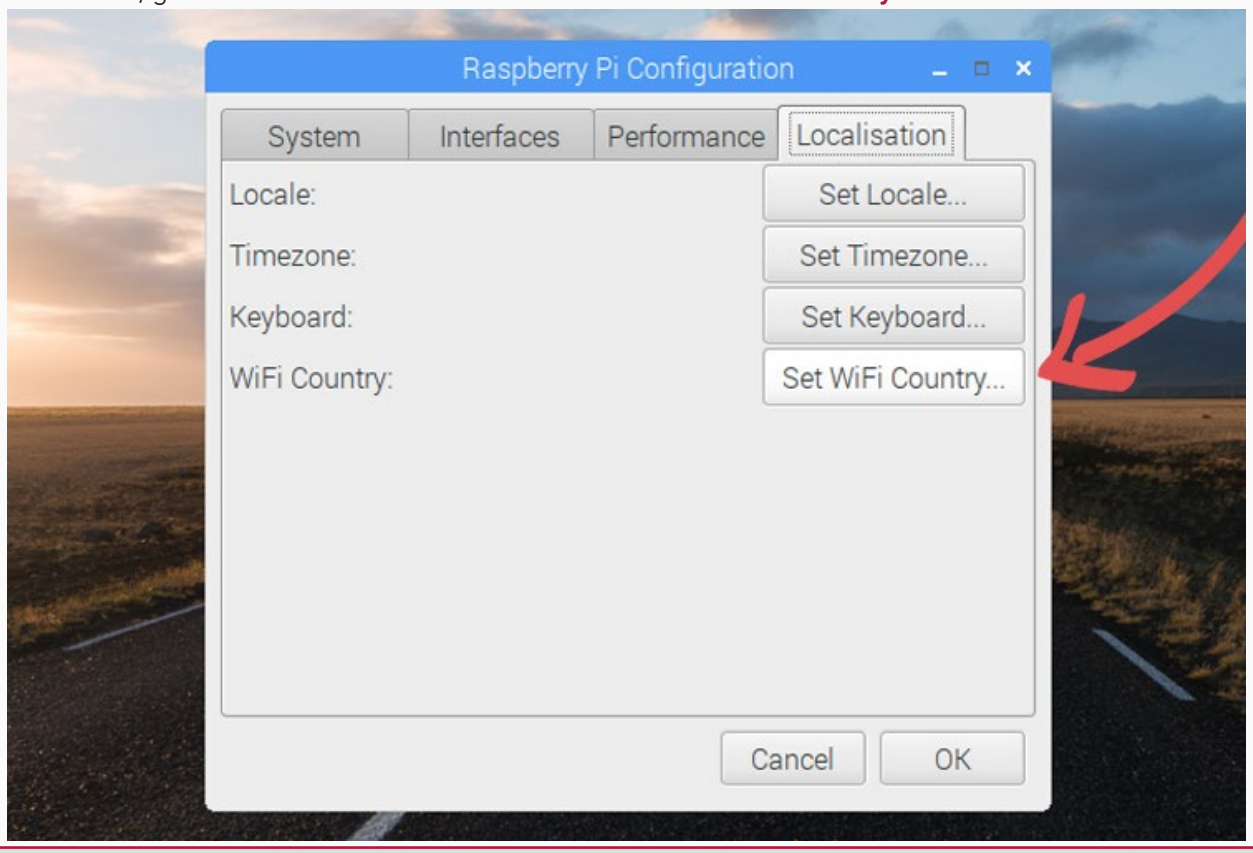
Updating WiFi Country

It is also a good idea to set the country for your WiFi signal. Selecting the country in the GUI is pretty easy to do.

1. Go to the **menu** icon, then **preferences** then select **Raspberry Pi configuration**.

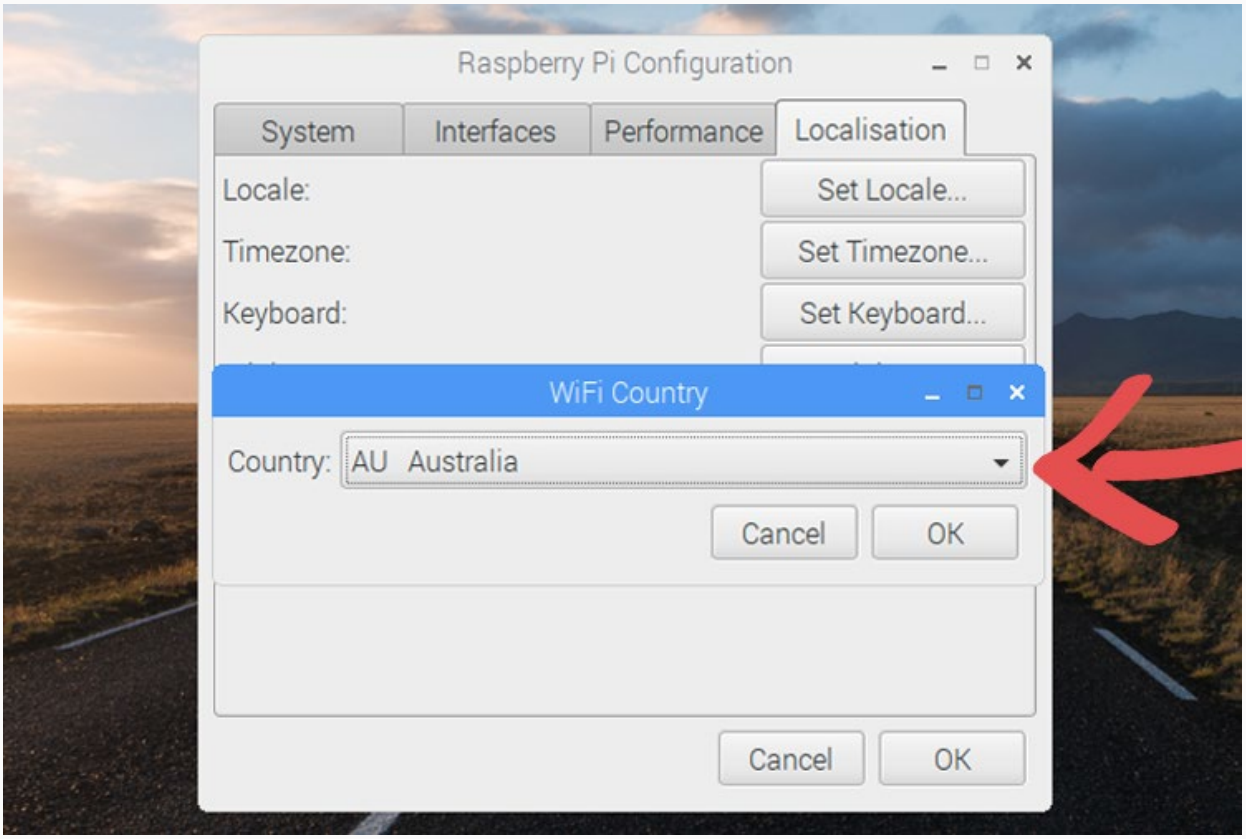


2. In this menu, go to the **Localisation** tab and select "**Set WiFi Country**".



Updating WiFi Country

3. Select your country and hit save.



4. Select your country and hit save.

Predictable Network Names

Predictable network names caused quite a bit of confusion when it was first introduced to Raspbian in the Stretch update. Since then, it's been rolled back and can be activated manually.

You will find that predictable network names are useful when a device has multiple network connections. Instead of naming the interfaces **wlan0** or **eth0**, it instead names them **en** (ethernet) or **wl** (wlan) followed by an **x**. It is lastly followed by a mac address (without any colons). For example, a wlan connection with a predictable network name would be called **wlx6c198f049e5c**

The predictable naming comes in handy if you have three or more network interfaces. For example, on reboot, **wlan0** might become **wlan1** and vice versa. This switch of interface names may cause a lot of issues for people who rely on the interface names in their programs.

Predictable network names provide a naming convention that makes sense and is easy for a developer to predict.

If you wish to turn predictable network names on for use with the Raspberry Pi WiFi or ethernet, then do the following steps.

1. In the terminal, enter the following command.

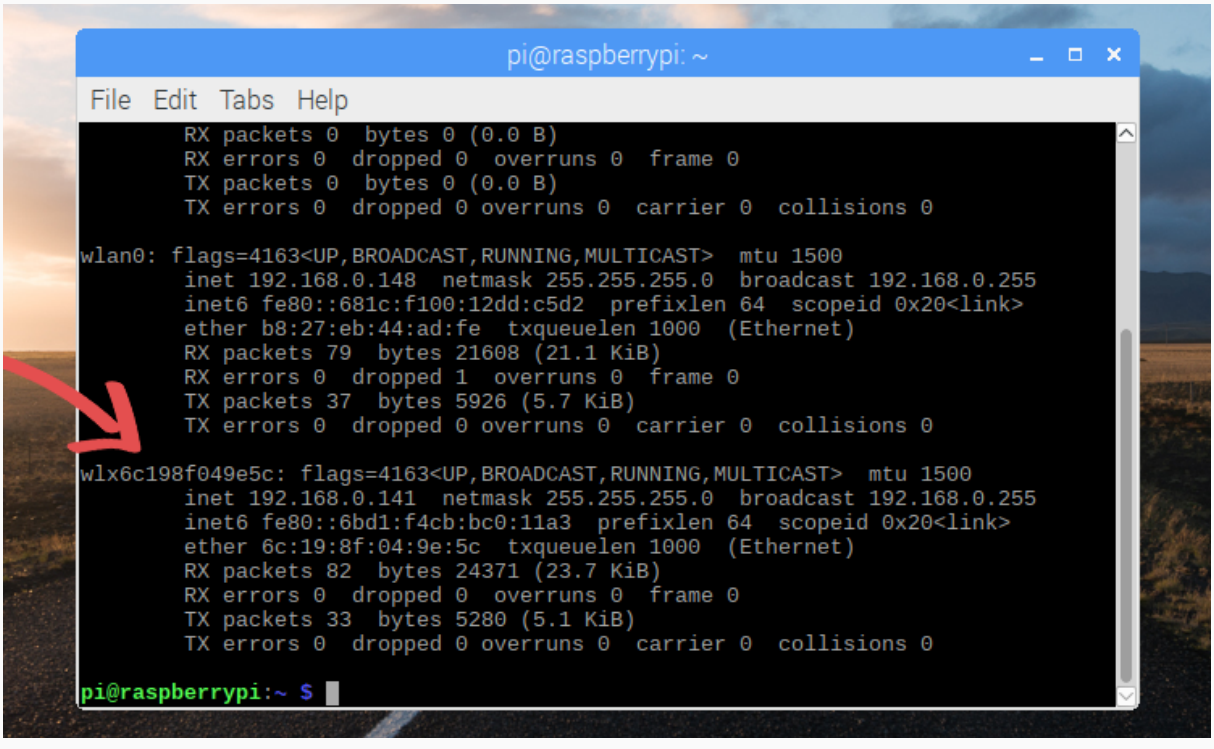
```
sudo raspi-config
```

2. Go to **network options**, then select **network interface** names. In here, select **yes** to turn on predictable network interface names.

3. Now exit raspi-config. You can test to see if predictable network names are working by entering the following command after you have plugged in a new network device.

Devices that are currently connected might still have the old name, and a restart will fix this issue.

```
sudo ifconfig
```



How to utilize SSH

Project Description

Raspberry Pi SSH is a way we're able to communicate to the Raspberry PI over a network so we no longer need to be physically located near the Raspberry Pi.

SSH or secure shell for anyone who is unfamiliar with the term is a common cryptographic protocol for communication over networks.

It allows us to be able to use the command-line without being on the Raspberry Pi, it is an incredibly handy tool, as you won't need an extra keyboard and mouse or even a monitor hooked up to the Raspberry Pi.

SSH is fantastic for a lot of Raspberry Pi projects that don't require you to be at the Raspberry Pi itself, and is a tool that we will rely on heavily for almost all of our tutorials.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

Enabling the Raspberry Pi SSH Server

To setup, the Raspberry Pi SSH server requires us to use the raspi-config tool to enable it. To do this we will need a keyboard, mouse, and a monitor for the Raspberry Pi.

Don't worry though as once we have set up the SSH server you won't have to worry about having access to a keyboard, mouse or monitor.

1. First turn on and log in to the Raspberry Pi. (Default **username** is **pi** and default **password** is **raspberry**)

2. If you are not setup to boot into the terminal, now will be the time to load up the terminal from the GUI. The icon should be located on the desktop under the name **LXTerminal**.

3. If you need the local IP Address now is a good time to get it. To do this simply type the command below:

```
hostname -I
```

4. Now type the following command to get into the Raspberry Pi Configuration Tool:

```
sudo raspi-config
```

5. Now use the **arrow keys** on your keyboard to go to **Interfacing Options** -> **SSH** press **Enter** to go into a menu.

6. You will be greeted with the message "**Would you like the SSH server to be enabled?**", type **Yes** to enable SSH.

Please note that you may have to restart your Raspberry Pi for SSH to properly activate, to do this quit out of the raspi-config menu by pressing **ESC**.

Once you have exited out of the rasp-config menu, you will need to type the following command into the terminal:

```
sudo reboot
```

Once that has completed, SSH should now be successfully set up, and we should be now able to connect to it from other local devices.

Remember to have the local IP address you got from using the **hostname -I** command on hand. As you will need this to connect to your Raspberry Pi.

Over the next two pages, we will go into connecting to the Raspberry Pi by using an SSH client on Windows and, how to connect to it on both Mac OS X and Linux based operating systems.

Connecting to the Raspberry Pi via SSH on Windows

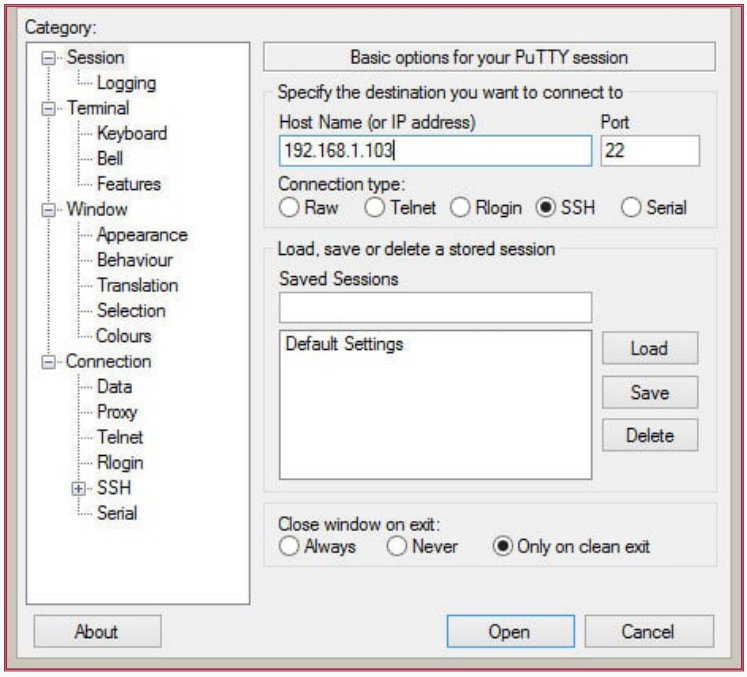
If you're a Windows user, then the process of connecting to the Raspberry Pi over SSH is straightforward.
We will need to get a free application called Putty first as Windows does not come with its own SSH client like Linux and Mac OS X.

1. To use SSH on Windows, we will need an additional tool called Putty. You can download putty from the developer's official site here http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

2. Download **putty.exe** under "**For Windows on Intel X86**"

3. Once **putty.exe** has finished downloading, all we need to do is simply run this file. It will not need any installing as it is a portable application.

4. Upon loading putty you will be greeted with a screen like below, this is where you will enter the IP address and port number for putty to connect to.



5. In this screen, we will need to input the IP of the Raspberry Pi into the Host Name Field

6. Also, make sure that the port number is set to **22**. This port is the default port for SSH connections.

7. Now click on "**Open**" this will begin the SSH connection to the Raspberry Pi

8. You will be prompted with a security alert about the security certificate. This alert is nothing to worry about as in this case you know what you are connecting to. Simply click "**Yes**" to this.

9. Now login with the default username pi and default password raspberry. (Of course, if you have already changed the default password then use the password you changed it to.)

You're now able to connect to the Raspberry Pi and send commands remotely rather than needing to be physically at the Raspberry Pi itself.

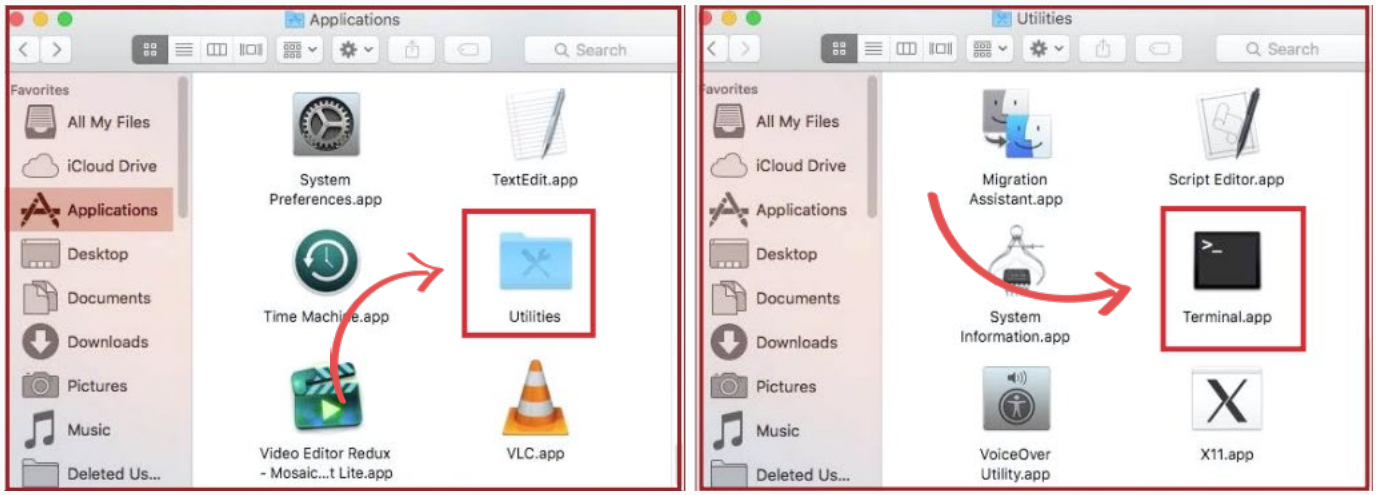
Connecting to the Raspberry Pi via SSH on Mac OS X and Linux

Connecting to the Raspberry Pi over SSH on Mac OS X or Linux is very easy to do. All we need to do is connect using the terminal, and thankfully, unlike Windows, both Mac OS X and Linux include support for SSH
as a base feature of the operating system.

If you're using a Mac or a Linux based computer then we will be able to use the terminal to connect, you will not need any 3rd party tools as SSH supports is included in both operating systems.

If your computer is running the Ubuntu Linux operating system, then you can quickly gain access to the terminal by pressing **Ctrl + Alt + T**.

If you're using a computer running Mac OS X this is found under utilities within the applications folder, the
app you need to look for is shown below:



1. Open the Terminal. Via the **utilities folder** on **Mac OS X** or using **Ctrl+Alt+T** on Ubuntu.

2. Type in the following command to begin the SSH connection with the Raspberry Pi. Replacing **192.168.1.103** with your own Raspberry Pi IP address.

```
ssh pi@192.168.1.103
```

3. You will now receive a security warning about the certificate the Raspberry Pi is using, you don't need to worry about this as you know your connecting to the Raspberry Pi.

Type "**yes**" into the terminal to continue.

4. Now login with the default **username pi** and default **password raspberry**. (Of course, if you have already changed the default password then use the password you changed it to.)

5. You should now be able to control the Raspberry Pi through the terminal.

Connecting to the Raspberry Pi via SSH on Mac OS X and Linux

If you are having trouble trying to get SSH working on your Raspberry Pi or are having trouble connecting to the Raspberry Pi over SSH here are a few tips you can try to get the connection to work.

"I am getting a connection error"

Ensure you have the correct IP address. Having the incorrect IP Address for your Raspberry Pi is the most likely cause for most issues with connecting with SSH to the Raspberry Pi.

There are two ways to check for the IP address of the Raspberry Pi. One way is to check your router and see what IP, it has given to the Raspberry Pi, the second is to use the command below on the Raspberry Pi for it to display its IP Address.

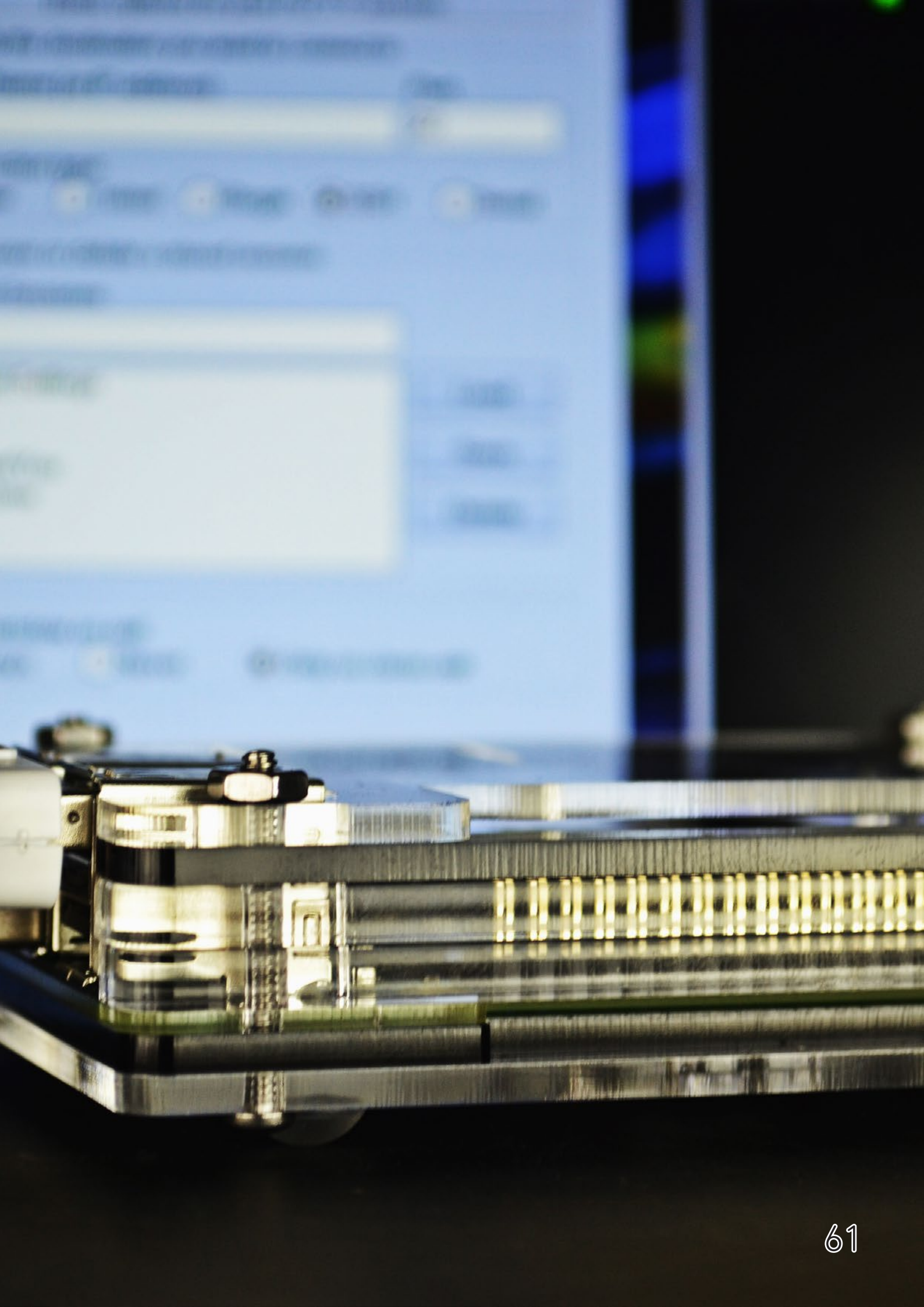
Ensure that you have correctly installed SSH and followed the correct steps for our guide on setting up SSH using the **raspi-config** tool.

Make sure the Pi is connected to the same network as you are on. (Unless of course, you have setup port forwarding and connecting outside the network)

If the Raspberry Pi is on an outside network, you will need to port forward on that network, and have that networks external IP address and no the local IP address for the Raspberry Pi.

"I am getting access denied"

Make sure you are using the correct password for the **pi**, the default password for the account is **raspberry** but remember if you have changed the password away from the default, you need to use that password.



Basics of the GPIO Pins

Project Description

In this Raspberry Pi GPIO tutorial, we are going to look at all the basics of the GPIO pins. GPIO stands for general purpose input and output.

These pins are used for Raspberry Pi to communicate with other circuitry such as extension boards, custom circuits and much more.

You can make the Raspberry Pi do some cool stuff by using these pins as they allow you to get the Raspberry Pi to control an assortment of different electronics, while also allowing different parts to increase the functionality of the Raspberry Pi.

Warning: Experimenting with the GPIO is risky and has a chance of bricking your Raspberry Pi. Knowing the power limitations of the GPIO pins and understanding what you're plugging in is necessary.

Equipment

Required
Raspberry Pi
4 GB (Micro) SD Card (8 GB+ Recommended)
Network connection
1x 100-ohm resistor
1x Red LED

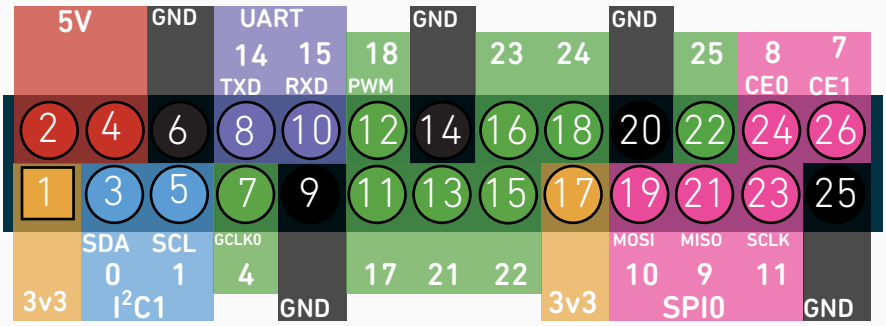
Optional
Raspberry Pi Case (optional)
GPIO Breakout Kit (optional)
Breadboard
Breadboard Wire/Jumper Cables

Introduction to the GPIO Pins

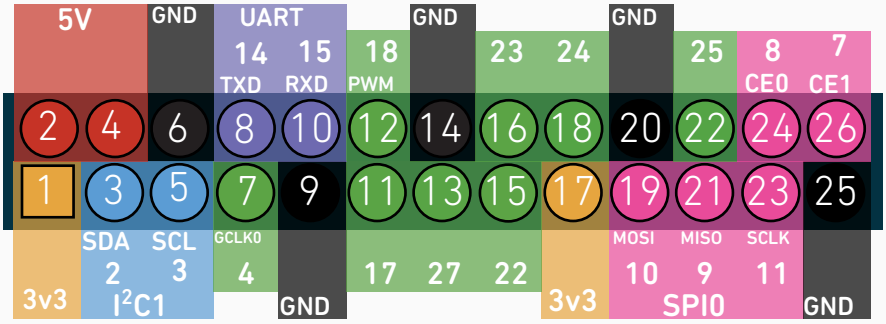
Here we will quickly be running over the GPIO Pins, the ones you should keep an eye out for and what their general layout is for the different versions of the Raspberry Pi's

If you're an owner of a Raspberry Pi B+, 2, zero or 3, then you will have 40 pins in total.
The earlier models of the Raspberry Pi such as the Raspberry Pi B, all have 26 pins.

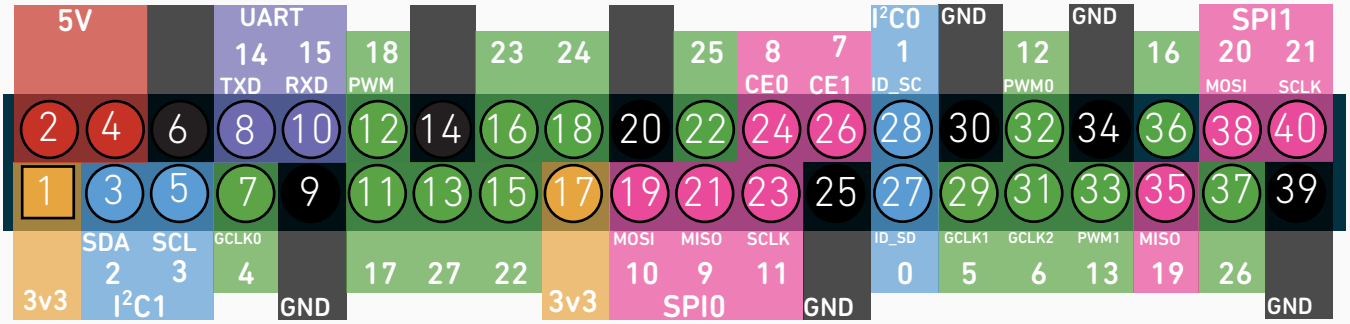
Raspberry Pi (Revision 1)



Raspberry Pi (Revision 2)



Raspberry Pi B+, 2, 3 & Zero



Key

- GPIO Pins
- Inter-Intergrated Circuit (I²C)
- Ground
- 5v Power Pins
- Serial Peripheral Interface (SPI)
- 3v3 Power Pins
- Universal asynchronous receiver/transmitter (UART)

As you can there are more than just your standard GPIO pins. There are some that reference I2C, SPI and UART. We will explain what these different pins are for next.

Explanation of the different GPIO Pins

• **GPIO** is your standard pins that can simply be used to turn devices on and off. For example, a LED. It is also possible to do other more complicated tasks using the GPIO pins.

• **I2C** (Inter-Integrated Circuit) pins allow the Raspberry Pi to connect and talk to hardware modules that support this protocol (I2C Protocol). The I2C protocol will typically take up 2 pins.

• **SPI** (Serial Peripheral Interface Bus) pins can be used for the Raspberry Pi to connect and talk to SPI devices. Pretty much the same as I2C but makes use of a different protocol.

• **UART** (Universal asynchronous receiver/transmitter) is the serial pins used to communicate with other devices.

• **DNC** stands for do not connect, this is self-explanatory.

• The power pins pull power directly from the Raspberry Pi.

• **GND** are the pins you use to ground your devices. It doesn't matter which pin you use as they are all connected to the same line.

Setting up the GPIO Pins

If you are on the latest version Raspbian, then you can start programming and using the GPIO pins without needing to do any extra configuration.

However, it is recommended that you update your Pi to the latest packages anyway. If you haven't done this, then you can do it by running the following commands:

```
sudo apt-get update
sudo apt-get upgrade
```

You can also make sure the Raspbian. Raspberry Pi GPIO interface is installed by just running the following command:

```
sudo apt-get install rpi.gpio
```

Configuring the Raspberry Pi for I2C

Setting up the I2C pins on the Raspberry Pi is super easy and will only take a couple of minutes to do.

1. Firstly, go to the Raspi-Config tool by entering the following command:

```
sudo raspi-config
```

2. In here go to **interfacing options** and then to **I2c**, **enable** I2c by pressing **yes**.

3. The Raspi-config interface should now alert you that it has now been enabled.

4. Now we need to reboot the Raspberry Pi using the following command:

```
sudo reboot
```

5. Now to make sure it has successfully been enabled, we can use the following command:

```
lsmod | grep i2c_
```

This command will return any modules that are running starting with i2c. It should return something like this: **i2c_BCM2708**.

Configuring the Raspberry Pi for 1-Wire

Configuring the Raspberry Pi 1-Wire is much like configuring the Raspberry Pi for I2c and SPI in that it is super easy and will only take a couple of minutes to get up and running.

1. Firstly, go to the Raspi-Config tool by entering the following command:

```
sudo raspi-config
```

2. Once the config tool has loaded go to **interfacing options** and then go to **1-Wire**.
3. Enable the 1-Wire interface by selecting **yes**. It will notify you that it has been enabled.
4. Now we need to reboot the Raspberry Pi using the following command:

```
sudo reboot
```

Configuring the Raspberry Pi for SPI

Configuring the Raspberry Pi for SPI is much like configuring the Raspberry Pi for I2c in that it is super easy and will only take a couple of minutes to get up and running.

1. Firstly, go to the Raspi-Config tool by entering the following command:

```
sudo raspi-config
```

2. With the config tool loaded go to **interfacing options** and then go to **SPI**.
3. Enable SPI by selecting **yes**. It will notify you that is has been enabled
4. Now we need to reboot the Raspberry Pi with the following command:

```
sudo reboot
```

5. To check that SPI has successfully started up and is running use the following command:

```
lsmod | grep spi_
```

This command will return any modules that are running starting with SPI. It should return something like **spi_bcm2835**. Make sure you have restarted your Pi before checking to see if the module has been loaded.

Using a breakout kit

A breakout kit allows you take all the pins via a ribbon cable and connect them to a breadboard or a different device. A breakout kit is a lot easier and safer than working in and around the Raspberry Pi.

There are a few different types of breakout kits you're able to buy for the Raspberry Pi GPIO pins. I prefer the T type as they are easy to read and use. If you're looking for a breakout kit for the Raspberry Pi, Amazon is always a good and easy place to look for one.

When connecting the ribbon cable, you need to make sure it is connected, so it is facing away from the board. You can see an example of a correctly set up ribbon cable with a T type breakout board right below.

Programming with the GPIO Pins

Programming with the GPIO pins on the Raspberry Pi is typically done using the programming language Python. This circuit and code is super easy to get going and won't take you long at all to complete.

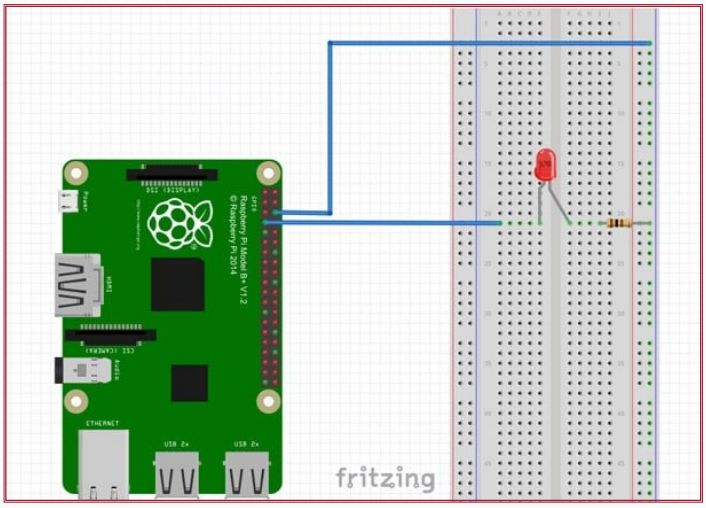
If coding sounds like it may be a little too involved for you and would prefer a simple solution, then something like Raspberry Pi Cayenne might interest you.

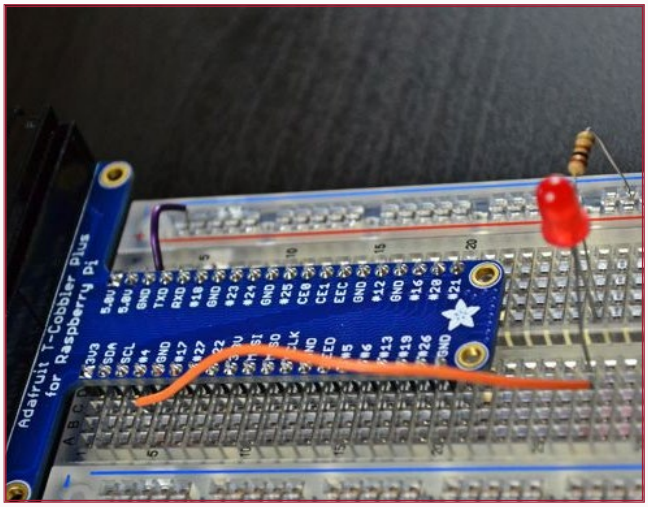
The Cayenne software package allows you to add sensors, control GPIO pins and lots more with a super easy user interface. If you're happy to code and learn lots about Python, then check out my example below.

1. Firstly, let's set up our little circuitI have a helpful, easy diagram to follow that you can find below.

If you have a breakout kit, the circuit will obviously be a bit different since your wires will come from the cobbler. We have included a picture to show how a setup with the breakout kit would look.

Alternatively, just connect the positive end of a LED up to **pin 7** and the negative end to a resistor that connects to **pin 6**.





2. Once you have put together the circuit correctly, make sure you run through it to ensure you have made no mistakes, a small mistake can stop the whole circuit from working correctly.

Now let's create a file so we can type out our python script, we will use this python script to get the Raspberry Pi to interact with our circuit.

We will use the following command to create our python script file.

```
sudo nano led_blink.py
```

Programming with the GPIO Pins

3. Now let's write out a little program. It's important to remember python is **whitespace sensitive**. For example, in the for loop make sure you have **4 spaces for anything within the loop**.

To explain a little further the GPIO.output(7, true) line will need to have **4 spaces before it**.

The code that we will be entering into this file is located on the following page, try to make sure you copy this script exactly.

```
#import the GPIO and time package
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)
# loop through 50 times, on/off for 1 second
for i in range(50):
   GPIO.output(7,True)
   time.sleep(1)
   GPIO.output(7,False)
   time.sleep(1)
GPIO.cleanup()
```

4. Now exit and save by pressing **ctrl+x** and then **y**.

5. Now if you run our script, the circuit should come to life. You can run a python script by doing the following command. It is essential that you run it using sudo due to GPIO's requirements.

```
sudo python led_blink.py
```

What should happen now, is that the LED in the breadboard should turn on for 1 second, then switch back off for 1 second.

If the script is executing correctly, it should do these 50 times, so it should take about 100 seconds for the script to run entirely.

If nothing appears to be happening, make sure that you have connected the circuit correctly. Remember that the **ground connection** should be connected to **pin 6** and the **Red LED** connection to **pin 7**.

Also, make sure that you have the LED connections the right way around, LED's are polarized, meaning power can only travel in one direction. The longer wire out of the LED tends to be the positive connection, the shorter wire the negative. The negative side of the LED also tends to have a flat edge.

If it is still not working, there's a chance the Red LED could be broken, there are two ways you can test this, you can try connecting the RED LED to pin 1 this will cause it to have constant power, so it should turn on immediately. If it doesn't turn on it means you will need to replace the LED as it has died.

The alternative is to try swapping out the LED with another. Swapping out the LED is probably the quickest way to check whether it is the LED that is going wrong rather than having to re-write the LED.

Quick rundown of the blinking LED python script

Here we will quickly be running through the script we used in our tutorial, giving a brief explanation of what each line of code does so you can better understand how you could go about fiddling with the code.

Line 2:

```
import RPi.GPIO as GPIO
```

This line does two straightforward but important things for this script to operate. The first keyword to see here is "**import**." This keyword tells the script it needs to import a certain package, in this case, it's the "**RPI.GPIO**" package. This package handles all interactions between the script and the Raspberry Pi's GPIO pins.

Next keyword to pay attention in this line is "**as**." While not the most important part of the code it does help reduce the amount you must type out. This keyword tells the script to set the package to **GPIO**, so the user no longer has to type out **RPi.GPIO** every time they want to call something in that package.

Line 3:

```
import time
```

This line just imports the **time** package into the script. We use this package to make the script sleep and wait a certain amount of time.

Line 4:

```
GPIO.setmode(GPIO.BOARD)
```

This line tells the **RPi.GPIO** package that we want to set the mode of interaction with the GPIO pins to board mode. This board mode means that the pins will be numbered logically as they sit on the board. It is a more natural way to understand the way of viewing the GPIO pins.

Line 5:

```
GPIO.setup(7, GPIO.OUT)
```

This line is crucial as it setups the actual GPIO pin for input or output. In this case, we want to use **pin 7** as an output, so we use **GPIO.OUT** to tell it to expect output when utilized.

Line 7:

```
for i in range(50):
```

This line starts a loop for our script to run through, it will continue to run until **i** is greater than 50. Every line we want to run within this loop needs to have an **indentation of 4 spaces**.

Line 8:

```
GPIO.output(7,True)
```

This line tells the Raspberry Pi to output to **pin 7** the value of **true** (this is the same as setting it to **1** or **GPIO.HIGH**) this turns the pin on, and it will turn on the Red LED.

Line 9:

```
time.sleep(1)
```

This line tells the script to sleep for 1 second. This function pauses the script from executing any further, which is great if you want to pause before doing certain tasks.

Quick rundown of the blinking LED python script

Line 10:

```
GPIO.output(7,False)
```

This line tells the Raspberry Pi to output to **pin 7** the value of **False** (This is the same as setting the value to **0** or **GPIO.LOW**). This output function turns the pin off and will turn off the Red LED.

Line 12:

```
GPIO.cleanup()
```

This line is very important as it sets all the GPIO pins back to their default values, exiting without calling **GPIO.cleanup()** can cause various issues, such as other scripts failing to be able to use the GPIO Pins.

Taking Screenshots
On your Raspberry Pi

Project Description

In this Raspberry Pi screenshots guide, we take you through the steps on how to take screenshots on the Raspberry Pi.

It's pretty straightforward but keep in mind this process may vary depending on the operating system you're using. This guide assumes you're using the Raspbian operating system.

There are many reasons why you might want to screenshot on the Raspberry Pi. It can make explaining an issue you're having a lot easier. You also might want to do it as part of some Raspberry Pi how-to tutorials.

In this tutorial, we will make use of a software package called Scrot. This package is a screen capture software that makes taking screenshots incredibly easy.

It's a command line tool but you can easily hotkey it to a key on the keyboard or even add it to the menu on Raspbian.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

Installing the Scrot Software

In the latest version of Raspbian, Scrot is already installed, so you're able to skip this installation process. If you're unsure, then you can easily check and install if it isn't already on there.

1. If you are on an older version of Raspbian and unsure if Scrot is installed or not, then you can run the following command to check.

```
scrot
```

2. If Raspbian responds with "**command not found**" then you will need to install Scrot.

To do this, run the following command.

```
sudo apt-get install scrot
```

3. That's all you need to do to get the software installed.

You can confirm that it is working correctly by typing the following command.

```
scrot
```

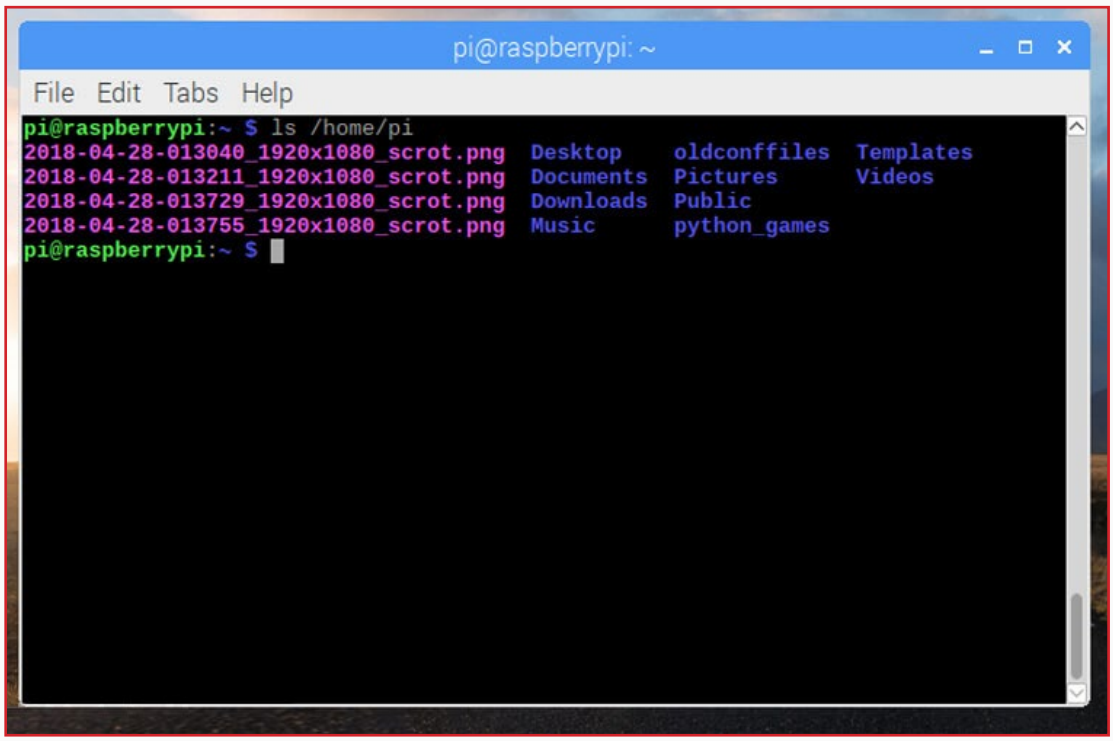
4. If no error occurs, then you can safely assume it's working.

You can also check the default folder, the home user directory to see the screenshots.

The command below will display all the files in the Pi users home directory.

```
ls /home/pi/
```

The images will have the timestamp, resolution and lastly "scrot" at the end.



Next, I will go through the steps of some methods you may find handy for when you go to take screenshots on the Raspberry Pi.

How to take a Screenshot on the Raspberry Pi

Using the Keyboard Hotkey

If Scrot is already installed, then you should find that the **print screen button** on your keyboard will take a screenshot.

You can test the keyboard shortcut by pressing the print screen button and then checking the **/home/pi** folder.

If there are screenshots inside this folder, then the keyboard is working correctly.

Using the Terminal

You can also use the terminal to trigger a screenshot event.

To trigger a screenshot simply enter the following command in and hit enter.

```
scrot
```

Taking a Delayed Screenshot

There are times when you will need to add a delay to the screenshot action.

For example, you can't take screenshots while you have the menu open, so a delay works perfectly if you need to capture this.

In the terminal, enter the following command to take a screenshot after a ten-second delay.

```
scrot -d 10
```

You can change the number to whatever delay you want it to be. Make sure you have **-d** before the number.

Change the Screenshot Save Location

There is always the chance that you would want to set a custom name and location for the screenshot, thanks to scrot this is very simply done.

It can be done by simply adding the location and name after the scrot command.

For example, if I wanted to name a screenshot "**example**" and have it stored in the "**Downloads**" directory, I would enter the following command into the terminal.

```
scrot /home/pi/Downloads/example.png
```

Always make sure you append the name with .png as anything else will confuse operating systems on how to interpret the file.

That's all you need to do for changing the screenshots location via the command line.

How to take a Screenshot on the Raspberry Pi

Mapping the Screenshot Command to the Keyboard

If the screenshot command isn't already a hotkey, then you can add it with a bit of a change to the Pi's configuration file.

To add a hotkey, you will need to add it to the **lxde-pi-rc.xml** file. You can open this file by running the following command.

```
sudo nano ~/.config/openbox/lxde-pi-rc.xml
```

Here is an example of adding the print hotkey. You will need to add this to the keyboard section of the file. **<keyboard></keyboard>**

The following few lines will bind the **scrot command** to the **print screen key** on the keyboard. You can always change the command, so it has a delay or saves to a certain spot.

```
<keybind key="print">

<action name="Execute">

<command>scrot</command>

</action>

</keybind>
```

Once you have made this change, save the file and exit by pressing **CTRL + X**, **Y** then **Enter**.

Now enter the following command to have your changes be loaded in.

```
openbox -restart
```

Your screenshot key should now work whenever you press it.

I hope this Raspberry Pi screenshot guide has shown you enough to be able to set your Pi up for screenshots.

How to Shutdown a Raspberry Pi

Project Description

In this guide, we will take you through all the steps on how to shutdown a Raspberry Pi correctly.

As you may have already noticed, there is no button to switch the Raspberry Pi on or off. Your first instinct is probably to pull the power cord, but this is highly not recommended.

There are many reasons why pulling the power cord out of your Raspberry Pi while the operating system is still running is bad.

Firstly, by pulling the power cord out early, you heighten the risk of your SD card becoming corrupt.

Secondly, anything that is running will not make a graceful exit and save. This forced exit may cause data loss depending on what your Raspberry Pi was doing at the time.

There are more issues that can arise from removing the power cord without first shutting down the operating system, but I am sure you get the point. Incorrectly shutting down the Pi will cause issues.

Luckily shutting down the Raspberry Pi is extremely easy, over the page we will show how simple it is to shutdown the Raspberry Pi and what that command does when you issue it.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

The Shutdown Command

The easiest way to shutdown the Raspberry Pi correctly is to use a very simple command. You can find the command right below.

```
sudo shutdown -h now
```

It will do the following process to ensure the operating system shutdowns gracefully.

1. The system will begin the shutdown process by sending **SIGTERM** to all the running processes, this allows the programs a chance to save and exit gracefully.

2. After an interval, the operating system will send a **SIGKILL** to any remaining processes, this will kill any lingering processes.

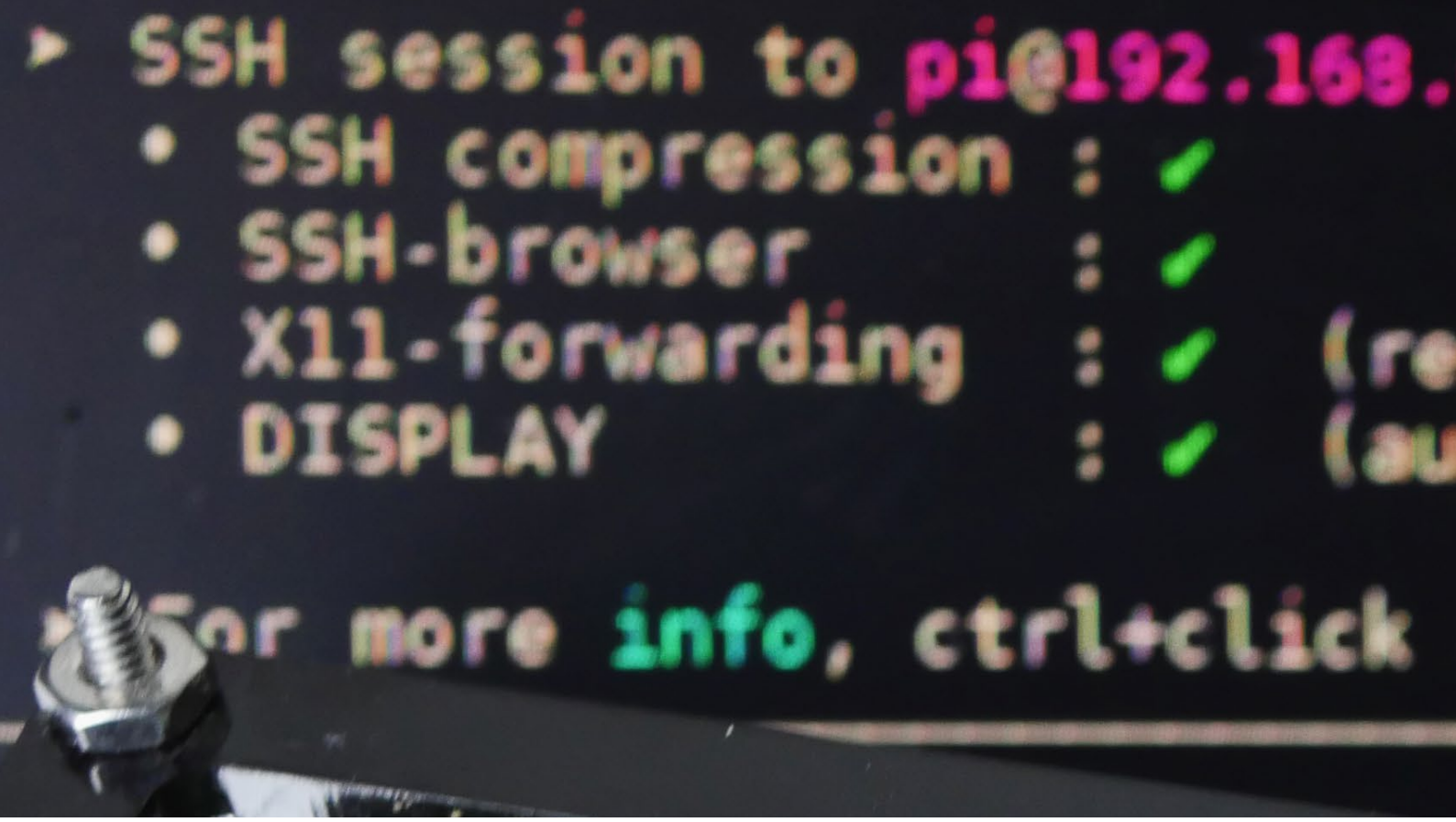
3. Lastly, the system will then move to unmount all the connected file systems.

4. The screen will now show "**System Halted**"

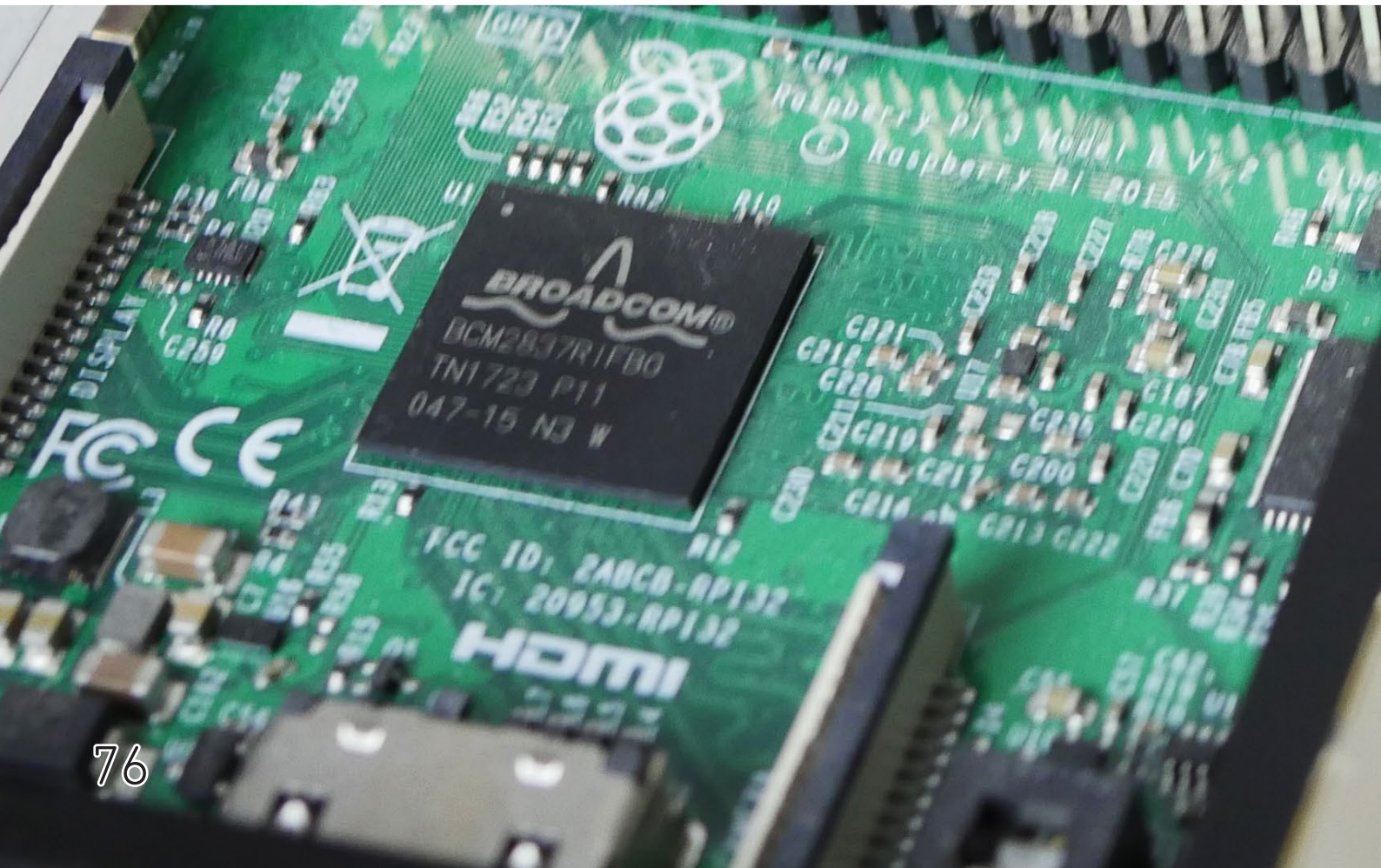
5. Now that the operating system has halted you can now safely remove the power cord with minimal risk to your Raspberry Pi and the operating system.

6. To start the Raspberry Pi simply plug back in its USB power cord.

There are further ways you can improve on this such as building your own power button by making use of the GPIO pins.



Fine tuning your Raspberry Pi




0.143
mote display is forwarded thro
tomatically set on remote serv
on help or visit our website
15:18:51 BST 20
n are free so
cribed in the
to the extent
.103
user has not
user and type

SSH Key Authentication
On Linux Operating Systems

Project Description

In this project, we will be showing you how to setup SSH Key Authentication on your Raspberry Pi. SSH Key authentication is an excellent way of securing your Raspberry Pi as only someone with the private SSH key will be able to authenticate to your system.

This works by generating an SSH Key pair, you will retain the SSH private key, but the public key will go onto the Raspberry Pi's operating system. These SSH keys act as a means of identifying yourself to the SSH server using public-key cryptography and challenge-response authentication.

If you value your security SSH Key's is something you should set up, it offers a few security benefits over password authentication. For starters, it is much harder for an attacker to be able to intercept and is also much more complicated to brute force. A standard SSH Key is usually 2048 characters long, compared to a password that is no longer than 32 characters.

Equipment

Required
Raspberry Pi
4 GB (Micro) SD Card (8 GB+ Recommended)
Network connection

Optional
Raspberry Pi Case (optional)

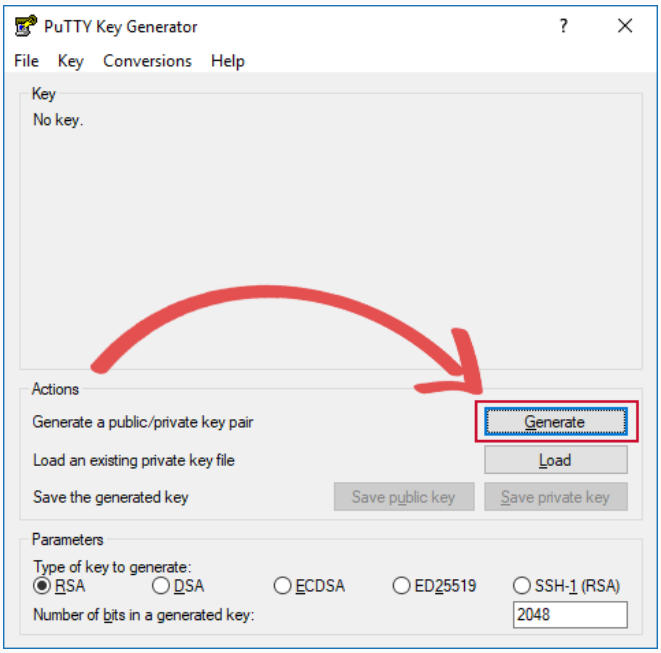
Generating SSH Keys on Windows

To generate SSH keys on a Windows-based operating system, we will have to rely on a piece of software called PuTTY.

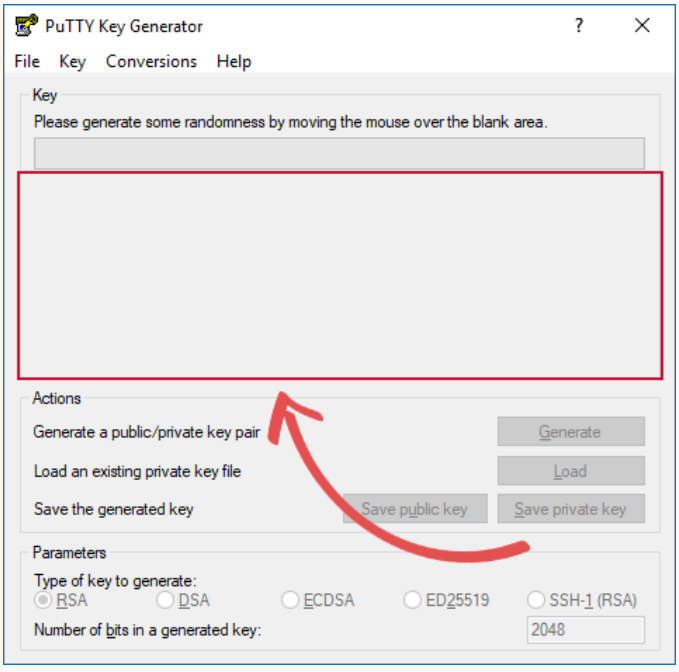
You can download PuTTY from https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html. Make sure you get the full package version as this includes the piece of software that we need to generate the SSH Keys for your Raspberry Pi.

1. Once you have downloaded and installed PuTTY to your computer go ahead and open up the program that was installed alongside it called PuTTYgen.

2. With PuTTYgen opened on your computer, click the "**Generate**" button as we have shown in the image below. Pressing this button will generate the public and private SSH keys that we will use to make our SSH connection to our device, in our case this will be the Raspberry Pi.



3. Once PuTTYgen has begun generating the SSH keys it will ask you to move your mouse in the space as we have indicated in the image below. By doing this, it helps ensure that the SSH key it generates should be genuinely unique and be hard for someone to be able to generate the same key quickly.



Generating SSH Keys on Windows

4. With the SSH keys now generated, there are a few more things that you need to do.

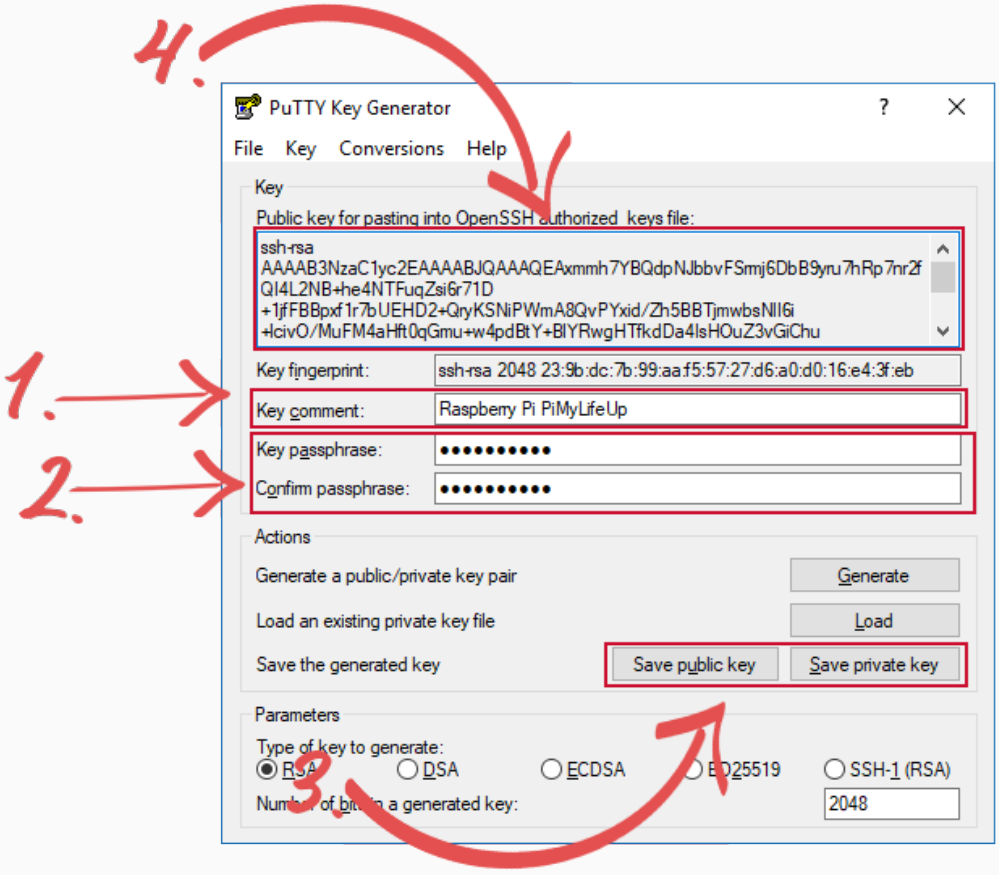
The first (**1.**) is to set a name for this SSH key, make this something memorable, so you know what SSH key is required when connecting.

Second off (**2.**) you should set a passkey, this ensures that even if someone managed to steal your private SSH key, they would still need to enter a password.

This passkey acts as a second line of defense. If you would prefer not to have to enter a password at all you can skip this step and leave the two fields empty. However, we do not recommend this if you value your security.

Thirdly (**3.**) we need to save the public key and the private key to somewhere safe on your computer. Make a note of the location you save both of these files as you will need these to make a connection to your Raspberry Pi. Also, make sure you end the files in **.ppk** so that PuTTY can pick them up.

Finally (**4.**) copy down the public SSH key that is featured in this text box. You can quickly select all the text by clicking on the box and pressing **CTRL + A** then **CTRL + C**. We will need this text shortly to add to our Linux systems SSH Authorized keys file. Without this, the system won't be able to see our private key as a proper authentication method.



5. The next steps of this tutorial will walk you through the process of copying the public SSH key to your linux device. In our case, we will be using a Raspberry Pi that is running the Raspbian operating system.

Goto the section titled "**Copying the Public Keys manually**" to continue on.

Generating SSH Keys on Linux based systems

1. Generating SSH keys on a Linux system is a little easier as the SSH tools to do this are usually included with the main operating system, meaning we do not have to install any additional packages.

To generate SSH Keys open up a terminal session on your Linux device and enter the following command.

```
ssh-keygen
```

2. With the ssh-keygen tool now running you will be first asked to enter a file in which to save the key. For this tutorial, just press Enter to leave this as the default.

Since we are doing this on our Raspberry Pi's Raspbian installation, this default directory was located at **/home/pi/.ssh/id_rsa**.

3. After setting the file in which to save the key we now need to decide whether we want to use a passphrase.

Personally, we recommend that you utilize a passphrase as it ensures that your private key will have a bit of extra security even if someone manages to steal the file as they will need to enter the passphrase to decrypt the private key.

So at this step enter a passphrase (Make sure this is something secure but rememberable), or if you do not like having to enter a password, you can press **Enter** but remember this means anyone who has your private key can access your device without entering any password.

4. Now you have the choice to either copy your SSH public key by utilizing the **ssh-copy-id** tool or manually copy the key itself.

If you want to copy the key over manually, then follow **step 5** and **step 6**, otherwise skip to the next section titled "**Copying the Public Key using SSH Tools**".

5. If you want to copy over your SSH public key manually we will need to get the contents of it now.

To get the contents of the public key, you can utilize the following command on your Linux based device.

```
cat ~/.ssh/id_rsa.pub
```

6. With the contents of the public key now handy we can now proceed to the next step of actually adding the SSH key to the **authorized_keys** file.

Go to the next section titled "**Copying the Public Keys manually**" to learn how to utilize the public keys contents to allow the private key to act as an authorization key.

Copying the Public Key using SSH Tools

1. On your Linux device (In our case one of our Raspberry Pi's), run the following command. Make sure that you replace **IP_ADDRESS** with the IP address of the remote machine that you want to copy the keys to.

Please note that you will be asked to login with both your username and password for that remote machine as the tool needs these to copy over your public key.

```
ssh-copy-id -i ~/.ssh/id_rsa IP_ADDRESS
```

Once done this tool will automatically add your public key to the **authorized_keys** file on the remote machine.

Copying the Public Keys manually

1. Now back on the Raspberry Pi, we need to utilize a few commands to setup our **authorized_keys** file, this is the file that the SSH daemon will check when a private key is used for authentication.

To begin let's create the folder that our **authorized_keys** file will be sitting in. To do this, we will be using the **install** command with a few parameters to set the correct permissions. Run the following command on your Raspberry Pi.

```
install -d -m 700 ~/.ssh
```

2. With the folder, newly created let's go ahead and put our public key in the **authorized_keys** file. To do this run the following command to begin editing/creating it.

```
nano ~/.ssh/authorized_keys
```

3. In this file copy and paste the contents of the public SSH key that you generated either on your Windows device or your Linux device. SSH will authenticate any private keys against the public key present in this to see if it is a legitimate connection to authorize.

4. Once you have your public SSH key entered into the **authorized_keys** file, you can save and quit out of the file by pressing **Ctrl + X** then **Y** and finally **Enter**.

5. With the file now saved we need to make sure it has the correct permissions. To do this, we need to run the following commands. These commands will assign the correct permissions to the file so that it can be read by SSH when you try to log in.

If you are not using the default "**pi**" user on Raspbian make sure you replace the text "**pi**" in the following command with the name of the user you want to use this for authentication.

```
sudo chmod 644 ~/.ssh/authorized_keys
sudo chown pi:pi ~/.ssh/authorized_keys
```

6. With the SSH private key now saved and the permissions correctly set we can now proceed to log in. We will do this before we disable password authentication, so we do not lock ourselves out of our Raspberry Pi.

For our tutorial, we will be showing you how to connect to your Raspberry Pi using your private key and PuTTY.

Connecting using your private key on Linux

1. Utilizing the private key you generated is dead easy on the Linux device that you generated the key in the first place.

The SSH tool by default on most Linux based systems is designed to automatically make use of the private key when attempting to make a connection.

As long as you are using the machine, you used to generate the private key you can use the SSH command as shown below.

The system will automatically try to use the private key that we created earlier to make the connection.

Thanks to copying the public key into the **authorized_keys** file of the remote host, it will be able to recognize our incoming private key and be able to accept our connection.

```
ssh IP_ADDRESS
```

2. If you set a passphrase, you will now be asked to enter that before you can continue, this is required to unlock your private key.

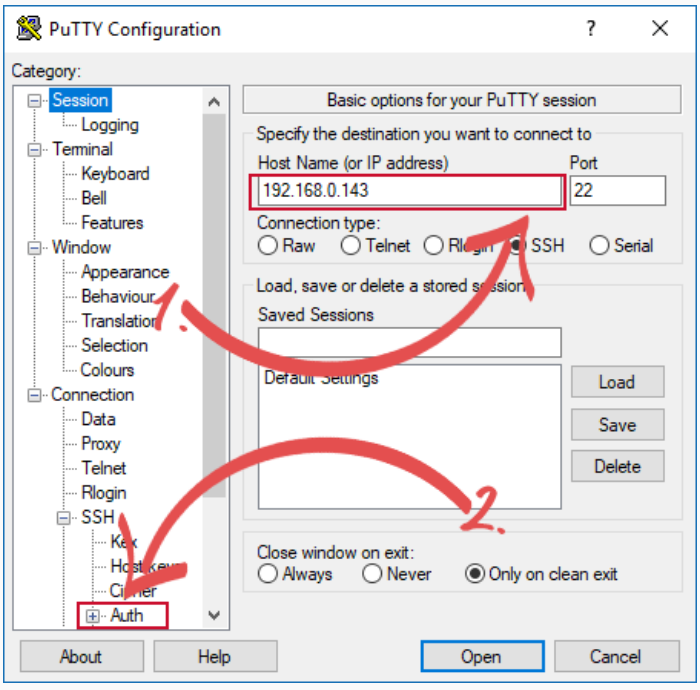
After entering your passphrase, you should now be logged into the remote machine.

If you don't like having to enter your passphrase every time, don't worry we will go into how to cache this later on in the tutorial.

Connecting to your Raspberry Pi using a Private Key with PuTTY

1. In this section of the tutorial, we will be showing you how to use your private key with PuTTY to connect to your Raspberry Pi. Connecting using a private key is a relatively simple process.

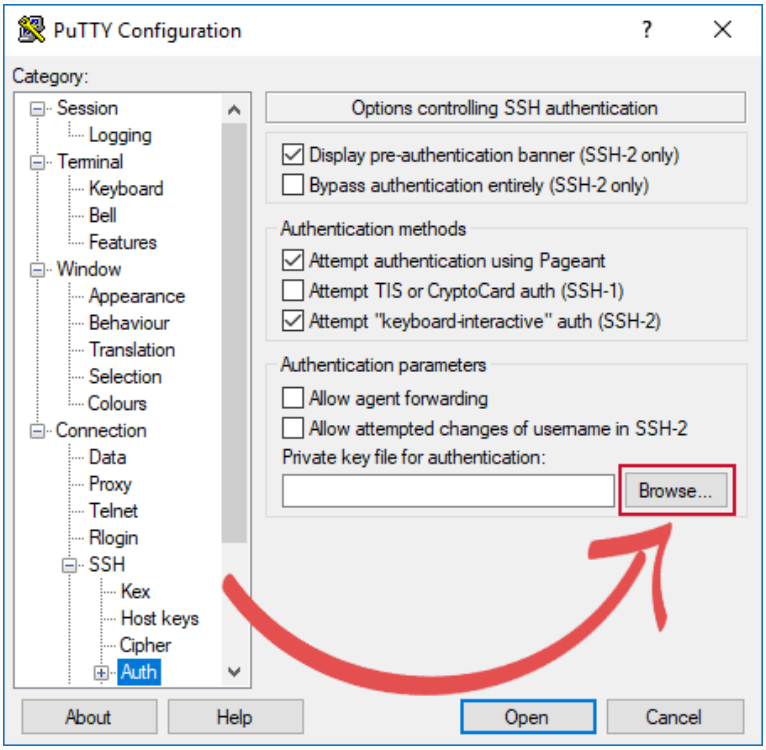
Start off by opening up PuTTY on your computer and entering your Raspberry Pi's IP address (**1.**) then click on "**Auth**" under the "**SSH**" section (**2.**)



Connecting to your Raspberry Pi using a Private Key with PuTTY

2. Next, you need to press the "**Browse**" button. This button will allow you to find and select the private key that we saved earlier on in the tutorial. Selecting this file will allow PuTTY to try and use it for authentication.

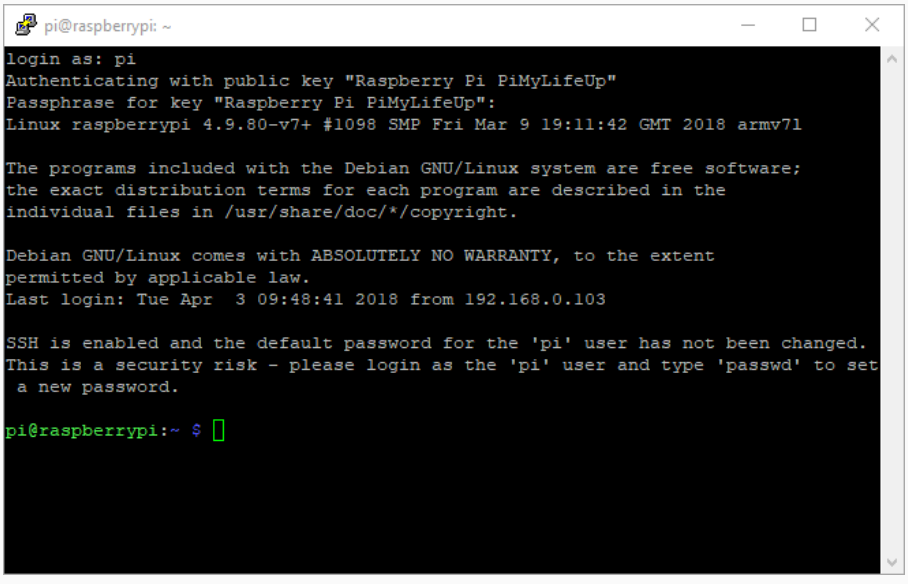
After you have selected the private key from the browser, you should now press the "**Open**" button to start the connection.



3. Upon connecting you will be first asked to enter a username, make sure this is the username that belongs to the private key that you are using otherwise authentication will fail.

After entering the correct username, you will be now asked to enter a passphrase for your private key if you set one earlier. If you get the passphrase wrong

Upon entering a correct passphrase you will be logged into the SSH session, and you can now proceed onto disabling password authentication completely.



Removing Password Authentication

1. To disable password authentication, we need to modify the **sshd_config** file. Within this file, we can change the behavior of the SSH daemon.

To modify this file run the following command on your Raspberry Pi.

```
sudo nano /etc/ssh/sshd_config
```

2. Within this file, we need to find the following line and change "**yes**" to "**no**".

This simple change will completely disable the ability to login to your Raspberry Pi with just a password.

From now on you will require the private key to gain access to the system through SSH.

If you are having trouble finding the line you can use **CTRL + W** to find it quickly.

Find

```
#PasswordAuthentication yes
```

Replace With

```
PasswordAuthentication no
```

3. Now you can save and quit out of the file by pressing **CTRL + X** then **Y** and finally **ENTER**.

4. With the changes now made to the **sshd_config** file, we should now restart our Raspberry Pi to ensure the changes are loaded in.

Remember to make sure that your private key, is in fact, allowing you to connect to your Raspberry Pi as passwords won't work after restarting.

Once you are happy with everything, use the following command on your Raspberry Pi to restart it.

```
sudo reboot
```

5. Now if everything is working as it should you should, only be able to perform an SSH connection if you have a valid private key.

Without the private key, the connection will be refused by the SSH agent. As you can no longer use your password, keeping your private key safe is a very crucial task as it is now your only way of remotely accessing your device.

If for some reason you manage to lose your private key or forget the passphrase for your private key there is still one way of gaining back access to your device.

To fix any issues that may arise with your SSH connection you can still physically connect a keyboard and mouse to your device to regain control.

To restore password access over the SSH connection, you should try reverting the change we made to the **PasswordAuthentication** setting and then refollow the tutorial to set back up the SSH keys.

Caching SSH passphrase for the current terminal session

1. If you are using the SSH bash tool, you can cache the passphrase for your private key while the current session is still going.

To do this, we must first start up another session of the ssh-agent. We can do this by running the following command within the terminal session.

```
eval $(ssh-agent -s)
```

Upon entering this command, you will be shown a process id for the ssh-agent that we just loaded. You can use this process id a later on to kill the agent and remove the passphrase caching.

The process id should appear something like, "Agent pid **26484**", you need to make a note of the number.

2. Now that we have started up our additional session of the ssh-agent let's go ahead and add our private key to it.

We can do this by just typing in the following command, be prepared to enter your private key's passphrase.

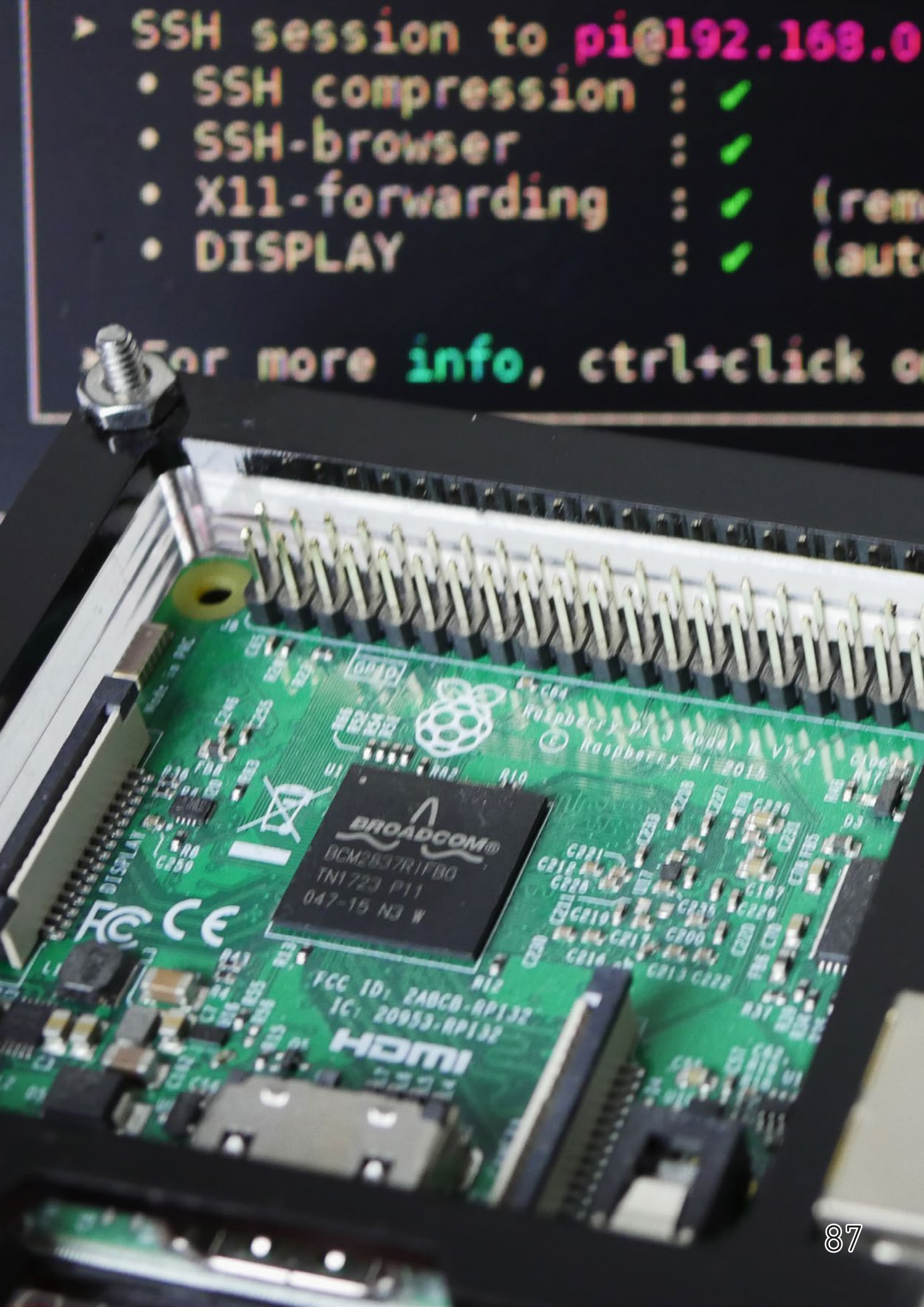
```
ssh-add
```

3. With your SSH key now added to the agent, you should be able to login to any remote machine that has your key authorized without needing to enter your passphrase.

4. To remove your private key from the SSH-agent cache, you will need to kill the ssh-agent we started earlier.

The easiest way to do this is to make use of the process id that we grabbed earlier. Just insert that process id after the command 'kill' to kill the process.

```
kill PROCESSID
```


SSH session to pi@192.168.0
SSH compression : ✔
SSH-browser : ✔
X11-forwarding : ✔
DISPLAY : ✔
For more info, ctrl+click
BROADCOM
BCM2837RIFB0
TN1723 P11
047-15 N3 W
FCC ID: 2ABCB-RPI32
IC: 20953-RPI32
HDMI

Setting up VNC
For Remote Desktop Access

Project Description

In this tutorial, we will go through the all the steps to setting up a Raspberry Pi VNC server.

Once we have done this, you will be able to remote desktop to your Raspberry Pi, so you're able to control as if you were sitting there with it in front of you.

The process of setting up a VNC server on the Raspberry Pi is easy. Utilizing VNC is incredibly handy if you don't have a monitor or are too far away to access the Raspberry Pi physically.

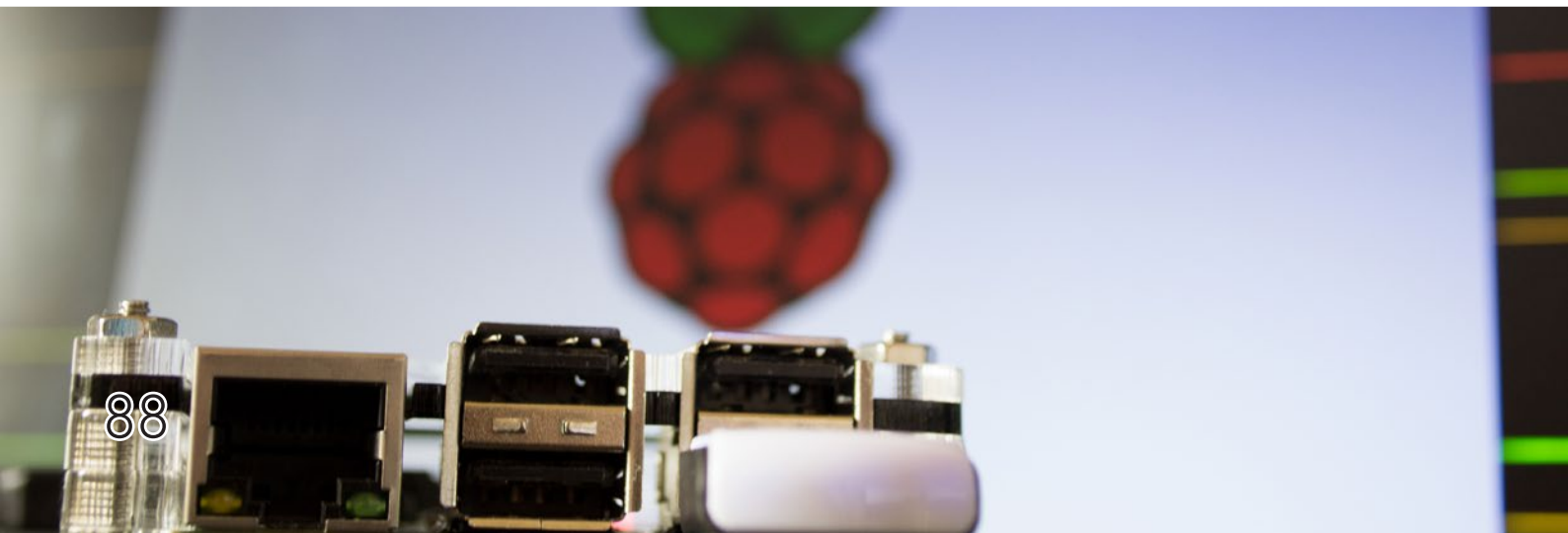
Remote desktop is a simpler alternative to the command line available over SSH. It is a must-have if you prefer being able to use the actual desktop GUI and are not comfortable using the command line.

While we tend to control the Raspberry Pi using the command line, it all just comes down to preference and how comfortable you are with either the GUI or the command line.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case



Connecting to the Raspberry Pi via SSH on Mac OS X and Linux

As in all my tutorials, you will need to have Raspbian pre-installed on your Raspberry Pi.

Now installing the VNC server software package onto your Raspberry Pi is incredibly easy and only requires entering a couple of commands into the Terminal. You can open the terminal from the desktop if you are running Raspbian in desktop mode.

1. Before we get started we need to make sure your Raspberry Pi is looking at the latest repositories and is up to date by running the following commands:

```
sudo apt-get update
sudo apt-get upgrade
```

2. Once those two commands have finished running, enter the following to install the **tightvncserver** VNC server. This server we are installing will allow the VNC Client to interact with the Raspberry Pi

```
sudo apt-get install tightvncserver
```

3. Now all you need to do is setup and run the VNC server, this is easy to do, in fact, you can start the setup process with a single command. Just run the following command:

```
svncserver :1
```

4. You will now need to set up a password for when you connect to the Raspberry Pi via the VNC client. It is important to remember that the password will be **truncated to 8 characters** so try and choose a password that works best for you that is no longer than **eight characters long**.

5. The setup will next ask you to verify the password, just enter the password you entered previously to proceed.

6. Next, it will ask if you want a separate view only password, just enter "**n**" into the terminal.

7. Now that is all done the VNC server will soon be up and running. The following command will need to be run every time the Raspberry Pi reboots, or for some reason, the process is stopped.

```
vncserver :1
```

8. If for any reason, you want to stop the VNC server from running, then the below command will kill the process.

```
vncserver -kill :1
```

The Raspberry Pi VNC server should now be up and running correctly. Our next few steps will show you how to have the VNC Server start automatically on boot.

Having the VNC server start automatically (Optional)

Having to run the command above every time you restart your Raspberry Pi is not the ideal solution. You might want to instead, have the VNC Server start automatically whenever your Raspberry Pi is booted up.

The following steps are straightforward and will take you through the steps to having the server start-up on port one at boot up.

1. First open the rc.local file by entering the following command. This file controls what is started at bootup.

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

2. Now enter the following before the exit 0 line in the rc.local file.

```
su - pi -c '/usr/bin/tightvncserver :1'
```

3. Save and exit by pressing **Ctrl + X** then **y**.

4. Now we need to restart the Raspberry Pi by running the following command. Using this command will test to make sure that the VNC server is booting up as intended.

```
sudo reboot
```

5. You can confirm that it is running by running the following command.

```
vncserver:1
```

6. It should say "**A VNC server is already running at :1**" if its booting up as intended

7. If for any reason, you want to stop the VNC server from running, then the below command will kill the process.

```
vncserver -kill :1
```

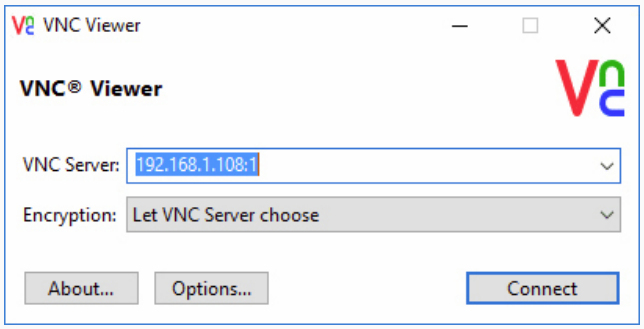
Setting up the VNC client on your PC

Installing and setting up a VNC client on your computer is incredibly easy. There are quite a few different VNC software packages out there, but in this tutorial, we are going to be using realvnc. It's free and available on multiple operating systems.

1. To begin head over to **https://www.realvnc.com/download/viewer/** and download the VNC viewer client relevant to your operating system.

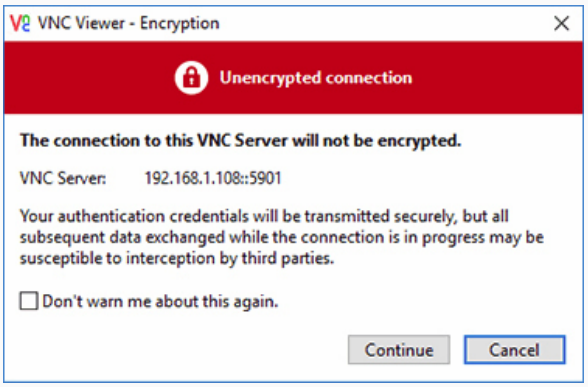
2. Once downloaded, simply open the program.

3. Enter your IP address with the port 1 in the field that says VNC server. For example, mine is, **192.168.1.108:1**



4. Now simply press the "Connect" button.

5. It should then give you a warning message, just press the "**Continue**" button.



6. Then it will ask you for the password you set for it earlier. Just enter the password that we set earlier.

7. The VNC window should load, and you will now have access to your Raspberry Pi remotely.

Troubleshooting the Raspberry Pi VNC Server

I can't connect! – Be sure to double check the IP you entered and make sure it is the same as the Pis. Also, be sure to make sure that the VNC server software is running.

Most routers also dynamically give out local IP's so it is possible that the local IP will change over time. To fix this setup a fixed IP or check the IP whenever you go to connect to the device.

I want to access it outside of my local network! – Opening up access outside of your local network will open security issues, but you are able to do this by setting up port forwarding. Remember the VNC server is running on port 1.

If you are unsure on how to port forward on your router, we recommend checking the following website: **https://portforward.com/**, it is probably the more reliable website for information on port forwarding on a huge variety of different routers.

Please note that port forwarding can expose your network to security risks.

I want a different/larger sized screen on my VNC client – When you go to start the VNC server you can add an extra command called geometry and then the screen size width and high you want (This is in pixels). The command is:

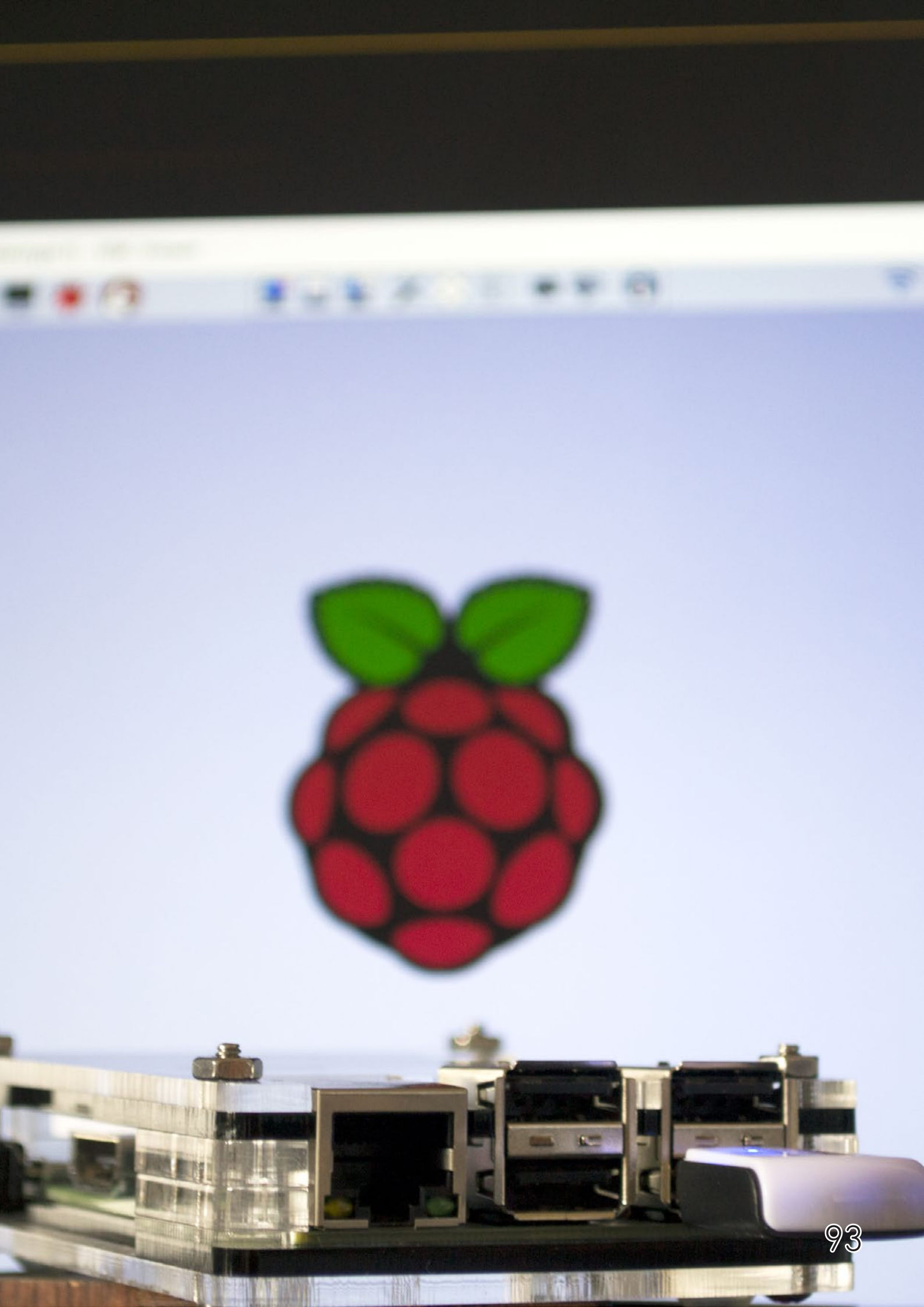
```
vncserver :1 -geometry 1440x900
```

Remember that if you have setup vncserver to autoboot, you will have to edit the **rc.local** file again with the updated command. Edit the file by using the following command.

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

Now enter the following before the exit 0 line in the rc.local file. Replace your pre-existing vncserver line with the following line.

```
su - pi -c '/usr/bin/tightvncserver :1 -geometry 1440x900'
```



Setting up VNC
For Screen Sharing

Project Description

In this tutorial, we are going to show you how to setup VNC so that you can share screens. Sharing screens is incredibly helpful if you require viewing what is being displayed on the main screen.

Usually, VNC will create a new session on the Raspberry Pi meaning you won't be able to see what is happening on the screen from the current user or any other users that are connected to the Raspberry Pi.

To be able to share the screen this we utilize a VNC package called Vino. Vino uses the same VNC protocol as others, but instead of creating a new session it just repeats what is currently being displayed in the main session.

Please note to use Vino you have to be running a version of the Raspbian operating system that has a GUI. You can not be running a lite version or headless version of Raspbian. If you want just to share the terminal, then look onto our Terminal Sharing tutorial that teaches you how to utilize tmate.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

Installing Vino for VNC Screen Sharing

1. One of the very first things we must do before we go ahead and get screen sharing up and running on our Raspberry Pi is to make sure it is running the latest available software.

Do this by running the following command within the terminal on the Raspberry Pi.

```
sudo apt-get update
sudo apt-get upgrade
```

2. With our Raspberry Pi now entirely up to date, we can now move to install Vino; this is the VNC server we will utilize to provide us screen sharing on the Raspberry Pi.

We have to install Vino since the default one present on the Raspberry Pi does not accept this behavior and creates each connection is a new session.

Type the following command onto the terminal on the Raspberry Pi to install it.

```
sudo apt-get install vino
```

3. Using the gsettings command, we can control various settings for Vino; this allows us to switch on and off things such as password authentication and encryption.

You may need to run some of these to get Vino to play nicely with your

Enabling Login via password

To enable logging in via password, utilize the following two commands on your Raspberry Pi's terminal.

Make sure that you replace "**YOUR_PASS_HERE**" with the password you would like to use to log in over VNC.

```
gsettings set org.gnome.Vino authentication-methods "['vnc']"
gsettings set org.gnome.Vino vnc-password "$(echo -n "YOUR_PASS_HERE" | base64)"
```

Disabling Login via password

If for some reason you don't want any authentication requirements when connecting to your Raspberry Pi over VNC you can enter the following command.

This command will disable the password requirement.

```
gsettings set org.gnome.Vino authentication-methods "['none']"
```

Disabling Encryption

Some VNC Viewers such as TightVNC do not support the encryption method that Vino utilizes. To avoid issues with this, we can disable the encryption by running the following command on the Raspberry Pi.

We recommend you first try out your VNC Viewer with this option enabled before trying it without as encryption does have its obvious security benefits.

```
gsettings set org.gnome.Vino require-encryption false
```

Installing Vino for VNC Screen Sharing

Disabling User Confirmation

Like encryption, sometimes the user confirmation can cause issues with VNC Viewers. If you have problems with connecting to your Raspberry Pi via VNC, then try disabling this by running the following command.

```
gsettings set org.gnome.Vino prompt-enabled false
```

4. Now you can utilize the command below to grab your Raspberry Pi's local IP address. Remember when typing in this command to capitalize the 'I'

```
hostname -I
```

5. Now we can proceed to start the Vino VNC server. We can do this by running the following command on the Raspberry Pi.

```
/usr/lib/vino/vino-server
```

6. With the Vino server up and running now is the time to test it out on your device, our next section will go over utilizing a VNC Client on Windows.

If you already have a VNC Client and can test the connection you can skip past the next part. If you have issues, try disabling encryption or user confirmation.

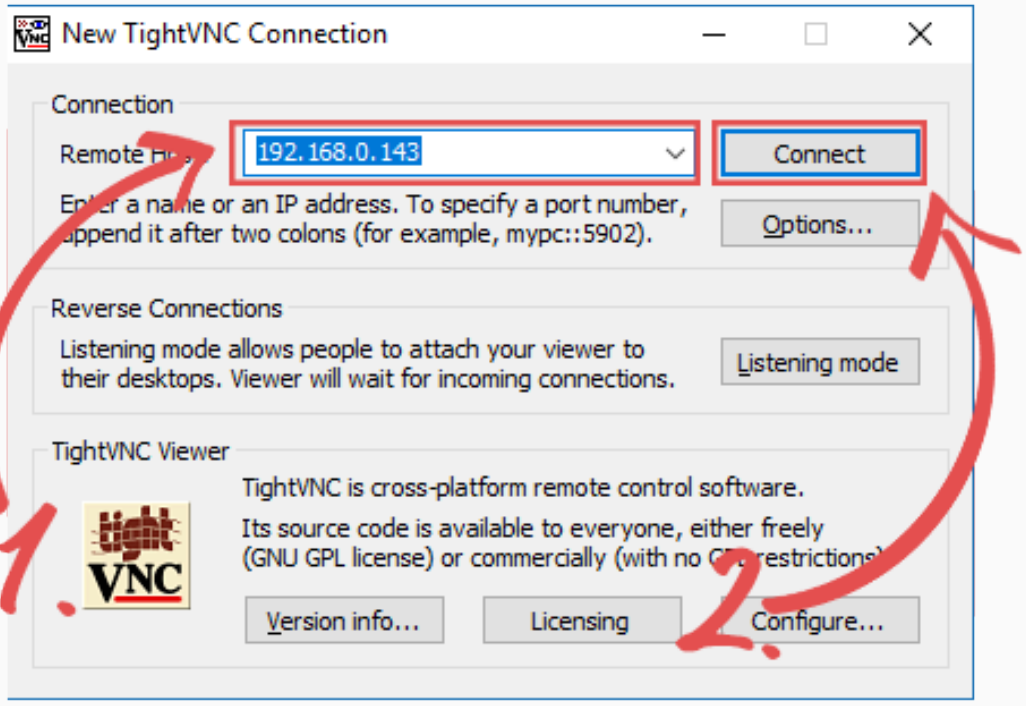
Testing Vino with a VNC Client

1. On the device, you want to use to connect to your Raspberry Pi download a compatible VNC Client. For our guide, we utilized the popular TightVNC client for Windows. You can find download TightVNC from their official website here: http://www.tightvnc.com/download.php

Once you have downloaded and installed a VNC Client to your computer, you can proceed to the next step of this guide.

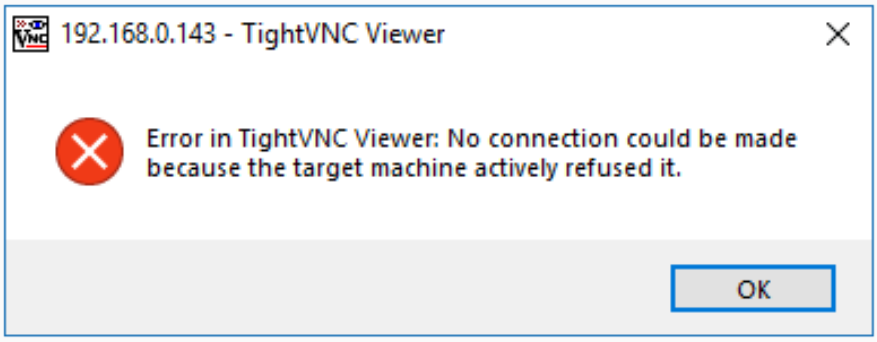
2. With TightVNC or any other VNC client now installed to your computer, launch it up. If you are using TightVNC, you should see a screen like what is shown below appear on your screen.

In the **text box** next to "**Remote Host:**" (**1.**) type in the IP address of your Raspberry Pi. Then simply press the "**Connect**" button (**2.**).



3. If a successful connection has been made, then you should now be seeing your Raspberry Pi's desktop.

If you encounter the error "**No Security Types Supported**" then go back to **Step 3** in our "**Installing Vino**" section and follow the steps for disabling encryption.



4. If you don't run into any errors and can see your desktop through the VNC client, then you can safely proceed onto the next section of this tutorial that shows you how to setup Vino, so it automatically starts at boot.

Starting Vino at Startup

1. Now getting Vino to startup at boot on your Raspberry Pi requires us to do a few things first. One of the first things we must do is create a script that will trigger the command to start up Vino Server when called.

We create this within the **sudoers.d** directory to ensure it will be launched with the correct rights for any user.

Start writing the script within the **/etc/sudoders.d/** by typing the following command into the terminal on your Raspberry Pi.

```
sudo nano /etc/sudoers.d/vinosrv.sh
```

2. Within this file type in the following two lines of code.

The first line just tells the interpreter that the operating system should try running the application **/bin/bash** to read the file. The second line executes the **vino-server** application.

```
#! /bin/bash
/usr/lib/vino/vino-server
```

3. Once you are sure, you have entered the correct two lines of code into the file you can save it by pressing the following. Press **Ctrl + X**, then **Y** and then press **Enter** to save the file.

4. Now we need to make the vinosrv file executable; this is so that on autostart the users can run the file.

To do this, you need to just type in the following command onto the terminal on the Raspberry Pi.

```
sudo chmod + x /etc/sudoers.d/vinosrv.sh
```

5. With the file now made executable, we will now add the file to the autostart for LXDE. We add it to this as it ensures that the Vino-Server will be loaded when the operating system's interface loads up.

Begin modifying the file by running the following command on the Raspberry Pi.

```
sudo nano /etc/xdg/lxsession/LXDE-pi/autostart
```

6. To this file add the following line to the bottom, this will ensure that Vino server is loaded at the startup of the operating system's interface.

```
@/etc/sudoers.d/vinosrv.sh
```

7. Your file should end up looking like something below with the addition of the line above.

```
@lxpanel --profile LXDE-pi
@pcmanfm --desktop --profile LXDE-pi
@xscreensaver -no-splash
@point-rpi
@/etc/sudoers.d/vinosrv.sh
```

8. You can now save the file by pressing **Ctrl + X** then **Y** and then **Enter**.

Starting Vino at Startup

9. Now try restarting your Raspberry Pi by running the following line on your Raspberry Pi.

```
sudo reboot
```

10. With the Raspberry Pi rebooted try making a connection using a VNC client to your Raspberry Pi.

If successful you have finished setting up Vino on your Raspberry Pi, and successfully have screen sharing up and running.

How to setup
Terminal Sharing

Project Description

In this tutorial, we are going to show you how to utilize a software package called tmate so that you can share your terminal with other computers, even when you are behind a firewall.

While the Raspberry Pi comes with VNC which allows you to share your desktop with other computers, it does not come with any methods to just share access to the terminal. That can be an issue when you are running a headless system as VNC will not be of any use.

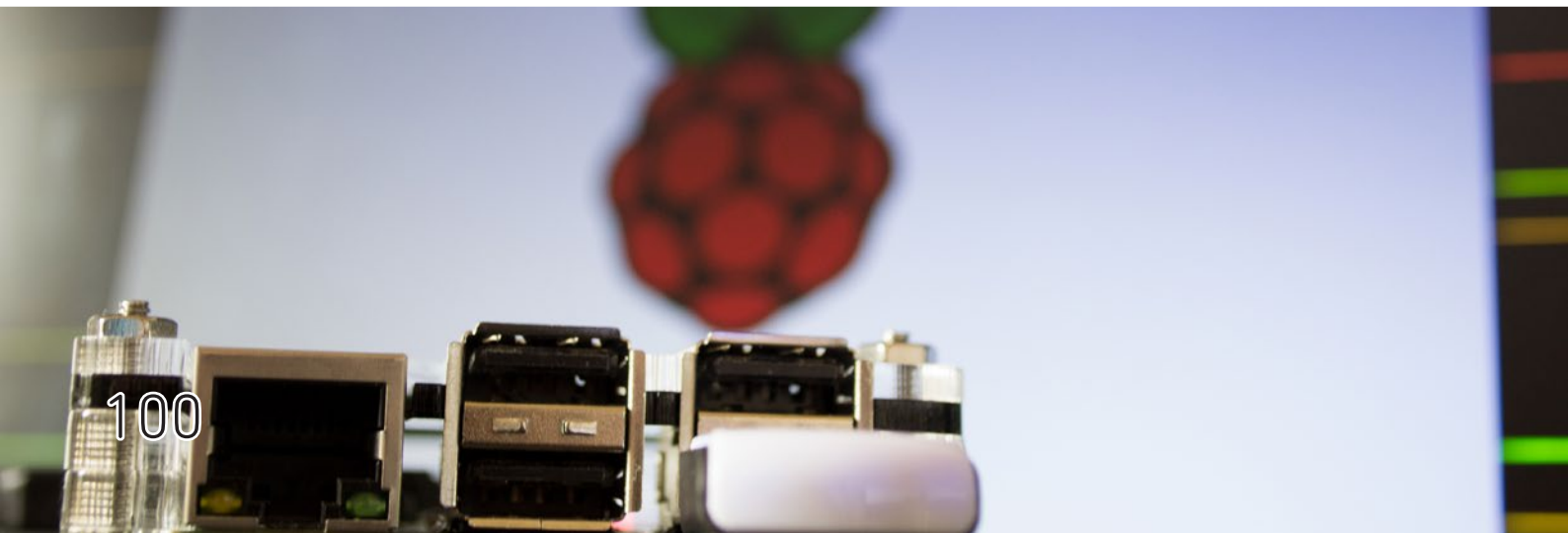
Our solution to this is to utilize a program designed to share the terminal. In our tutorials case, this is tmate.

This tmate application allows you to share access to the current terminal session with anyone you share the special key with regardless of if you are behind a firewall. Tmate works by relaying it through tmates own secure servers.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case



Installing tmate on Raspbian

1. Before we get started with installing tmate on our Raspberry Pi, we must first ensure our Raspberry Pi is entirely up to date. To do this type in the two commands below into terminal:

```
sudo apt-get update
sudo apt-get upgrade
```

2. Now it's our turn to install tmate, luckily unlike some other distributions tmate is already available in the Raspbian package repository, this is simple as running the following command on your Raspberry Pi.

```
sudo apt-get install tmate
```

3. Before we go ahead and run tmate we must first generate some SSH keys, the reason for this is that tmate uses them to encrypt the connections.

To generate an SSH key for your Raspberry Pi simply run the following command within the terminal. This command will create an SSH key based on the RSA encryption method with a size of 4096 bits.

```
ssh-keygen -t rsa -b 4096
```

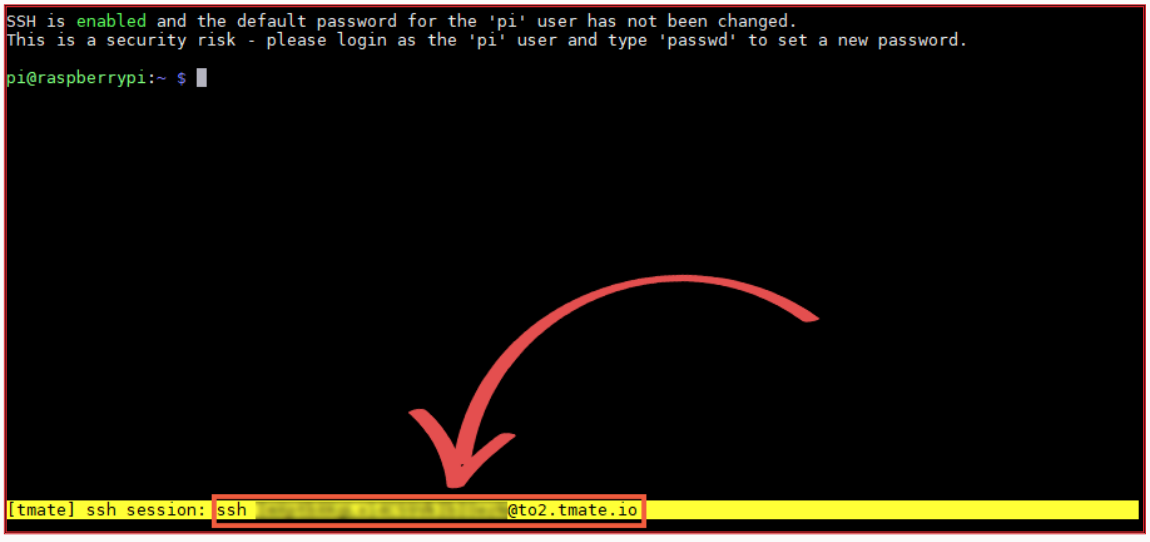
4. Now that we have generated an SSH key for our Raspberry Pi we can now proceed to run tmate. To run tmate, you just need to run the following command.

```
tmate
```

5. Upon running tmate, your SSH session should change, and you should see a clear session as we have shown below. You will also have a yellow bar that shows the current status of tmate.

As we have shown below, the initial connection should end up showing you a URL to utilize for the other computer to connect to the SSH session. We will show you over the page a couple of ways of utilizing it if you are unsure.

This URL should appear like **ssh xxxxxxxxxxxxx@to2.tmate.io**, where **xxxxxxxxxxxxx** is a random combination of letters and numbers. Copy the URL as you will need this later to connect to the tmate session.



6. Once you are finished with your tmate session, you can stop it by simply typing in "**exit**" and pressing the **Enter** key.

Utilizing tmate from anywhere

1. One of the easiest ways to connect to a tmate session is to make use of their web terminal. To do this just take the URL you grabbed from creating your tmate session.

The URL should be something like that we have shown below.

```
xxxxxxxxxxxxx@to2.tmate.io
```

2. From this URL you will want to take everything before the **@** symbol. So you should end up with something like below, where the x's are your unique session ID.

```
xxxxxxxxxxxxx
```

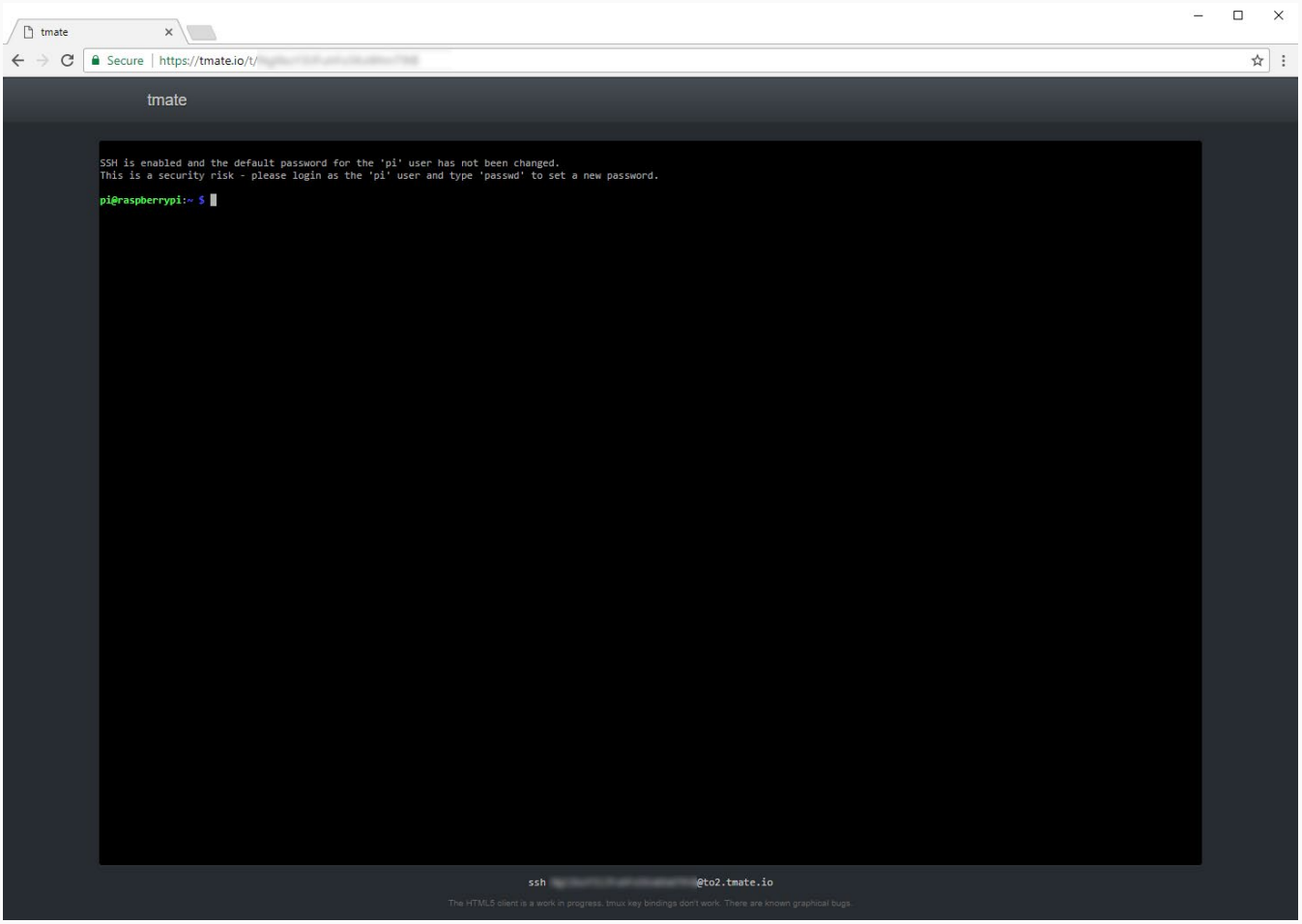
3. Now with your unique ID in hand, all you need to do is just go to **https://tmate.io/t/** with your unique ID added to the end of it.

So your URL should end up something like what we have shown below.

```
https://tmate.io/t/xxxxxxxxxxxxx
```

4. You should see something like below appear in your web browser upon a successful connection. If it appears, then you are ready to start utilizing your remote connection.

One extra note that we should mention, we highly recommend that you change the default password of the Pi user before proceeding with this, for simplicity sake we did not do this in our tutorial.



Utilizing tmate from a Linux system

1. Utilizing tmate from a Linux based system is probably the second most straightforward ways to use tmate but is likely more of a secure solution then tmate's web-based terminal. You can also use Windows 10's Linux subsystem to do this as well.

To begin with, we must first generate an SSH key for the local user that we plan on using to connect to our remote tmate session. Without it, tmate will refuse the connection.

Start off by opening up a new terminal on your device and type in the following command.

This command will generate a public/private RSA key pair and will be used to help secure the connection with tmate. If you already have created this in the past, then skip this step.

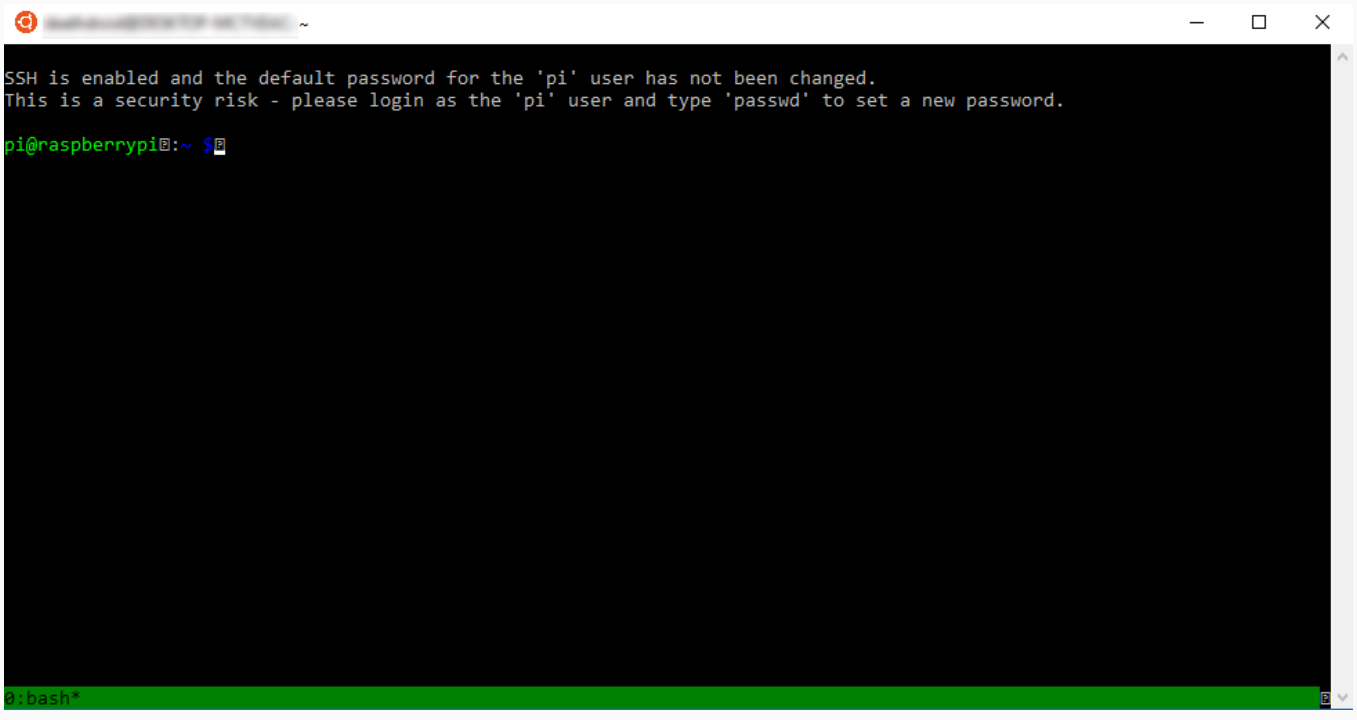
```
ssh-keygen
```

2. With the key now generated all we need to do to connect to the tmate session is utilize that URL that we obtained in the first section of this tutorial.

On the Linux based system of your choice simply type in ssh followed by the URL as we have shown below.

This command will immediately make the connection to your remote terminal session.

```
ssh xxxxxxxxxxxxx@to2.tmate.io
```

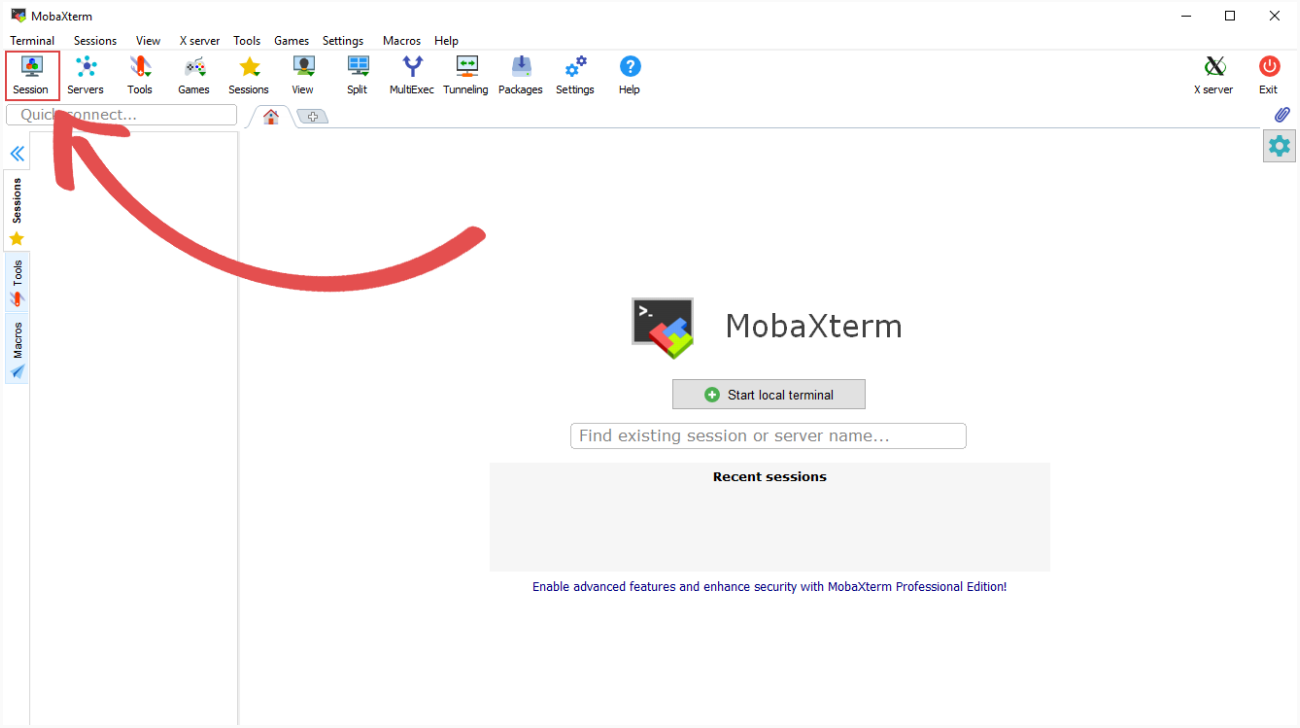


Utilizing tmate from a Windows system

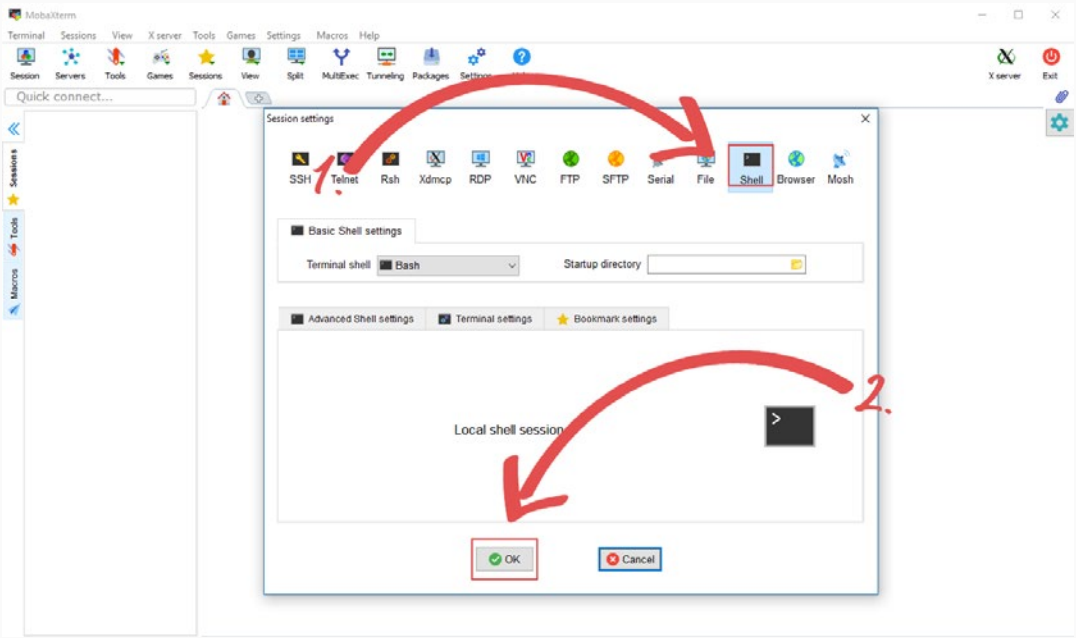
Utilizing tmate from a Windows system is slightly more complicated as it currently does not have a built-in SSH client like Linux/Unix based systems. Now here you can use a program such as Putty or MobaXTerm.

In this guide, we are going to be showing you how to connect to your remote terminal session by utilizing MobaXTerm as that is our current SSH client of choice.

1. Begin by opening up MobaXTerm on your computer and click the "**Session**" button.



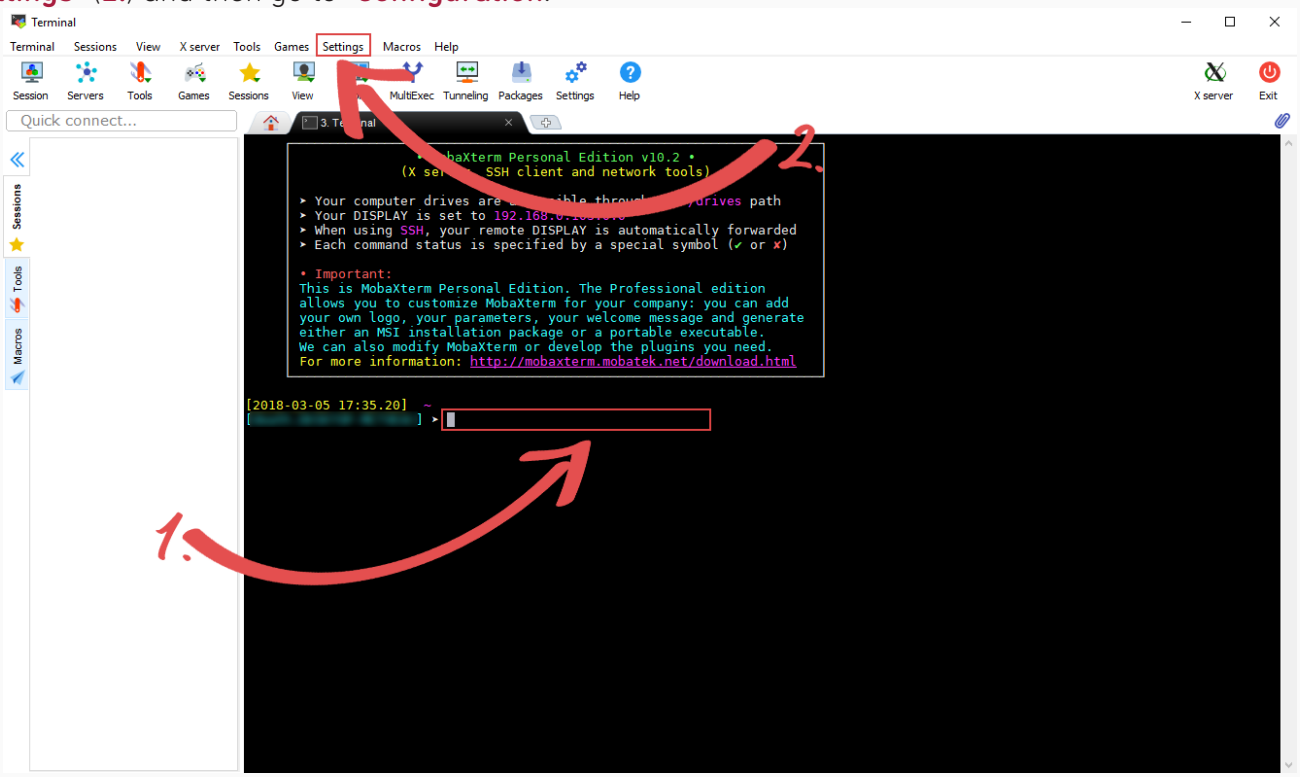
2. On this screen select the "**Shell**" tab (**1.**), then just click the "**Ok**" button at the bottom of the screen. There is no need to make any changes to any options on this screen.



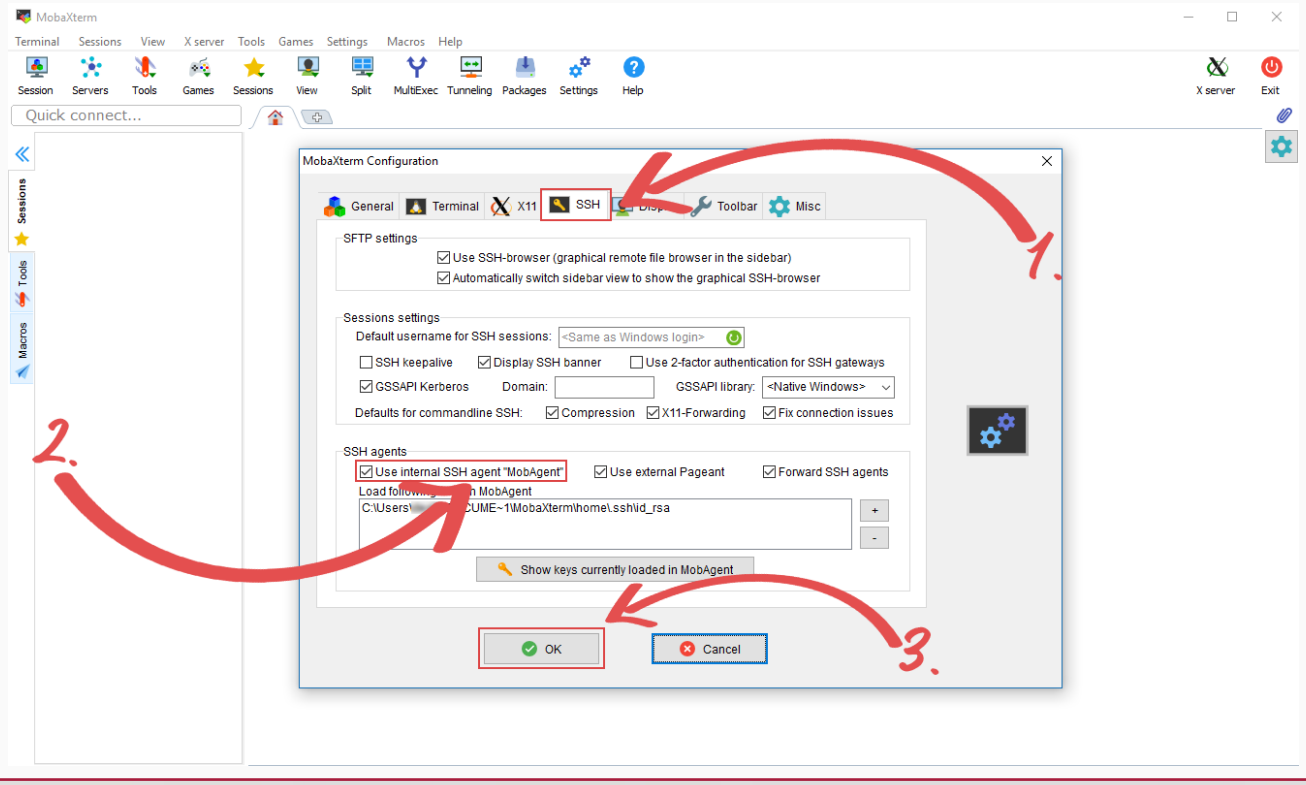
Utilizing tmate from a Windows system

3. Now that we are in the local terminal we just need to enter the following command. Just type in **ssh-keygen** into the terminal (**1.**) and press **Enter**.

This command will generate the SSH keys that we need. After you have done that you need to click "**Settings**" (**2.**) and then go to "**Configuration**."



4. Now that we are in the configuration screen, we need to go to the **SSH** tab (**1.**)
Within here make sure "**User Internal SSH agent**" is checked (**2.**)
Finally, click the "**OK**" button (**3.**)

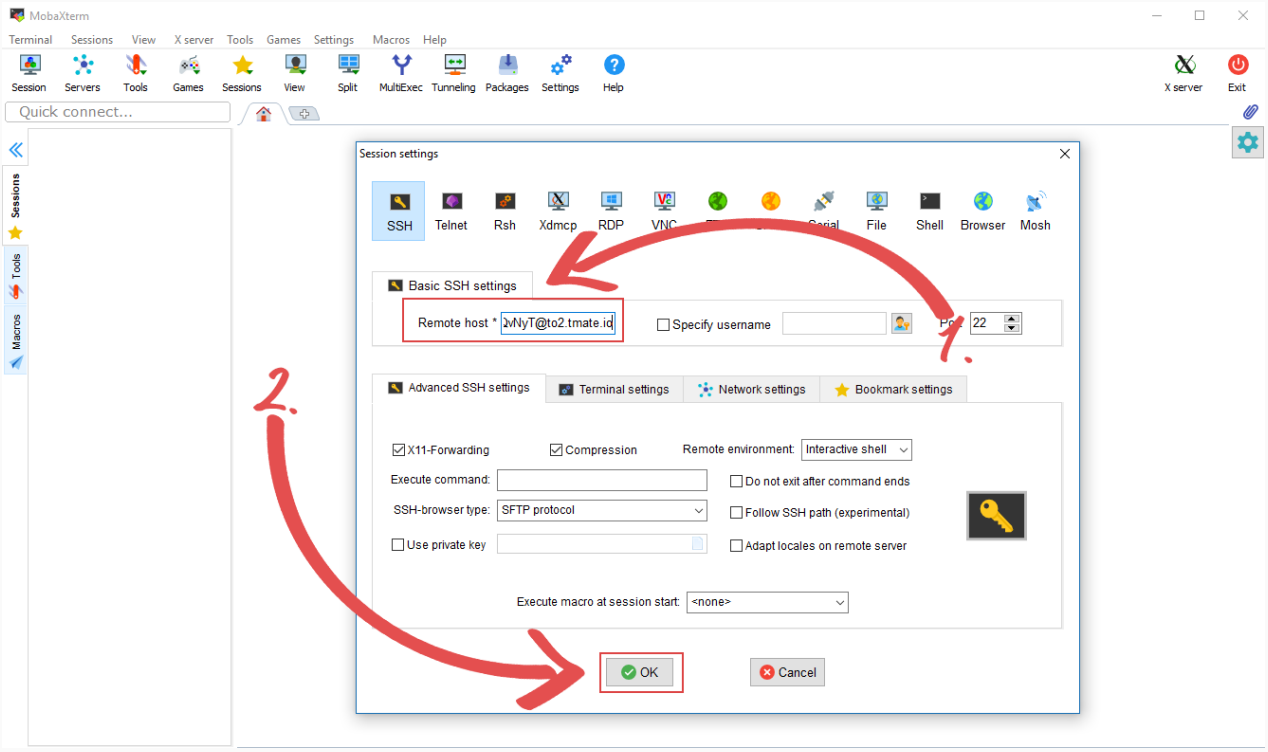


Utilizing tmate from a Windows system

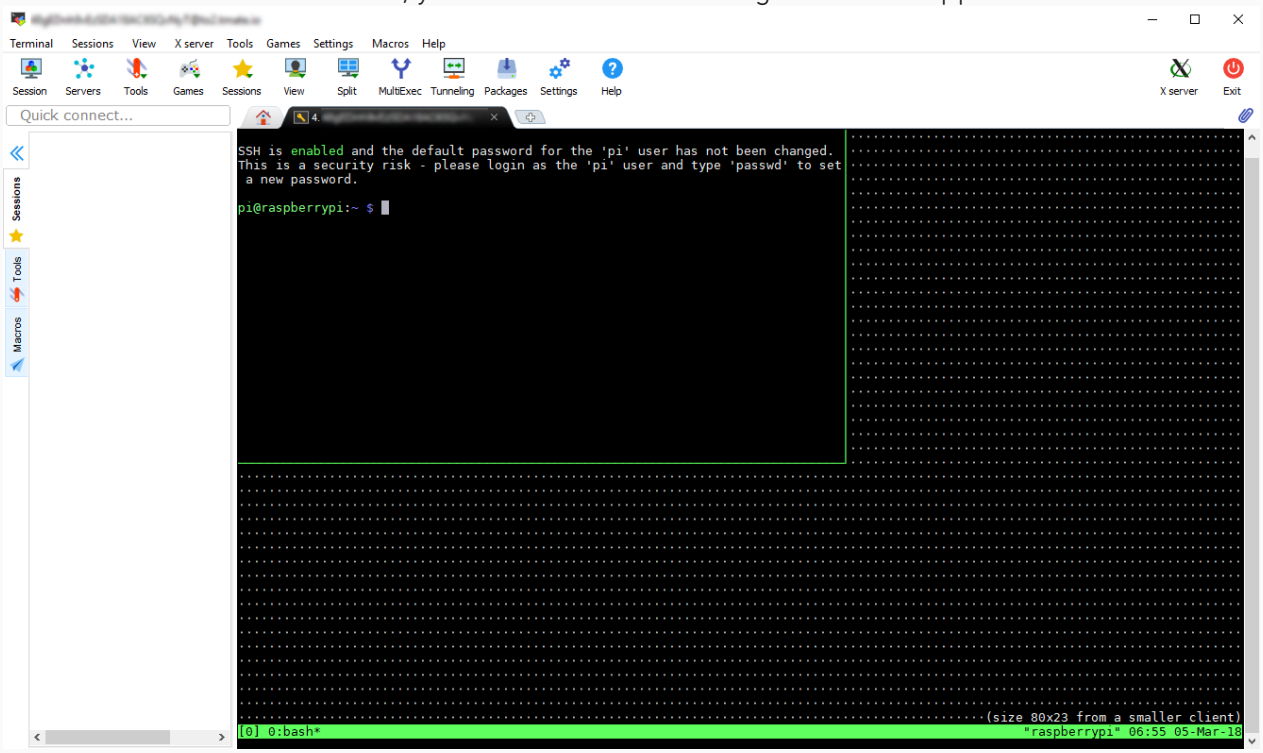
5. With the SSH key now generated and the internal SSH agent enabled we can now go ahead and connect to our tmate session. You just need to copy and paste the URL created for you into the remote host textbox (**1.**)

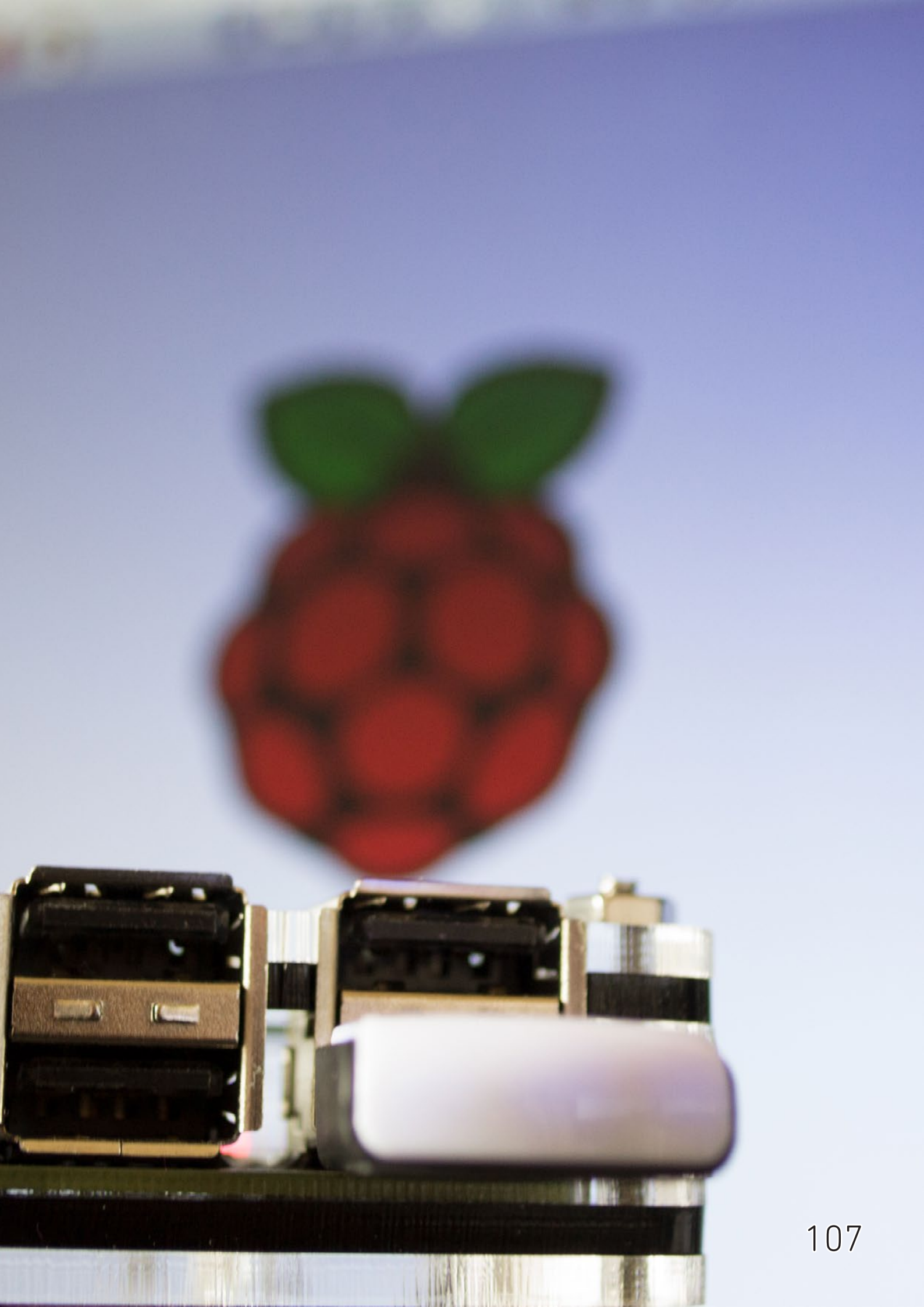
The tmate URL should look something like "xxxxxxxxxxxxx@to2.tmate.io."

With the URL entered into the text box all you need to do is press the "**OK**" button (**2.**).



6. Once the session has connected, you should see something like below appear in the terminal.





Setting up FTP
On the Raspberry Pi

Project Description

In this tutorial, we will go through the all the steps to set up FTP on a Raspberry Pi.
Once we have done this, you will be able to access all the files on your Raspberry Pi over FTP.

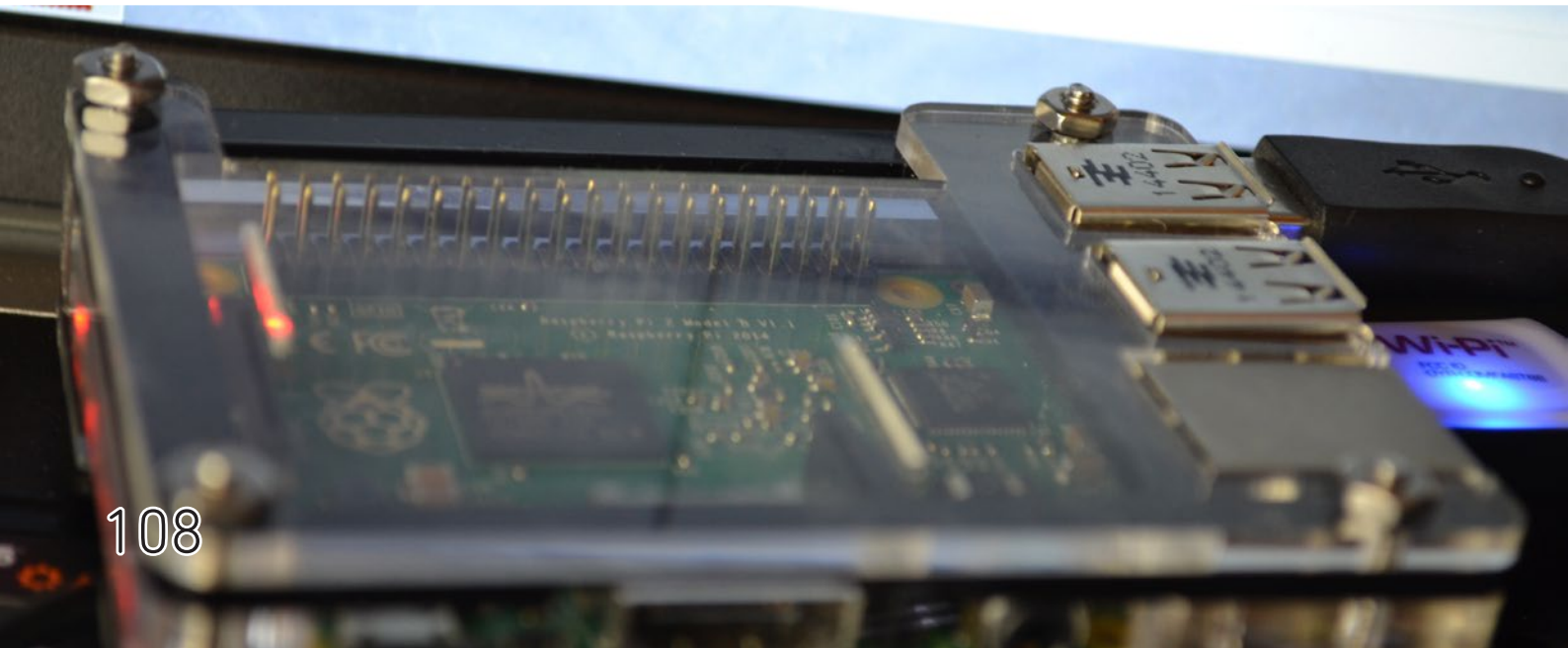
Please note that if you have enabled SSH, you can already access your files by using SFTP on port 22. SFTP is a lot more secure than plain FTP, so it is recommended to utilize it where possible.

However, if you are intent on using FTP our guide will explain how to set it up and access it from a computer using an FTP client such as the popular FileZilla.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case



Setting up the FTP server on the Raspberry Pi

1. Start off by entering the following commands to install the FTP server we intend on using.

```
sudo apt-get update
sudo apt-get install vsftpd
```

2. Once that has finished installing, we now need to edit the config file to setup the vsftpd FTP server properly. Use the following command to begin editing the file.

```
sudo nano /etc/vsftpd.conf
```

3. In this file, you will need to add or uncomment the following lines by removing the **#** in front of the line.

These lines set up the basic settings for the FTP server.

```
anonymous_enable=NO
local_enable=YES
write_enable=YES
local_umask=022
chroot_local_user=YES
user_sub_token=$USER
local_root=/home/$USER/ftp
```

4. Now we need to save the file by pressing **Ctrl+X** and then pressing **Y**.

5. We need to create the FTP directory so we can transfer files. The root directory for FTP is not allowed to have write permissions meaning you can't copy files to it so we will need a subfolder called **files**.

Replace **<user>** with the relevant user. For instance, if you are using the default Raspbian user, replace it with **pi**

```
mkdir /home/<user>/FTP
mkdir /home/<user>/FTP/files
chmod a-w /home/<user>/ftp
```

6. Now we need to restart the FTP service, so it loads in the settings by entering the following command:

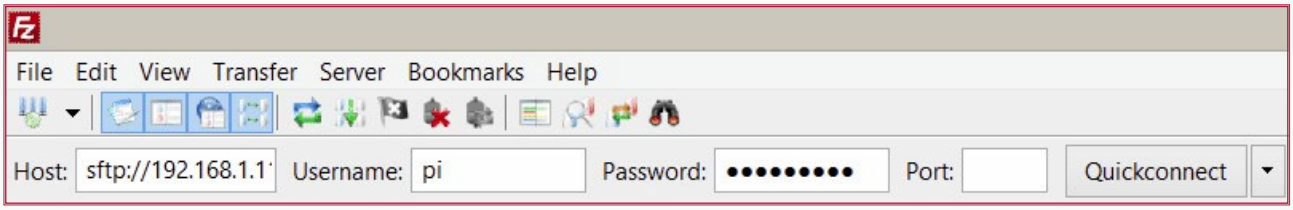
```
sudo service vsftpd restart
```

7. You should now be able to connect to your Raspberry Pi's FTP using port 21 now.

Connecting to the FTP Server

1. On your computer, head to **https://filezilla-project.org/download.php?show_all=1** and download and install the FileZilla client for your operating system.

2. Now enter the Raspberry Pi's IP address, username, password, and port 21.



3. Now all you need to do to connect is to press the **Quickconnect** button. Assuming you have entered your details correctly, it should now successfully log in.

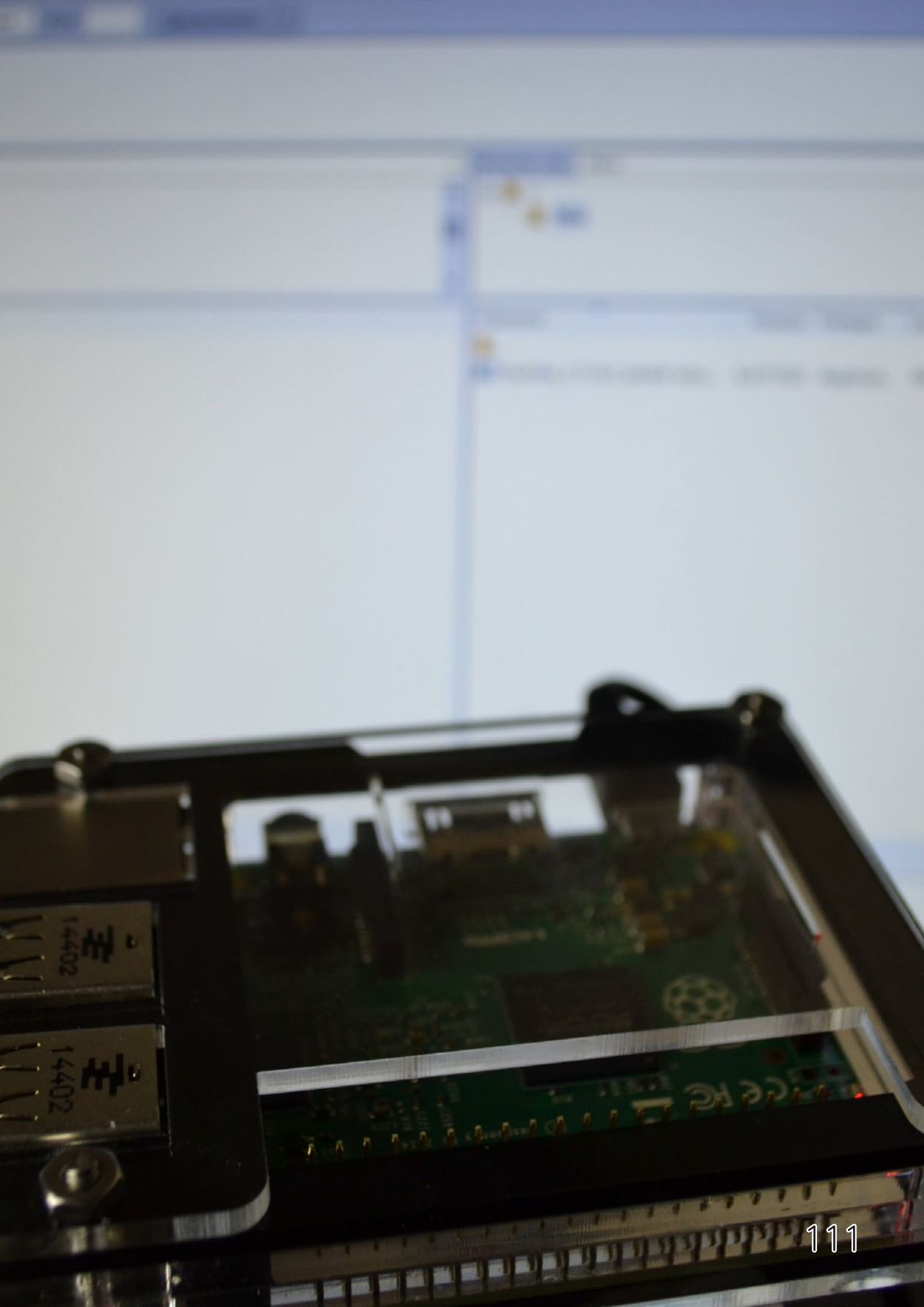
4. Test to make sure that it is correctly working by transferring some files to the Raspberry Pi. If you are having issues make sure you have correctly entered the changes in **vsftpd.conf** as mentioned on the previous page.

Permission denied: This means you are trying copy files to or from a place where your current user doesn't have the correct permissions.
To fix you will need to change the permission of that location or login with a different user.

Troubleshooting the Raspberry Pi FTP Server

You can connect to the Raspberry Pi's FTP server by using any standard FTP client. For this guide, we are going to recommend using FileZilla however you can use any of the vast numbers of FTP clients that are out there.

We choose FileZilla as it is actively developed and has support for a variety of different operating systems.



Installing Visual Studio Code On the Raspberry Pi

Project Description

In this Raspberry Pi visual studio code (VS Code) tutorial I go through all the steps to installing this handy code editor.

If you're using the Raspberry Pi as a desktop and want a decent program to edit and create code, then visual studio code is perfect. It is feature packed and contains basically everything you will need to make your very own programs.

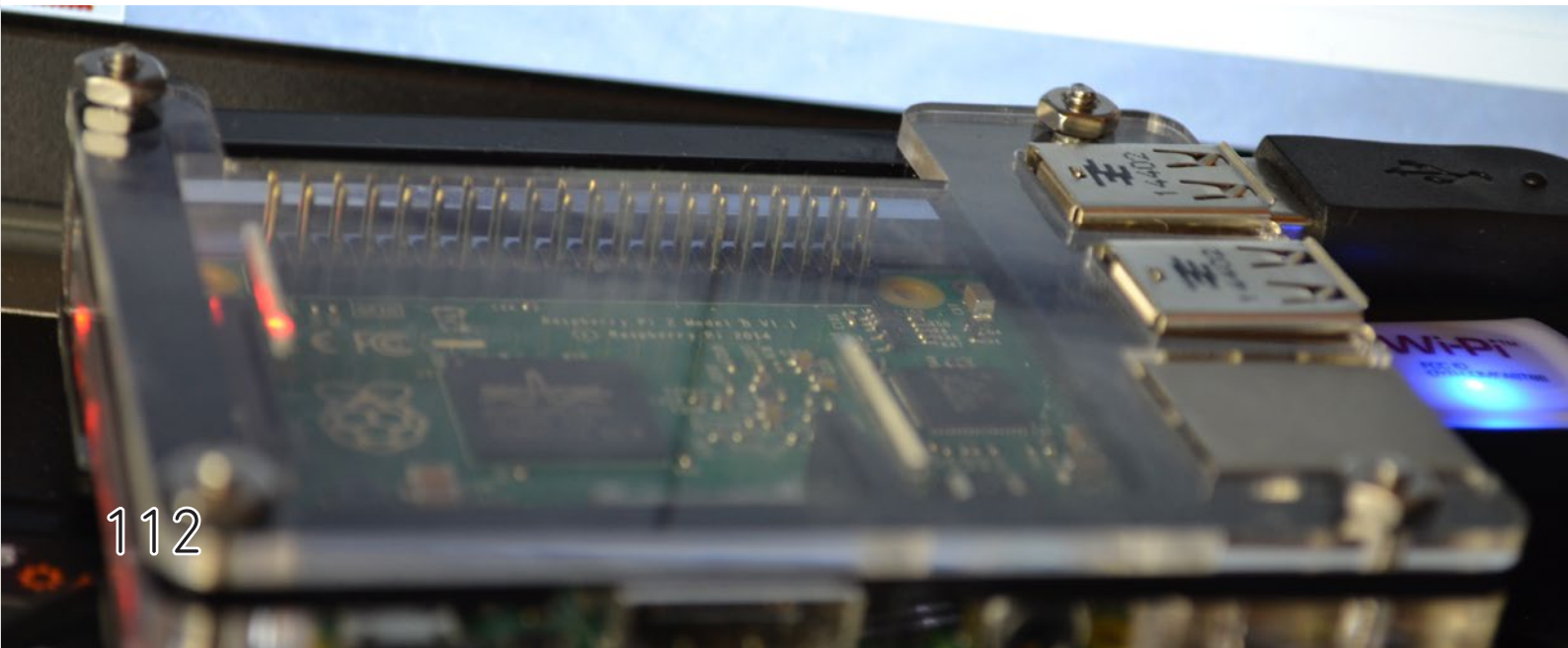
I recommend using a Git service such as a self-hosted Git, GitHub or even Gitlab. These software packages are not just great for version control but also making sure that your code is backed up and readily available.

There are many other code editors that you can use on the Raspberry Pi. Each of the editors has their own pros and cons. I find VCS one of the best that you can easily install on the Raspberry Pi.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case



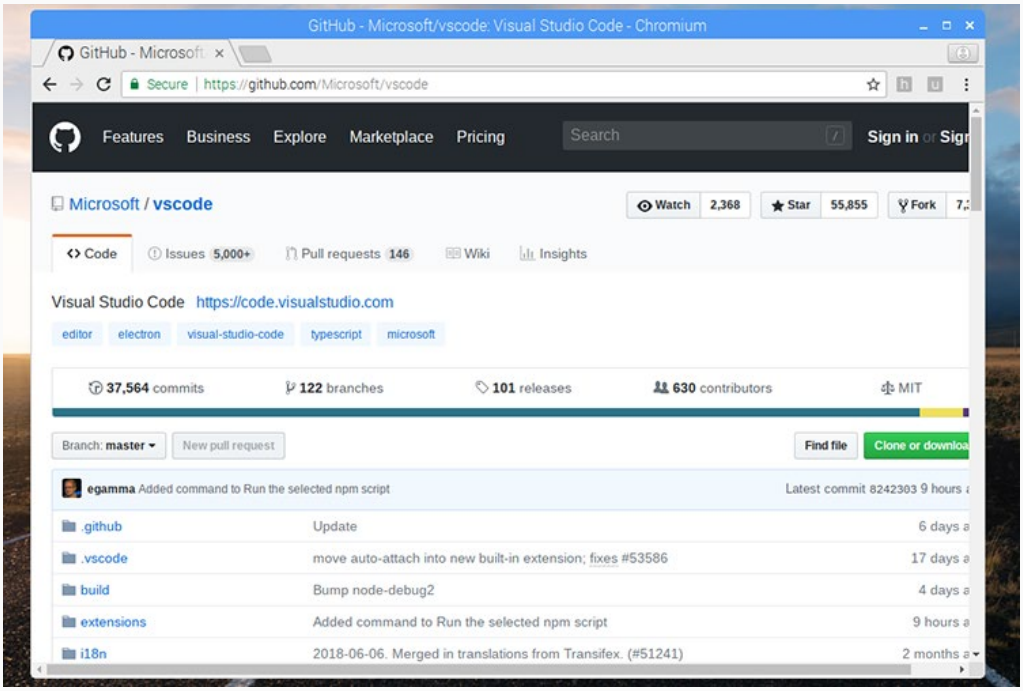
Main Features of Visual Studio Code

Many features of Visual Studio Code makes it stand out from alternatives. One of the best parts is that it's completely free and very customizable.

Open Source & Free

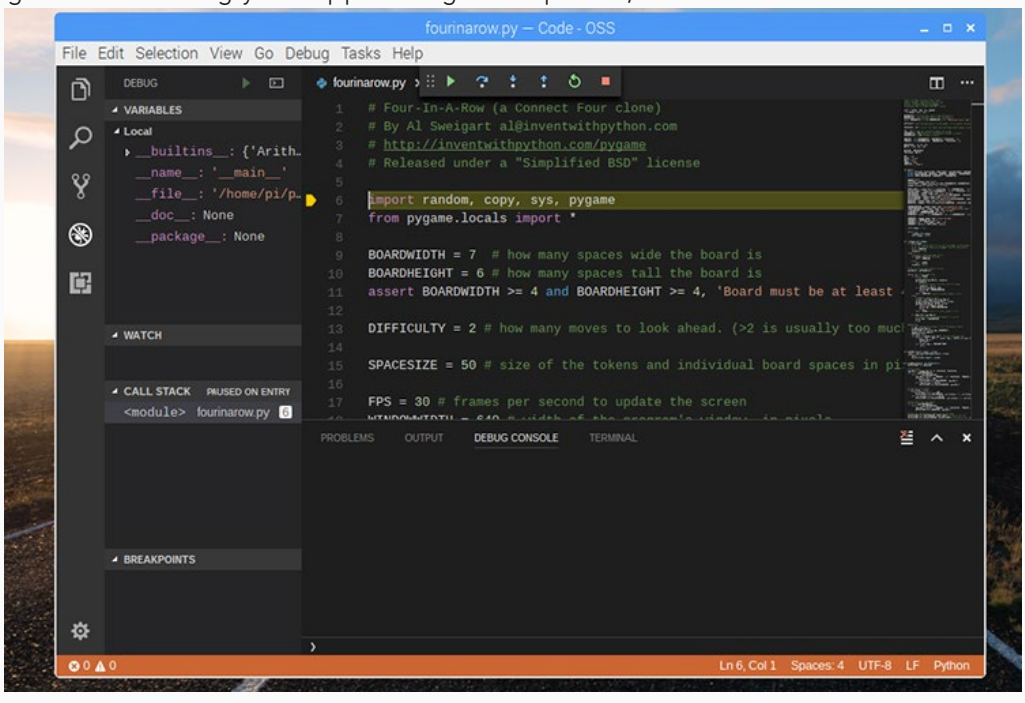
An exciting part of VS Code is that it is completely open source and you can view all the code on GitHub at https://github.com/Microsoft/vscode. It is registered under the MIT license iving you flexibility if you wanted to fork and edit the code.

You can also check the Raspberry Pi build on GitHub at https://github.com/headmelted/vscode-arm. These two Github's are important if you wanted to check out what you're about to install.



Debugging Capabilities

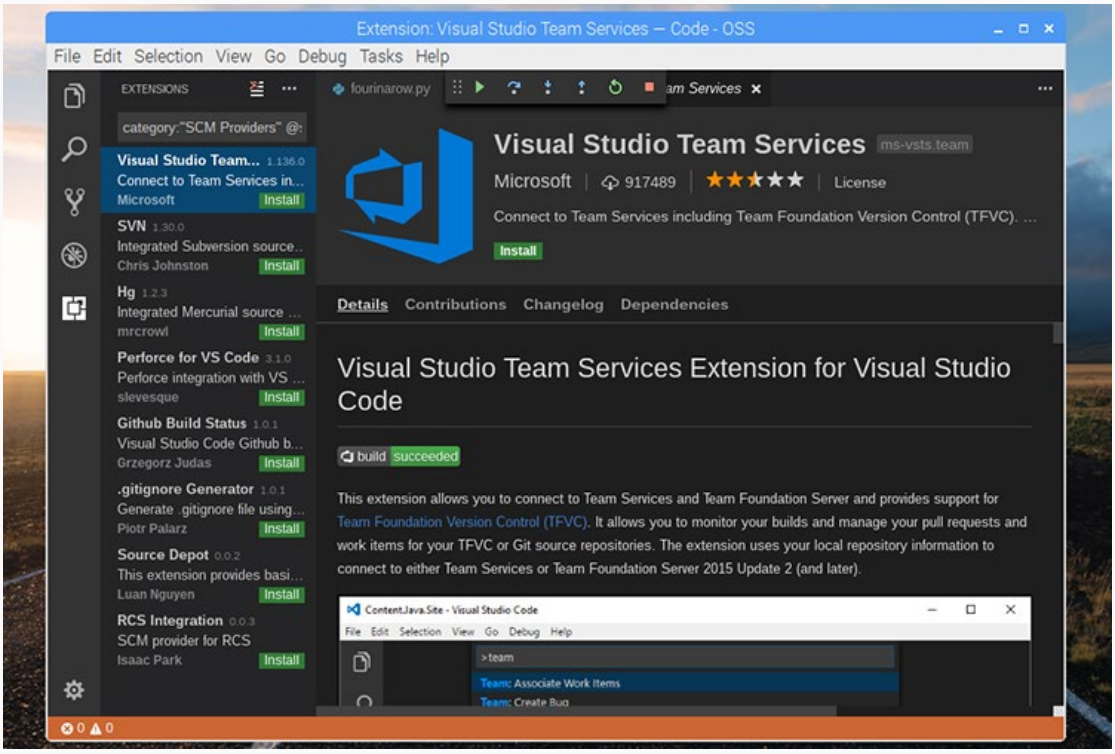
You can debug your code in the editor which helps reduce the amount you need to rely on print statement debugging. You can debug your apps using breakpoints, call stacks and an interactive console.



Main Features of Visual Studio Code

Built-in Git

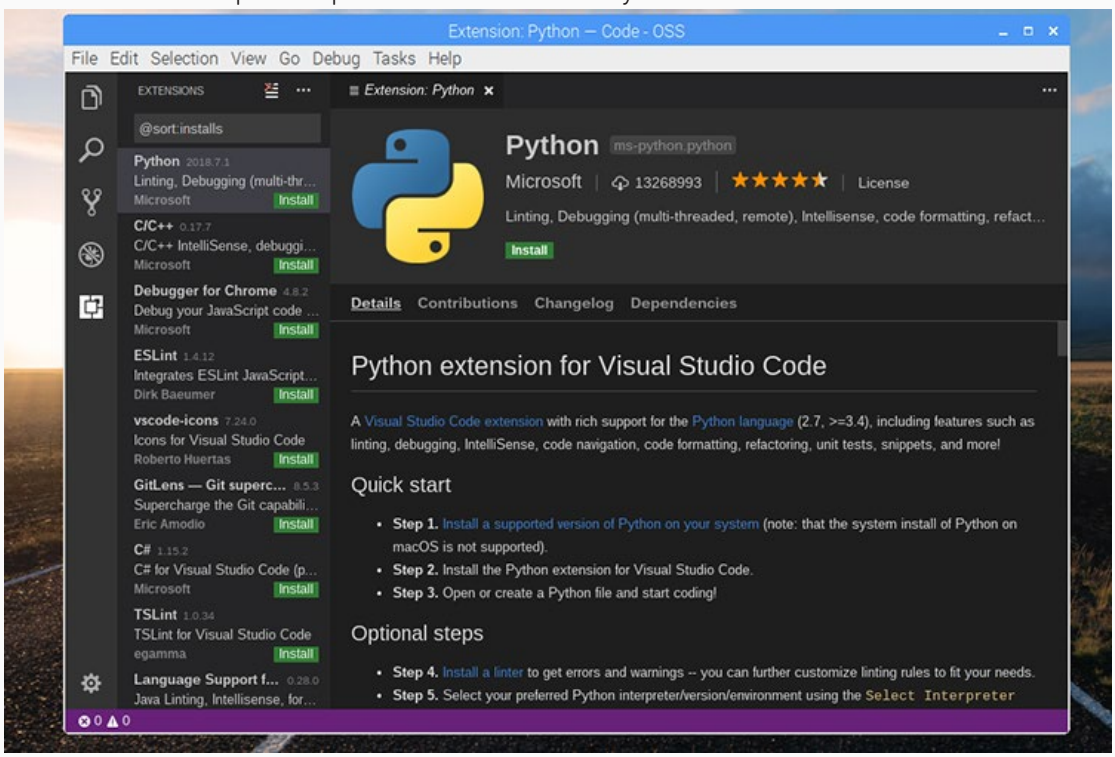
This feature is super handy if you use version control a lot. It works with Git and other SCM (Software Configuration Management) providers and allows you to review diffs, stages and commit files from within the editor.



Extensible and Customizable

One of the coolest features of Visual Studio Code is how extensible. For example, you can install new language packs, themes, debuggers, formatting tools, and so much more.

All your extensions are in separate processes so that they won't slow down the editor.



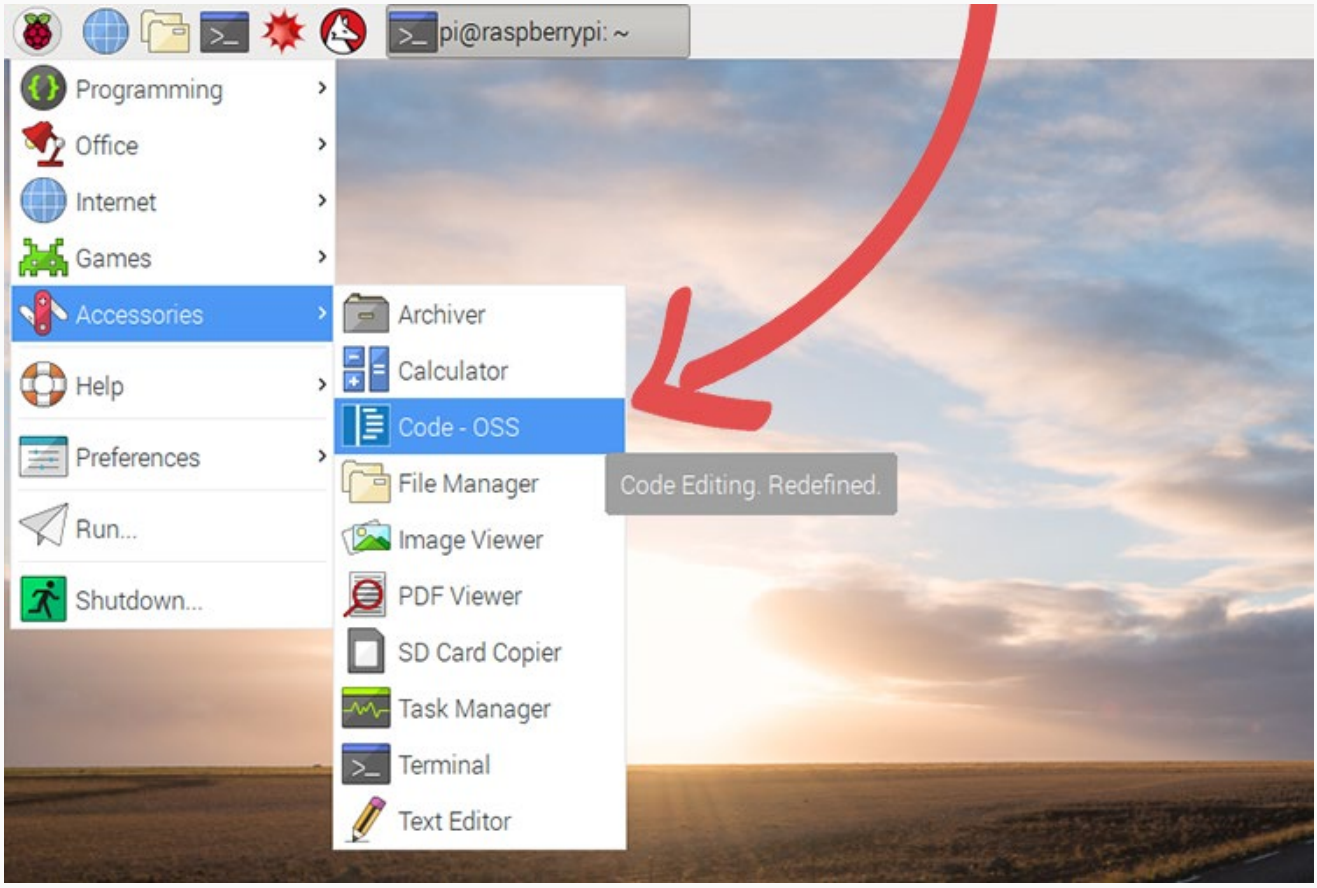
Installing Visual Studio Code

Jay aka Headmelted, provides builds of Visual Studio Code that can work on the less popular platforms. This range includes operating systems such as ChromeOS, Raspbian, Linux Mint, Fedora and more.

To install visual studio code, you only need to run a straightforward command.

```
curl -L https://code.headmelted.com/installers/apt.sh | sudo bash
```

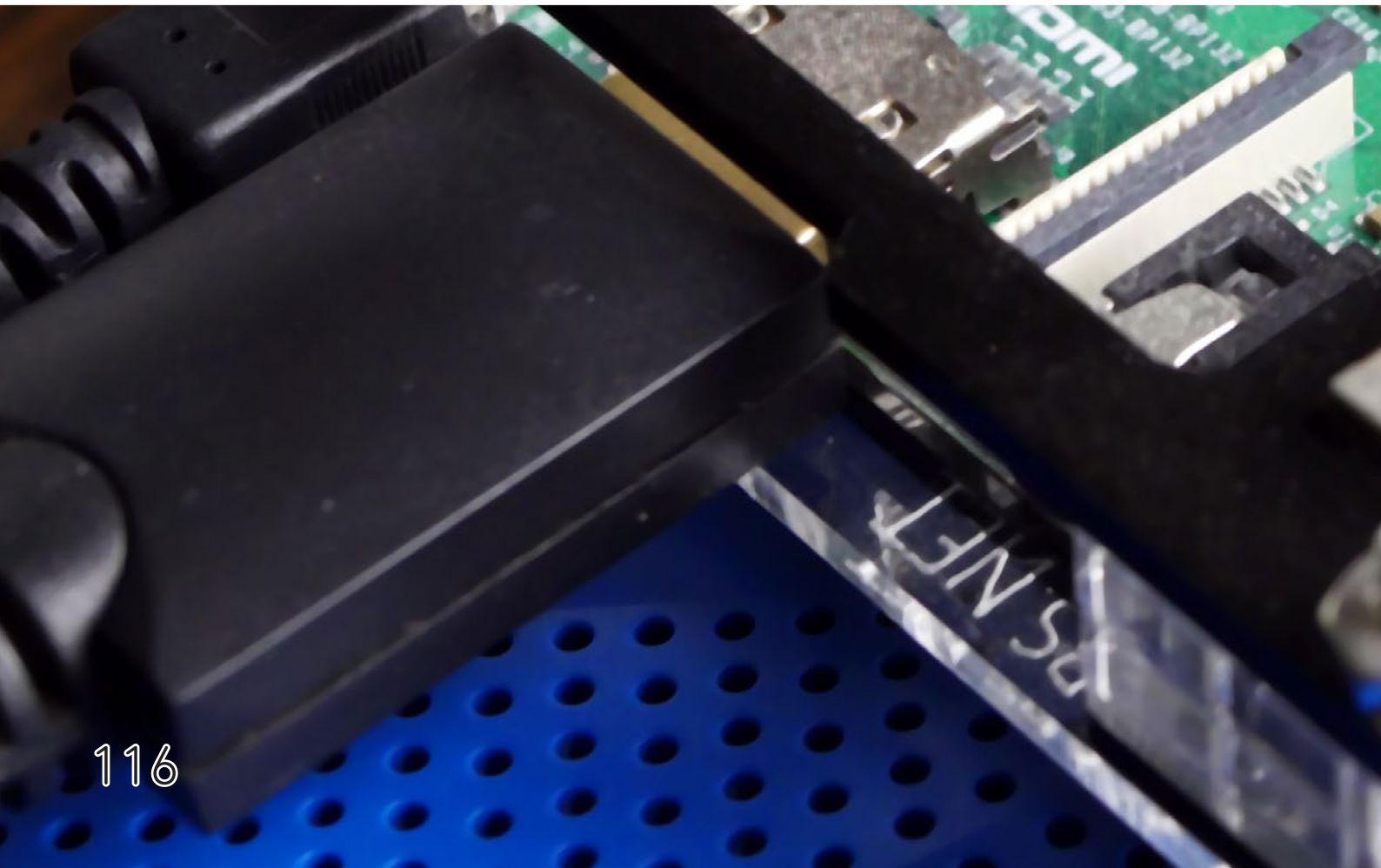
Once done you should be able to find the Visual Studio Code under the accessories menu called Code-OSS.

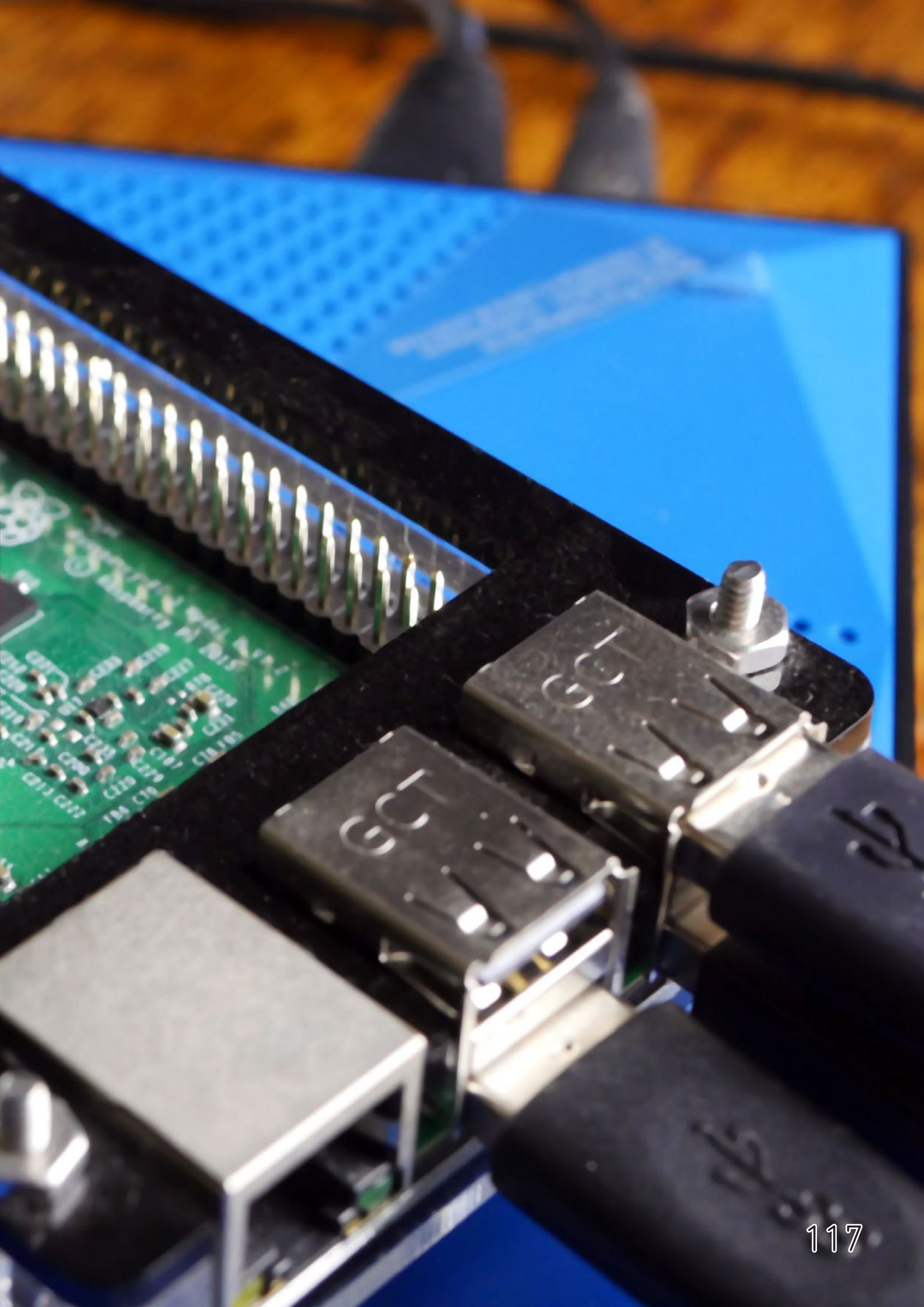


You can now start coding to heart's desire on the Raspberry Pi. You should realize quickly why Visual Studio Code has become a favorite amongst many avid programmers.

I hope that this tutorial has shown you all the steps to set up this code editor.

Dealing with Storage





Partitioning & Formatting Drives

Project Description

In this guide, we will be showing you how to partition and format a drive on a Linux based operating system.

Within this tutorial, we will be showing you how to use three crucial pieces of software that allows you to create and modify partitions quickly. These three pieces of software are **blkid**, **parted** and **mkfs**.

Parted is the software that we will be using to create the partitions on a drive, while we don't go too deep into all the capabilities of the software we do show how you can efficiently use it to create partitions on a drive. One thing to note about **parted** is that it doesn't format the partitions you create, for that, you have to use a piece of software called **mkfs**.

The next most important piece of software we show to you is **mkfs**, which is the actual piece of software that we will utilize to format our newly created partitions.

We also show you **blkid**, which is a handy piece of software that can be used to display all available block devices quickly.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- USB Drive

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case
- Network Connection

Partitioning a drive on Linux via commandline

1. For most of this tutorial, we will be making use of a piece of software called parted. Parted is the CLI (Command-line interface) version of its GUI brother, gparted.

Parted will allow us to do numerous things such as being able to format drives as well as creating different partitions.

Before we get too far ahead of ourselves, we must first work out what drive we want to format. For that purpose, we can use a tool such as blkid to find the drive we want to format.

Enter the command below to get information on your available drives.

```
sudo blkid
```

Below we have included a sample of what the command provided for us, the drive we are interested in is located at the bottom of the list mounted at **/dev/sda**.

```
/dev/mmcblk0p1: LABEL="boot" UUID="6228-7918" TYPE="vfat" PARTUUID="76b5801c-01"
/dev/mmcblk0p2: LABEL="rootfs" UUID="6bfc8851-cf63-4362-abf1-045dda421aad" TYPE="ext4" PARTUUID="76b5801c-02"
/dev/mmcblk0: PTUUID="76b5801c" PTTYPE="dos"
/dev/sda: PTUUID="09c74626" PTTYPE="dos"
```

2. Now that we have our mount location for our USB drive available to us we can go ahead and start interacting with it using the **parted** software.

Now before we go ahead and start formatting our drive, we must first create the partition table for the drive. If your drive already has a partition table you can skip to the next step.

To create a partition table we can use a straightforward command, we need to reference the mount path, in our case **/dev/sda**, the **mklabel** command and the type of partition table we want to create.

In this case we are going to be using the more modern **GPT** table type, however, if you want or need you can utilize the much older **msdos** partition table (It's also known as MBR (Master Boot Record))

To set this, run the following command. Swapping out the mount path and table type where required.

```
sudo parted /dev/sda mklabel gpt
```

If a boot label already exists you will be shown a warning asking if you want to continue.

3. To create a partition on our new drive, we need to know what file format we intend on using, and the starting and ending position for that partition.

For example, if we wanted to create a 2GB partition on our empty 4GB USB, we would set the start position at 0 and the end position at 2GB.

Also if you want the partition to cover a percentage of the drive or the whole drive, you can specify sizes in %. For instance, 100% would fill the remaining space after the starting position.

Since we want to do that with our USB, we will be running the following command on it. Referencing its mount position of **/dev/sda**, the filesystem we want to use, **fat32**, and our starting and ending positions for the partition.

The parted tool also supports other filesystems such as **ext2**, **ext3**, **ext4**, **fat16**, **fat32**, **NTFS** and more.

```
sudo parted /dev/sda mkpart primary fat32 0G 2G
```

Partitioning a drive on Linux via commandline

4. Now that we have created our first partition on the drive the partition should now be displayed using the blkid tool.

Just rerun the following command in your terminal on the Raspberry Pi to list all available partitions.

```
sudo blkid
```

Below you can see what the output looks like now, with our partition at **/dev/sda1** now being shown.

```
/dev/mmcblk0p1: LABEL="boot" UUID="6228-7918" TYPE="vfat" PARTUUID="76b5801c-01"
/dev/mmcblk0p2: LABEL="rootfs" UUID="6bfc8851-cf63-4362-abf1-045dda421aad" TYPE="ext4" PARTUUID="76b5801c-02"
/dev/mmcblk0: PTUUID="76b5801c" PTTYPE="dos"
/dev/sda1: PARTLABEL="primary" PARTUUID="47470959-981d-4fe9-bb58-84bafeddbab5"
```

5. With our first partition now created we still have 2GB of spare space available on our 4 GB drive.

We can now go ahead and format that partition using the various available mkfs tools if you don't need to know how to handle multiple partitions you can move onto our "Formatting a partition" section.

To help give you more of an idea on how to use parted we will make another three partitions, two 512 MB partitions, and one 1 GB partition.

One thing to remember when creating additional partitions is that your new partitions must allocate space that isn't already allocated. For example, the first partition we created filled from 0MB to 2 GB meaning that any additional partition must fill space after the first 2 GB of data.

The starting and ending positions are something that you will see more of in the three commands below.

Additionally our examples below we will be using the filesystem formats, NTFS, ext2, ext4. We are using a variety of filesystem formats to give you an idea of how each of these will be formatted.

```
sudo parted /dev/sda mkpart primary ntfs 2GB 3GB
sudo parted /dev/sda mkpart primary ext2 3000MB 3500MB
sudo parted /dev/sda mkpart primary ext4 3.5G 100%
```

6. Now that we have successfully created all our partitons lets go ahead and rerun the blkid tool to see our new partitions show up.

```
sudo blkid
```

Below is the output of the command.

```
/dev/mmcblk0p1: LABEL="boot" UUID="6228-7918" TYPE="vfat" PARTUUID="76b5801c-01"
/dev/mmcblk0p2: LABEL="rootfs" UUID="6bfc8851-cf63-4362-abf1-045dda421aad" TYPE="ext4" PARTUUID="76b5801c-02"
/dev/mmcblk0: PTUUID="76b5801c" PTTYPE="dos"
/dev/sda1: PARTLABEL="primary" PARTUUID="47470959-981d-4fe9-bb58-84bafeddbab5"
/dev/sda2: PARTLABEL="primary" PARTUUID="07e3d328-3fb7-4a09-8ded-cf74178bde2a"
/dev/sda3: PARTLABEL="primary" PARTUUID="82d3de54-07e2-42de-94b6-825bf8dcfe7a"
/dev/sda4: PARTLABEL="primary" PARTUUID="df1af8e1-5e8b-4909-b754-f3a54aa4c2ac"
```

7. Now that we are all done with our partitions, have them created, and can see them we should now proceed onto actually formatting these partitions. Move onto the section titled "Formatting partitions".

Formatting partitions on Linux via commandline

1. While we have now created all our partitions, we need to format them. The parted tool only creates the partitions themselves and sets a few bytes so that the system can understand what format to expect on that partition

To format these partitions we will be making use of a tool called mkfs, this tool is primarily designed to format partitions and has support for a wide range of filesystems.

Thankfully mkfs is very easy to use. Basically you need to call it with the filesystem type you want to use and the device that it should be formatting.

Let's begin by formatting our first partition. This partition is located at **/dev/sda1**. For this partition, we will be formatting it in the fat32 filesystem type. For mkfas this means we need to use **vfat** as the filesystem type.

To do this we can use a shorthand version of the mkfs command, you can see below that we simply need to call **mkfs.vfat** followed by the partitions mount location.

```
sudo mkfs.vfat /dev/sda1
```

2. As you can see with the previous command it is very easy to format your partitions, and is a much simpler process then creating partitions.

Below we will showcase a few other examples of some of the other filesystems you can format using this method.

```
sudo mkfs.ntfs /dev/sda2
sudo mkfs.ext2 /dev/sda3
sudo mkfs.ext4 /dev/sda4
```

3. Now that we have formatted all of our partitions we can now mount them if we choose to do so. If you want to verify that each partition has actually been formatted we can again make use of the blkid tool to show us all the partitions and their formatted filesystem type.

```
sudo blkid
```

Below you can see what the output looks like now, with our four different partitions being formatted with four different filesystem types.

```
/dev/mmcblk0p1: LABEL="boot" UUID="6228-7918" TYPE="vfat" PARTUUID="76b5801c-01"
/dev/mmcblk0p2: LABEL="rootfs" UUID="6bfc8851-cf63-4362-abf1-045dda421aad" TYPE="ext4" PARTUUID="76b5801c-02"
/dev/sda1: UUID="DCF5-DDF6" TYPE="vfat" PARTLABEL="primary" PARTUUID="d7e89b84-3817-4393-b348-75f47618dd3e"
/dev/mmcblk0: PTUUID="76b5801c" PTTYPE="dos"
/dev/sda2: UUID="656268AC53B5530D" TYPE="ntfs" PTTYPE="dos" PARTLABEL="primary" PARTUUID="07e3d328-3fb7-4a09-8ded-cf74178bde2a"
/dev/sda3: UUID="9887c8c7-f5f9-4440-9ec1-b84793877366" TYPE="ext2" PARTLABEL="primary" PARTUUID="82d3de54-07e2-42de-94b6-825bf8dcfe7a"
/dev/sda4: UUID="3d8b1983-4a46-4c7d-9c97-e4adb233dcb7" TYPE="ext4" PARTLABEL="primary" PARTUUID="df1af8e1-5e8b-4909-b754-f3a54aa4c2ac"
```

4. You should hopefully now have an idea on how you can use tools that come with the Raspbian operating system to create new partitions and also format those partitions in various filesystem types.

If you are interested in what various filesystems that are supported by default you can look at our list below. These can vary depending on what Linux distribution you are running, additionally you can add support for NTFS and exFAT by installing certain packages

Supported Filesystem types: msdos, bfs, cpm, ext2, ext3, ext4, minix, fat, vfat, hfs, vxfs, rf, rk, dec, NTFS.

Adding Support for exFAT
on the Raspberry Pi

Project Description

In this guide, we will be showing you how you can enable support for the exFAT filesystem format on your Raspberry Pi.

For those who do not know, exFAT (Extended File Allocation Table) is a proprietary filesystem format developed by Microsoft designed to be optimized for flash memory such as USB flash drives and SD cards.

Due to exFAT being patented by Microsoft it cannot become an official part of Linux despite a kernel implementation being released by Samsungs. Due to this, we must implement support for the exFAT filesystem another way.

That's where the FUSE kernel implementation comes in handy. FUSE will allow us to implement support for the exFAT format without having to recompile the kernel.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- USB Drive

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case
- Network Connection

Adding support for exFAT

1. Before we get started with adding support for the exFAT file system to our Raspberry Pi, we must first ensure that our Raspberry Pi's operating system is completely up to date and has the latest package list.

For us to update our Raspberry Pi and grab the package list, we need to enter the following two commands into its terminal by either using the terminal on the Pi itself or over SSH.

```
sudo apt-get update
sudo apt-get upgrade
```

2. Now that we have updated our Raspberry Pi and have grabbed the latest package list we can now install the packages we need.

To enable support for the exFAT filesystem on our Raspberry Pi, we will need to utilize two particular packages.

The two packages that we will be installing to our Pi is **fuse-exfat** and **exfat-utils**. These two packages work together to allow the operating system to talk to hard drives that are in the exFAT format.

Fuse-exfat works as a module to FUSE (Filesystem in Userspace) software system that allows the Raspbian operating system to mount and interpret exFAT drives without requiring extra privileges automatically.

Additionally, thank's to FUSE being baked into Raspbians kernel we are not required to install any special kernels to add support.

FUSE acts as a bridge between the kernel and userspace to allow developers to add support for additional filesystems without having to release a customized kernel.

Exfat-utils provides all the utilities that you need to deal with the exFAT format, including the ability to format drives to the format from your Linux devices.

To install these two essential packages, we need to enter the following two commands into the Raspberry Pi.

```
sudo apt-get install fuse-exfat
sudo apt-get install exfat-utils
```

Mounting an exFAT drive manually from terminal

1. WWhile Raspbian should automatically detect and mount your exFAT drives after installing the previous two packages, there may be a time where you have to mount it manually.

Mounting an exFAT drive can be done in the same way you would mount any other partition. Instead you will be utilizing the "-t exfat" argument to tell the mount command to recognize the file system as exFAT.

To begin, we must first create a folder where we will mount our desired drives. For our tutorial, we will be just calling this folder exfat.

Create this new exfat folder by running the following command on the Raspberry Pi.

```
sudo mkdir /media/exfat
```

2. With the folder now created we need to mount our drive. In our example below, we have already found where our device is located, in our case, this is /dev/sdb1.

To mount this device to the folder, we created in the previous step we need to utilize the following command, referencing both the SD Cards location, the folder location and that we want to utilize exFAT.

```
sudo mount -t exfat /dev/sdb1 /media/exfat
```

3. You should now be able to interact with your device at the **/media/exfat** location.

Formating a drive as exFAT

1. If you want to format your drive in the exFAT format from your Raspberry Pi, you can use that by utilizing a piece of software that came in the "**exfat-utils**" package.

This piece of software is called **mkfs.exfat**, to use it you need to specify a drive location as we have shown in the command below.

```
mkfs.exfat /dev/sdb1
```

Adding Support for NTFS on the Raspberry Pi

Project Description

In this guide, we will be showing you how you can add support for the NTFS filesystem to your Raspberry Pi.

While the Linux Kernel has some NTFS support, it is strictly read-only access. This read-only means that we have to install a separate userspace driver to be able to write to the NTFS drives.

For those who do not know NTFS (New Technology File System) is a proprietary file system developed by Microsoft to supersede both FAT (File Allocation Table) and HPFS (High-Performance File System).

It is used by the Windows operating system quite extensively, and large external hard drives often come preformatted as NTFS. For this reason, it is one of the best filesystems to get up and running on your Raspberry Pi.

Equipment

Required

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- USB Drive

Optional

- USB Keyboard
- USB Mouse
- Raspberry Pi Case
- Network Connection

Adding support for NTFS

1. Before we install the package that we need to enable our Raspberry Pi's operating system to work with NTFS filesystems we need first to update our Raspberry Pi and its package list.

To update your Raspberry Pi and to update its package list we need to run the following two commands within the terminal.

```
sudo apt-get update
sudo apt-get upgrade
```

2. With our Raspberry Pi now entirely up to date we can proceed to the next step of installing the package we require to enable support for the NTFS filesystem on our Pi.

To enable support for the NTFS filesystem, we will have to make use of a package designed for a piece of software called FUSE.

For those who do not know what FUSE is, it is simply a bridge between the kernel and the userspace. This bridge allows developers like Tuxera (The developer of NTFS-3G) to implement a new filesystem without having to have it baked in the Kernel itself.

Best of all FUSE makes it dead easy for the end user to add support, as you will see shortly we need to download a specific package to add support for the NTFS filesystem.

Now to add support for the NTFS filesystem to our Raspberry Pi all we need to do is enter the following command into the terminal to install the NTFS-3G package.eed to do is enter the following command into the terminal to install the NTFS-3G package.

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

3. With the NTFS-3g package now installed to your Raspberry Pi, it is now ready to accept NTFS drives.

Since the kernel already had read-only support for the NTFS filesystem, there is a chance that they may already be mounted to your Raspberry Pi.

The operating system will automatically try to detect any NTFS drives when they are connected and utilize the NTFS-3g FUSE driver that we just installed to interact with it.

To ensure that the operating system chooses the correct driver to read your NTFS drives that are currently plugged into your device, you may have to unplug them and plug them back in.

Doing this will force the kernel to have to re-evaluate the partitions and choose the correct filesystem driver.

Formating a drive as NTFS

1. Now if you want to format your drive in the NTFS format you can easily do this by utilising the mkfs.ntfs command. Use it alongside the location of your drive and press enter.

```
mkfs.ntfs /dev/sdb1
```

Mounting a USB Drive

Project Description

In this guide, we are going to show you how to mount a USB drive on a Raspberry Pi that is running the Raspbian operating system.

We will show you both how the Raspbian operating system automatically mounts a drive to the Raspberry Pi and how to mount USB Drives manually if you ever need to do so yourself, as some cases the system will simply fail to mount a drive, or will need remounting after being unmounted.

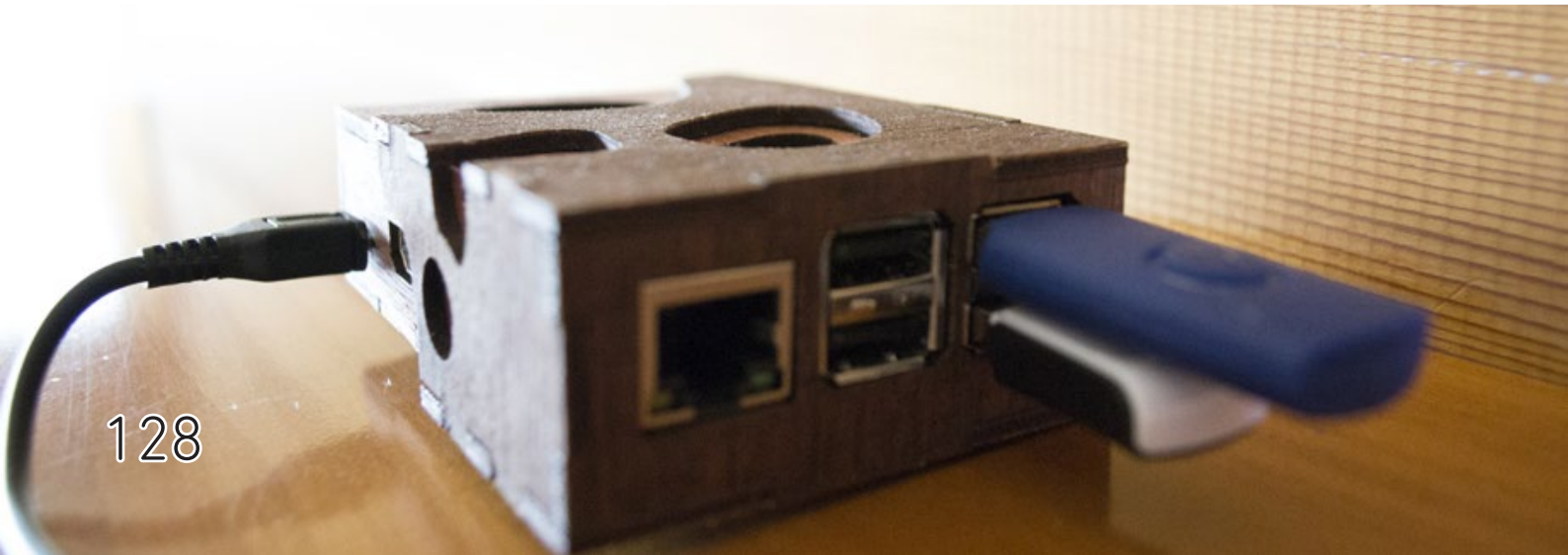
Equipment

Required

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- USB Drive

Optional

- USB Keyboard
- USB Mouse
- Raspberry Pi Case
- Network Connection



How the Raspberry Pi Mounts USB drives Automatically

If you want to check where your drive has been mounted to, you can simply use the following command:

```
sudo cat /proc/mounts
```

This command will output quite a bit of text. Any USB drives that are currently mounted to the Raspberry Pi are typically at the bottom of the text as shown in the sample output below.

```
/dev/sda1 /media/pi/CA1C-06BC vfat rw, nosuid, nodev, relatime, uid=1000, gid=1000, fm
ask=0022, dmask=0077, codepage=437, iocharset=ascii, shortname=mixed, showexec,utf8, f
lush, error=remount-ro 00
pi@raspberrypi ~ $
```

You can see that our drive that is located at **/dev/sda1** has been automatically mounted to the location **/media/pi/CA1C-06BC**.

The automatic mounting done by Raspbian will be fine for most projects and just normal use.

It will retain its mount location whenever you remove and re-insert the drive since it uses a **universally unique identifier** (UUID) of the drive for the mount folder name.

You can also find out the UUID of all your drives by making use of the following command:

```
ls -l /dev/disk/by-uuid
```

You might come across problems if you wish to allow access to the drive to a specific user that isn't the default user.

By default, the drives are mounted to a specific user, meaning other users or groups cannot gain access to that mounted drive.

In our next step, we will mount the drive using Linux's fstab file and force permissions of a given user and group so they will be able to access the drive without any issues.

Mounting a USB Drive to the Raspberry Pi manually

If you want to mount the drive with a specific user permanently, then we will need to setup the drive so that it is defined in the fstab file, and loaded under the rules set in that file.

The fstab file primarily handles how all disk partitions are loaded onto the system, among handling other different file systems.

1. Firstly install the following package if you're using an NTFS drive. Most external hard drives formatted by Windows are formatted as NTFS. Raspbian does not handle NTFS drives by default.

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

2. Now if you don't know the drives **universally unique identifier** (**UUID**) then we will need to get it.

You can get a list of drives and their UUID that are currently connected to the pi by running the following command.

```
ls -l /dev/disk/by-uuid
```

3. Now there is likely to be quite a few drives listed here. Simply look for any drive that has an address of **/sda*** where * is a number.

In my example that number happens to be 1, so the drive I am after will be **/sda1** and the **UUID** will be **CA1C-06BC**.

4. Now it is important that we create a clean directory to mount the USB to. If the directory has files in it, it will create issues with the mounting process.

To create the directory, enter the following command into the terminal:

```
sudo mkdir /media/usb1
```

5. Next, we will need to obtain both the **UID** and the **GID**.
These numbers are important as we will need them to set the correct permissions for the USB drive.

To get the gid enter the following command. (If you're using a different user than **pi** then make sure you update the username used in the following commands).

```
id -g pi
```

6. Get the **UID** of the USB drive by entering the following command.

```
id -u pi
```

7. Next, we need to make an edit to the **fstab** file. This is the file that is called on boot up to setup the drives. To edit this file simply enter the following command into the terminal:

```
sudo nano /etc/fstab
```

Mounting a USB Drive to the Raspberry Pi manually - Continued

8. Now add the following line to the bottom of this file. This line will tell the Raspberry Pi to mount the USB drive to the specified details.

Remember to update the line with all the relevant details for your USB drive. (For example, your drive will have a different UUID)

```
UUID=CA1C-06BC /media/usb1 auto nofail,uid=enter_uid_here,gid=enter_gid_here,noatime 0 0
```

Once you have entered this, you should end up with something like below.

```
# a swapfile is not a swap partition, no line here
#     use  dphys-swapfile swap[on|off]  for that
UUID=CA1C-06BC /media/usb1 auto nofail,uid=1000,gid=1000,noatime 0 0
```

9. Now since the Pi automatically mounts the drive, we will need to unmount the drive.

A simple way to do this is to use the following command (Replace **/dev/sda1** with the address relevant to your Pi). This command forces the system to disconnect that drive.

```
sudo umount /dev/sda1
```

10. Now using the following command, we can force the operating system to now re-mount the drive we just unmounted. This command will force the operating system to load the drive under the new **fstab** file.

```
sudo mount -a
```

11. If you want to make sure the drives are restored after the Pi has been shut down then simply run the following command:

```
sudo reboot
```

12. The drives should now be automatically mounted after the Raspberry Pi has finished rebooting. It will utilize the information you specified in the **fstab** file from now on.

Hopefully, you have your drive mounted on the Raspberry Pi now. If there are any issues, you can check out the troubleshooting guide on the next page.

Troubleshooting mounting your USB drive to the Raspberry Pi

These following issues will most likely arise only when you manually mount the drive.

One of the most significant problems that you will come across with mounting a drive is permissions.

If you are finding that you can't read/write files to the mounted drive, then it is likely the drive is mounted to the wrong user or group.

Another problem you may come across is the drive not being mounted on boot.

There have been some changes to Raspbian and the Raspberry Pi that might cause issues with mounting the drive-in time.

The best way to work around this issue would be to add the following lines before the **exit 0** line in the **/etc/rc.local** file.

```
sleep 20
sudo mount -a
```

These changes basically make the Raspberry Pi wait a little bit before mounting the drives, this should ensure that the drive is ready to be mounted once the sleep timer is up.

Network Basics


TP-LINK

Portforwarding
Your Raspberry Pi

Project Description

In this guide we will take you through on how to set up port forwarding for your Raspberry Pi, utilising your router. This is more of a general guide, as every router can differ greatly, we aim to just give you a basic idea on what your looking for and what sort of information you will need to set up port forwarding.

This tutorial compliements the Dynamic DNS tutorial but port forwarding is very important if you want web applications available on the internet.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

Setting Up Raspberry Pi Port Forwarding

Raspberry Pi port forwarding is a way we can allow for external access to the Pi. To do this, we will need to change some settings on the router.

All routers are different, but we will try and make this as generic as possible, but there could be a lot of differences between these instructions and your router. You can find more specific guides for your router at **https://portforward.com/**

The router I am using for this tutorial is the TP-Link AC1750 wireless dual band gigabit router.

1. On a computer that is connected to your local network, we need to first connect to the routers admin page. In any web browser, type in the following to the address bar: **192.168.1.1**

Please note that while the typical IP Address is **192.168.1.1**, it can differ between Router models.

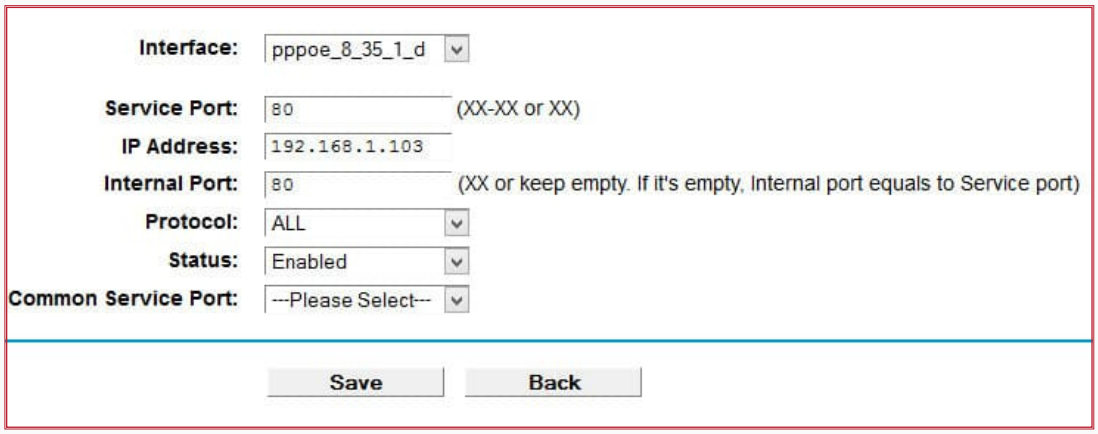
2. You will now likely be greeted with a message asking for you to log in, most routers including the TP-Link AC1750 have a default username and password of **admin.**

3. Now that you have logged in, head to **NAT->virtual servers**.

4. On this page enter the following

- **Service Port:** This is the external port
- **IP Address:** This is the IP of the Pi
- **Internal Port:** Set this to Pi's application port. (A web server runs on port 80 for example)
- **Protocol:** Set this to ALL unless specified
- **Status:** Set this to enabled

Below is an example of setting up access to a web server that is running on **port 80** with the Raspberry Pi having a **local IP address of 192.168.1.103**.



5. These settings will route traffic destined for the port specified to the port on the Raspberry Pi.

6. You should now be able to connect to the application on the Raspberry Pi outside your network.

The best way to check if you have port forwarded correctly would be to either get a friend to try connecting to your external IP address or go somewhere outside your local network and connect to the external IP Address.

Troubleshooting Raspberry Pi Port Forwarding

There are a few issues that might occur when attempting to set up your Pi for external access. Below are just a few issues you might come into when setting up Raspberry Pi port forwarding.

• Double check your router settings and confirm they are correct.

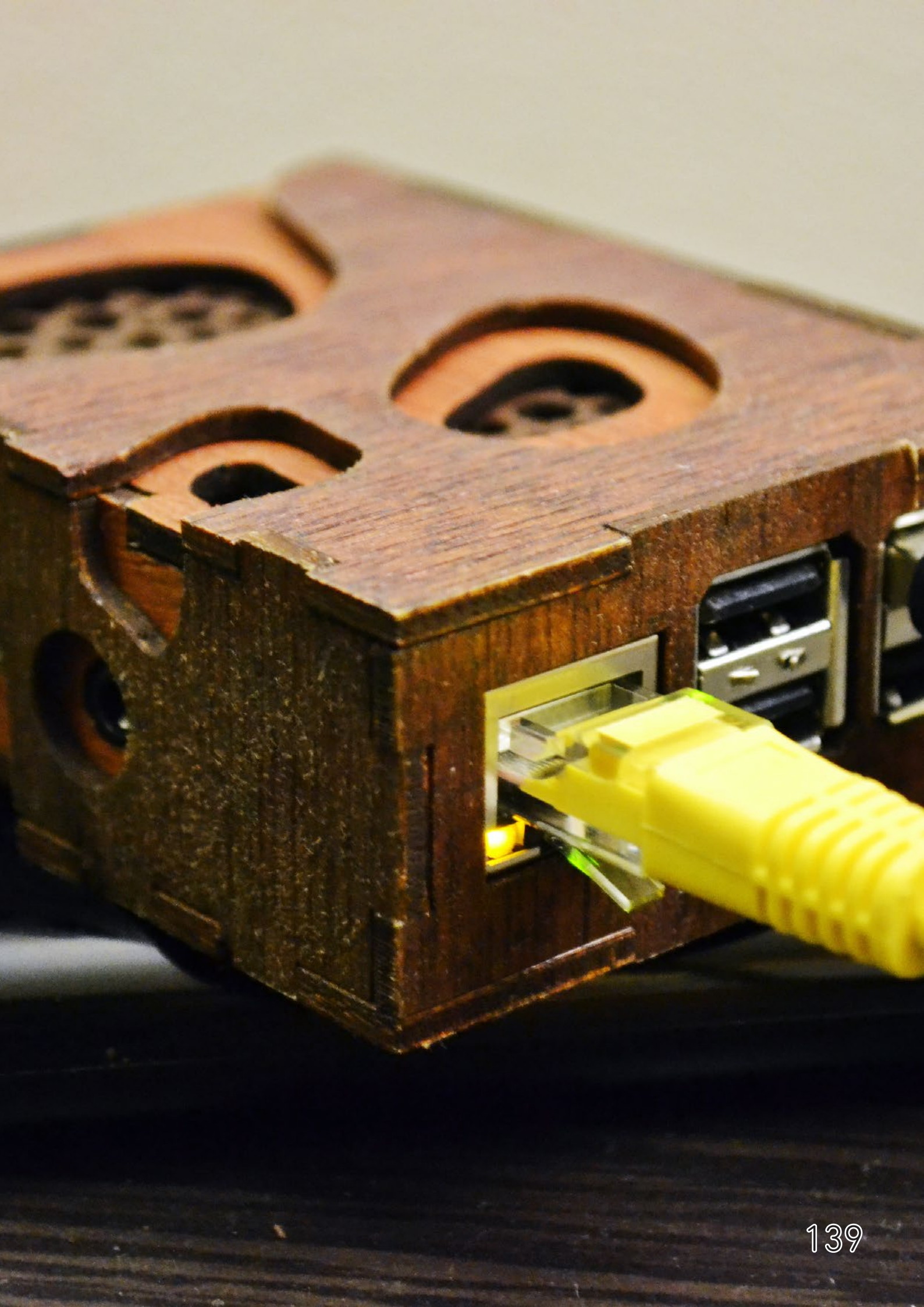
• Check that your external IP hasn't changed. ISP's will provide you with a dynamic IP rather than a static IP.

• Restarting the router might clear problems.

• Restarting the Raspberry Pi might also clear any problems.

• Make sure you have unblocked the correct port numbers for the application you are running on your Raspberry Pi. For instance, a web server such as apache requires port 80 to be unblocked. Whereas SSH requires port 22 to be unblocked for it to be accessed

• Ensure that your local IP address for your Raspberry Pi hasn't changed, with most routers it dynamically assigns the local IP address, so if the router is reset it can wreak havoc with your port forwarding settings. Luckily most routers also give you the option of setting local IP address as static for each device.



Using Dynamic DNS
With Your Raspberry Pi

Project Description

In this guide, we will take you through on setting up Raspberry Pi Dynamic DNS, something that will make it easier to connect to your Raspberry Pi from outside your local network.

Before completing this tutorial, you will have needed to have set up port forwarding for your Raspberry Pi

If your ISP supplies you with a dynamic IP (An IP that changes often), then it will probably be worth setting up your Raspberry Pi for use with Dynamic DNS. In our example, we are going to make use of the service No-IP.

A dynamic DNS means you will always be able to connect to the application on the Pi even if your external IP changes, this can be incredibly handy if you are away from your Raspberry Pi and something causes your IP address to be changed, such as restarting of the router.

Equipment

Required
- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection

Optional
- USB Keyboard
- USB Mouse
- Raspberry Pi Case

Setting Up Raspberry Pi Dynamic DNS Client

1. Before we get started, we need to setup our Dynamic DNS Client. This DNS Client will update our Dynamic DNS providers with our external IP address at a regular rate. We use the repository version of ddclient as a base install to get all the required dependencies installed.

Please note, just skip through the install, we will override these settings.

```
sudo apt-get update
sudo apt-get install ddclient
sudo apt-get install libjson-any-perl
```

2. After all of that, we will now replace that version of ddclient with a more recent one. This new version allows it to better support other services such as Cloudflare. We will get started by downloading the latest version.

```
wget http://downloads.sourceforge.net/project/ddclient/ddclient/ddclient-3.8.3.tar.bz2
tar -jxvf ddclient-3.8.3.tar.bz2
```

3. Once that has finished downloading and extracting, we will make use of the newer binary by copying it over the current one using the following command.

```
sudo cp -f ddclient-3.8.3/ddclient /usr/sbin/ddclient
```

4. Due to changes made in the most recent version of ddclient, the location of the configuration file has moved.

We will use the following commands to reposition the one that came with our first install

```
sudo mkdir /etc/ddclient
sudo mv /etc/ddclient.conf /etc/ddclient
```

5. Once the ddclient has completed installing, we will now go and directly edit the configuration file to make a few crucial changes.

Use the command below to launch an editor for the file.

```
sudo nano /etc/ddclient/ddclient.conf
```

6. We want to add a few key lines to this file. We will use these same lines throughout every different configuration. They define whether to use SSL and how to obtain the external IP address of the Raspberry Pi.

We will add more to the file later with our guides on using No-IP or Cloudflare.

```
use=web, web=checkip.dyndns.com/, web-skip='IP Address'
ssl=yes
```

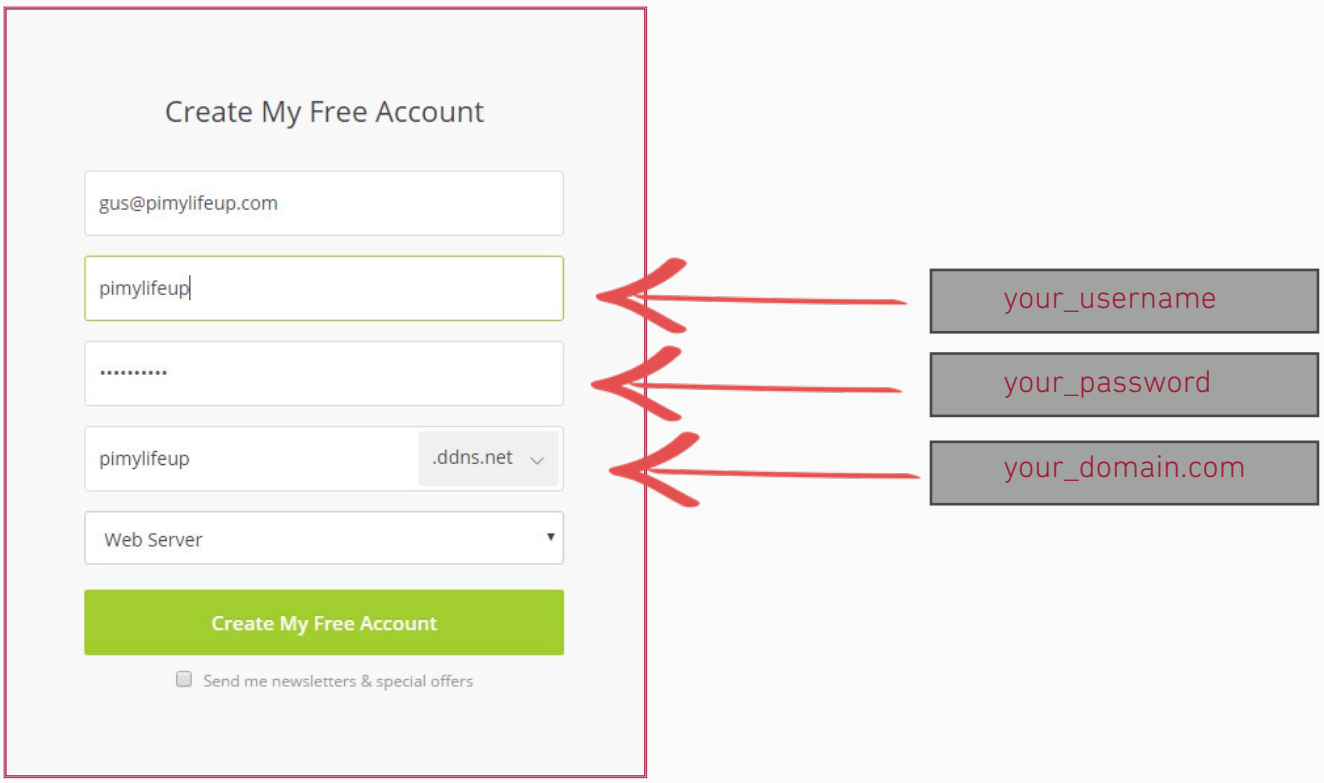
7. Now that you have done this we can move onto the next couple of pages to learn how to setup the ddclient with a Dynamic DNS provider.

The main difference between CloudFlare and No-IP is that you are required to own a domain name to make use of CloudFlare. No-IP, on the other hand, offers free subdomains, meaning it's a costless service to use.

While Cloudflare isn't a Dynamic DNS provider, we can use it the same way thanks to its API.

Configuring the Dynamic DNS client for use with No-IP

1. First, you will need to create a free account over at **https://www.noip.com/.**
Make sure you keep note of the username, password and the hostname that you pick. These will all be needed in the next step.



2. After you have finished creating your account, it's time to enter these into the ddclient configuration file. If you have closed the file from the previous part of the guide, then re-open it using the following command.

```
sudo nano /etc/ddclient/ddclient.conf
```

3. Add the following lines to the bottom of the file, making sure to replace the username, password, and hostname with the ones you used to create your No-IP account.

```
protocol=dyndns2
server=dynupdate.no-ip.com
login=your_username
password=your_password
your_domain.com
```

4. Once you have entered this into the file and swapped out the username, password, and domain name with your own, then save it and quit by using **Ctrl-X**.

5. Now, all we need to do is restart ddclient, as long as you have entered the correct information in step 3.

Use the following command to restart the dynamic DNS client:

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

Your IP should now be updated, and you will be able to use your chosen domain name to connect to your dynamic IP address.

Configuring the Dynamic DNS client for use with Cloudflare

1. Before you start using Cloudflare as a Dynamic DNS provider, you will need to purchase a domain name.

If you have a domain name already, you can skip ahead to the next step in this guide.

Otherwise, you can purchase a domain name from NameCheap by going to **http://pimylifeup.com/out/namecheap**

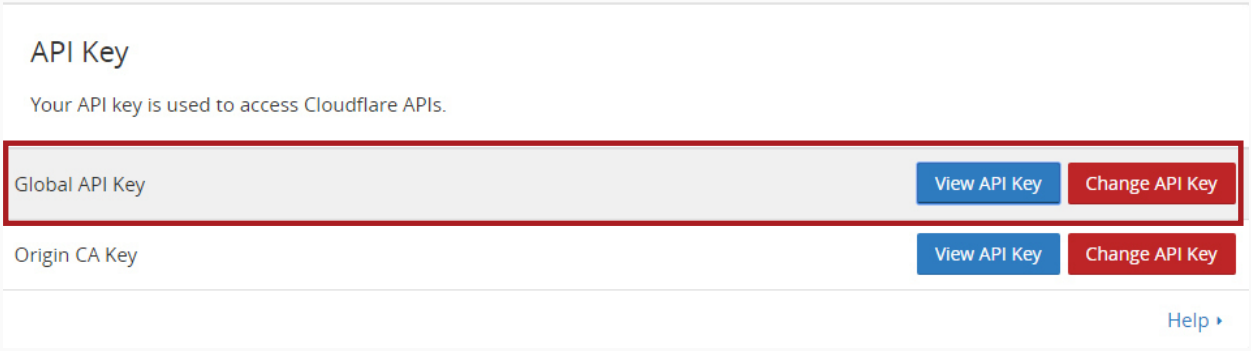
2. Now you will need to sign up to Cloudflare, don't worry it won't cost you anything. You can sign up for the service at **https://www.cloudflare.com/a/sign-up**.

Sign up and follow the steps given to setting up your domain name to point towards Cloudflares own domain name servers.

3. We will need to know the **email** you used to sign up to Cloudflare, the **domain name** you have connected to Cloudflare that you intend to use and you will also need the Cloudflare **api-key**.

To obtain your Cloudflare accounts api-key, you will need to head to your account page, **https://www.cloudflare.com/a/account/my-account**

On this page, you will need to scroll down to the API Key section and click View API key, on the Global API Key. Make sure you keep this safe.



4. After you have finished creating your account, it's time to enter these details into the ddclient configuration file.

If you have closed the file from the previous part of the guide, then re-open it using the following command.

```
sudo nano /etc/ddclient/ddclient.conf
```

5. Add the following lines to the bottom of the file, making sure to replace the username, api-key, and hostname with the ones you used to setup your Cloudflare account. The zone is the domain name itself.

```
protocol=cloudflare
server=www.cloudflare.com
login=your_email
password=your_api-key
zone=your_domain.com
anything.your_domain.com
```

Configuring the Dynamic DNS client for use with Cloudflare

6. Once you have entered this into the file and swapped out the username, password, and domain name with your own, then save it and quit by using **Ctrl-X**.

7. Now, all we need to do is restart ddclient, as long as you have entered the correct information in step 3.

Use the following command to restart the dynamic DNS client:

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

Your IP should now be updated, and you will be able to use your chosen domain name to connect to your dynamic IP address.

Running ddclient as a Daemon

Since we don't just want the IP address to update once, we still need to set up ddclient to run as a daemon so it can check for a change of IP address periodically and notify the dynamic DNS provider if necessary.

ddclient running as a daemon allows it to check for a change in the IP address periodically allowing it to update the dynamic DNS provider when necessary.

1. To make the ddclient run as a Daemon, we need to make a change to a configuration file. Run the following command to begin editing that file.

```
sudo nano /etc/default/ddclient
```

2. In that file, you will need to change **run_daemon** so that it matches the line below. This line tells it to run ddclient in daemon mode.

```
run_daemon="true"
```

3. You will also need to check two other lines in this file to make sure that they are false. Otherwise, ddclient will not run in daemon mode correctly.
Ensure the following two lines are matching, if they are set to true then change them to false.

```
run_dhclient="false"
run_ipup="false"
```

4. Press **Ctrl+X** and then **Y** to save the changes to the configuration file.

5. Now enter the following command to startup ddclient as a service.

```
sudo service ddclient start
```

6. To ensure the service has properly started, you can use the following command to check its status.

```
sudo service ddclient status
```

7. Some dynamic DNS providers require you to update your IP address somewhat frequently.
We can ensure you are not timed out by forcing ddclient to update your IP address once a week.
We achieve this by first opening the weekly crontab using the following command:

```
sudo nano /etc/cron.weekly/ddclient
```

8. To that file, add the following lines.

```
#!/bin/sh
/usr/sbin/ddclient -force
```

9. Save the changes by pressing **Ctrl-X** and then **Y** to accept the changes.

10. Lastly, we need to allow our new script to be executed. We can do this easily by doing the following.

```
sudo chmod +x /etc/cron.weekly/ddclient
```

ddclient should now be successfully running as a daemon.
Use sudo service ddclient status to make sure that it is running correctly.

Troubleshooting the Raspberry Pi Dynamic DNS Client

Troubleshooting ddclient is incredibly easy, mainly thanks to an easy to use "debug" mode.

This "debug" mode prints out every step that ddclient goes through for connecting to a dynamic DNS provider, allowing you to see at what point it is failing at.

The command for running ddclient in debug mode is the following:

```
sudo ddclient -daemon=0 -debug -verbose -noquiet
```

This command will print out a whole bunch of information, just keep an eye out for any errors that are present in the debug output.

If ddclient is successfully updating the IP address at your dynamic DNS provider, it should have a message at the end of the output that is something like the following.

```
SUCCESS:  updating backup: good: IP address set to 1.2.3.4
```

If you are running into errors, there are a few different things to keep an eye out for in the logs.

1. First is make sure that the details you put into the configuration file are correct, any mistakes here would cause ddclient to fail to update the IP Address.

2. Ensure that your computer can access checkip.dyndns.com, we set up the script to use this website to obtain your external IP address. If your internet has trouble for whatever reason, accessing this site, it will stop ddclient from running correctly.

You can try using an alternative website such as api.ipify.org.

You can achieve this by doing the following

2a. Edit the configuration file by using the following command.

```
sudo nano /etc/ddclient/ddclient.conf
```

2b. We need to modify one part in this file, that being the line that reads, **web=checkip.dyndns.com/**. In the case of **api.ipify.org**, the line would become.

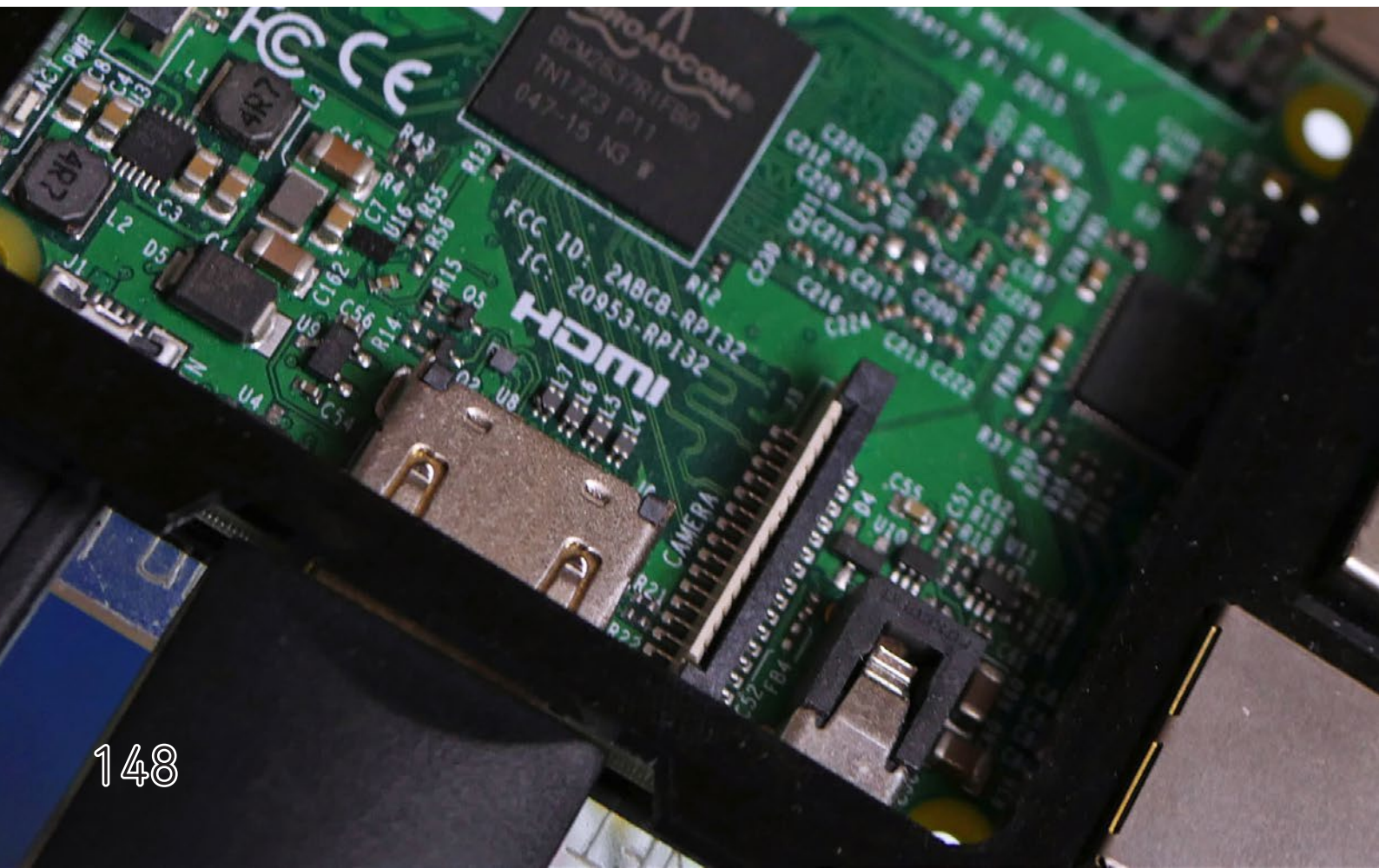
```
use=web, web=api.ipify.org/
```

You will notice we also removed the web-skip part of the configuration file as well.We removed this because **api.ipify.org** only shows the IP, meaning we don't need to skip any parts of the text before finding the proper IP address.

2c. Don't forget to save this file using **Ctrl-X** and pressing **Y,** and then type the following command into terminal, this will get ddclient to restart and load in the updated configuration.

```
sudo service ddclient restart
```

Web Browsers
For the Raspberry Pi



Installing & Using Firefox

Project Description

In this Raspberry Pi Firefox tutorial, we go through all the steps to getting the Firefox browser installed on the Pi. This tutorial is perfect if you're a fan of the browser and absolutely need it on the Pi.

This tutorial makes use of the default operating system for the Raspberry Pi, Raspbian. If you haven't got this installed, then the installation process may differ to what is detailed in this tutorial.

In this tutorial, we make use of the extended support release (ESR) of Firefox. This version has a lot of the features that you know and love with the Firefox browser. It does lack a few things that the latest version has but is still great.

It is likely that the latest version of Firefox will come to the Raspberry Pi but at the time of writing this tutorial, it doesn't currently work as intended.

Equipment

Required
Raspberry Pi
Network connection

Optional
Raspberry Pi Case (optional)

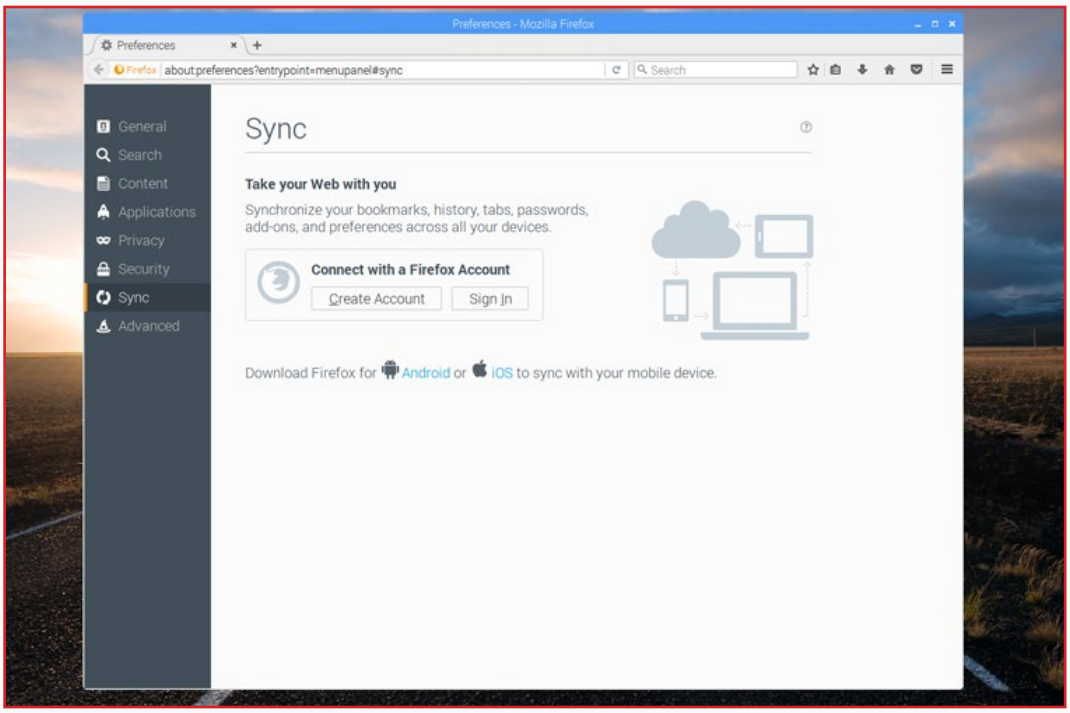
Firefox for the Raspberry Pi Features

There are many reasons why you may want to use Firefox over the other browsers for the Raspberry Pi. I will list below some of the highlights of what you will find with this browser.

Sync between devices

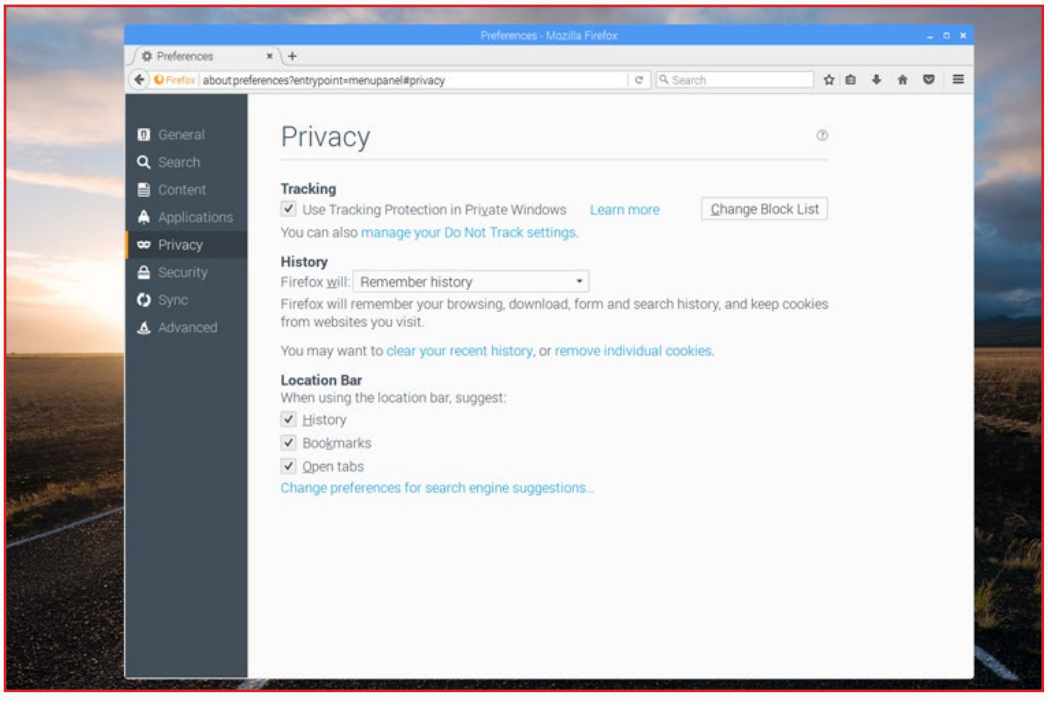
This feature is an extremely handy feature if you Firefox across all your devices.

You can have all your details synced across your phone, tablet, Raspberry Pi and any other electronic devices that make use of a browser.



Focus on Privacy

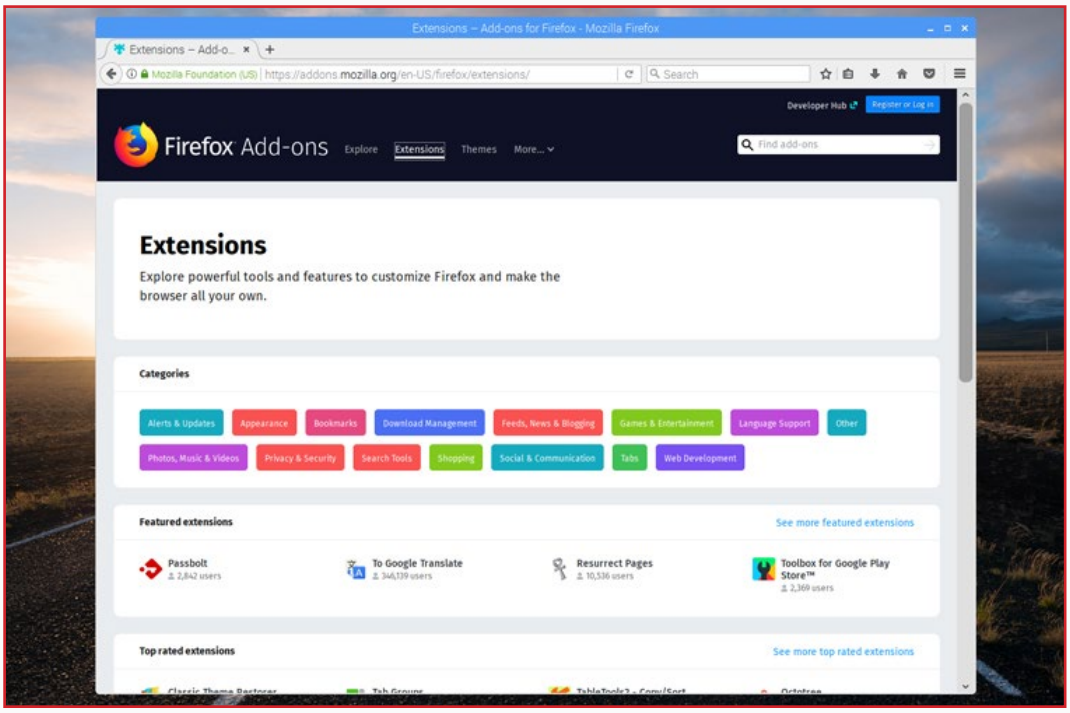
If privacy is your thing, then you will be pleased to hear Firefox has some features to try and keep your browsing as it should be, private.



Add-ons and Extensions

If you like having a good range of extensions, then Firefox is excellent.

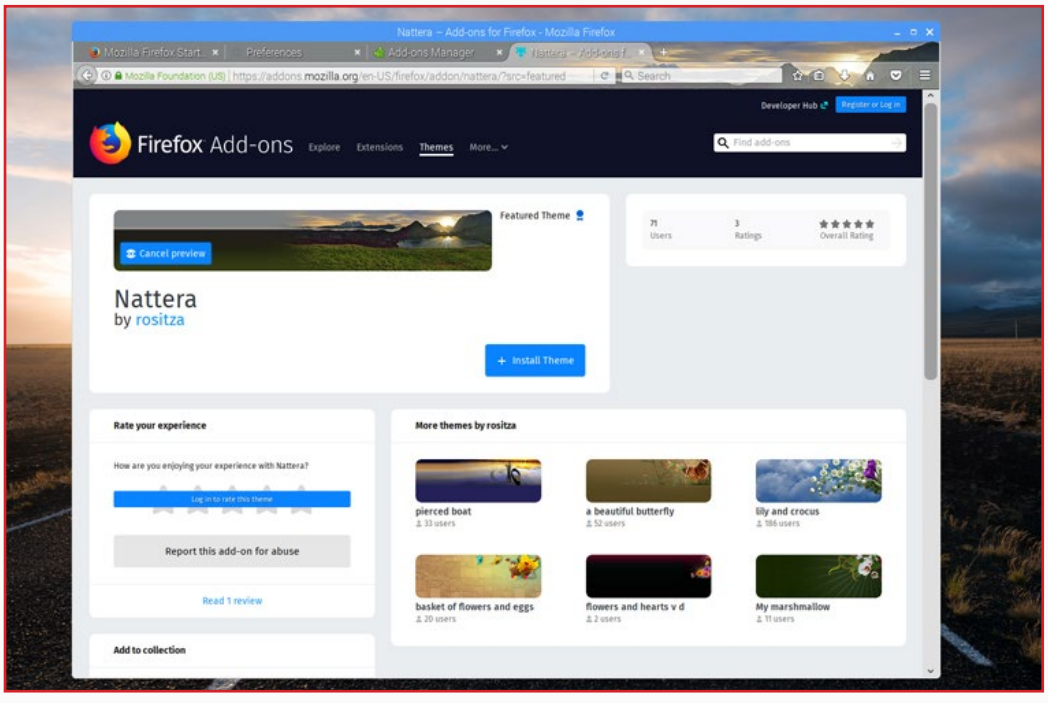
There are tons of add-ons you can install such as an adblocker, Grammarly, YouTube enhancer, Multi-Account containers and so much more.



Customizable

Much like the Vivaldi browser, Firefox too has an excellent range of options to help customize the browser best for you.

This range includes cool themes that you can download from the Firefox addon library.



How to install Firefox

The process of installing Firefox to your Raspberry Pi is an incredibly easy process and can be done in just a few short steps.

1. First off we need to make sure our installation of Raspbian on our Raspberry Pi is completely up to date.

We can ensure Raspbian is up to date by running the following commands on our Pi.

```
sudo apt-get update
sudo apt-get upgrade
```

2. Now that our installation of Raspbian is completely up to date, we can go ahead and install Firefox ESR (Extended Support Release).

Run the following command on your Raspberry Pi to install Firefox.

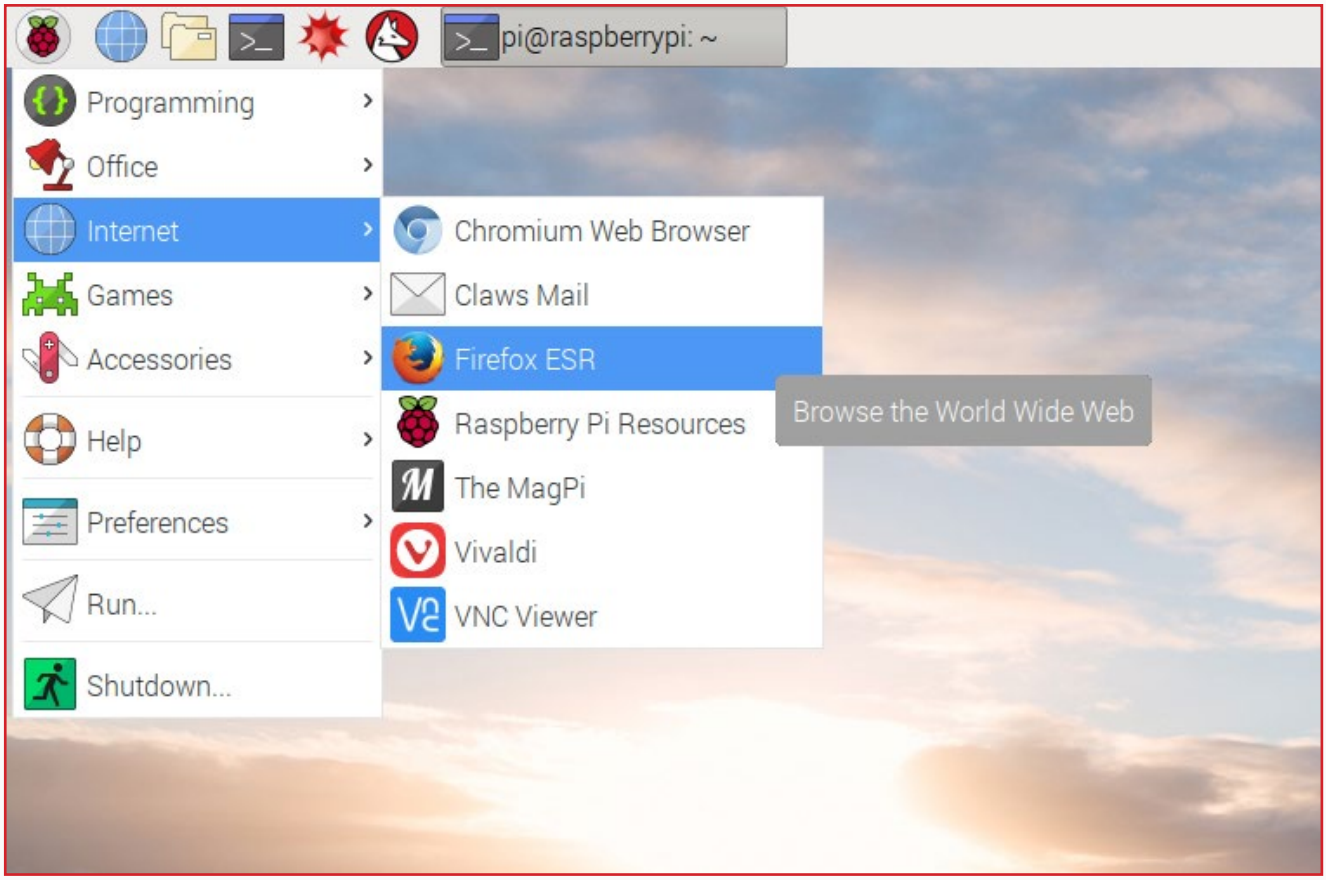
```
sudo apt-get install firefox-esr
```

3. Once the installation process has finished, go to the main menu.

Click the Raspberry Pi icon in the top left hand corner of the screen to open up the start menu.

With this menu hover over "**Internet**".

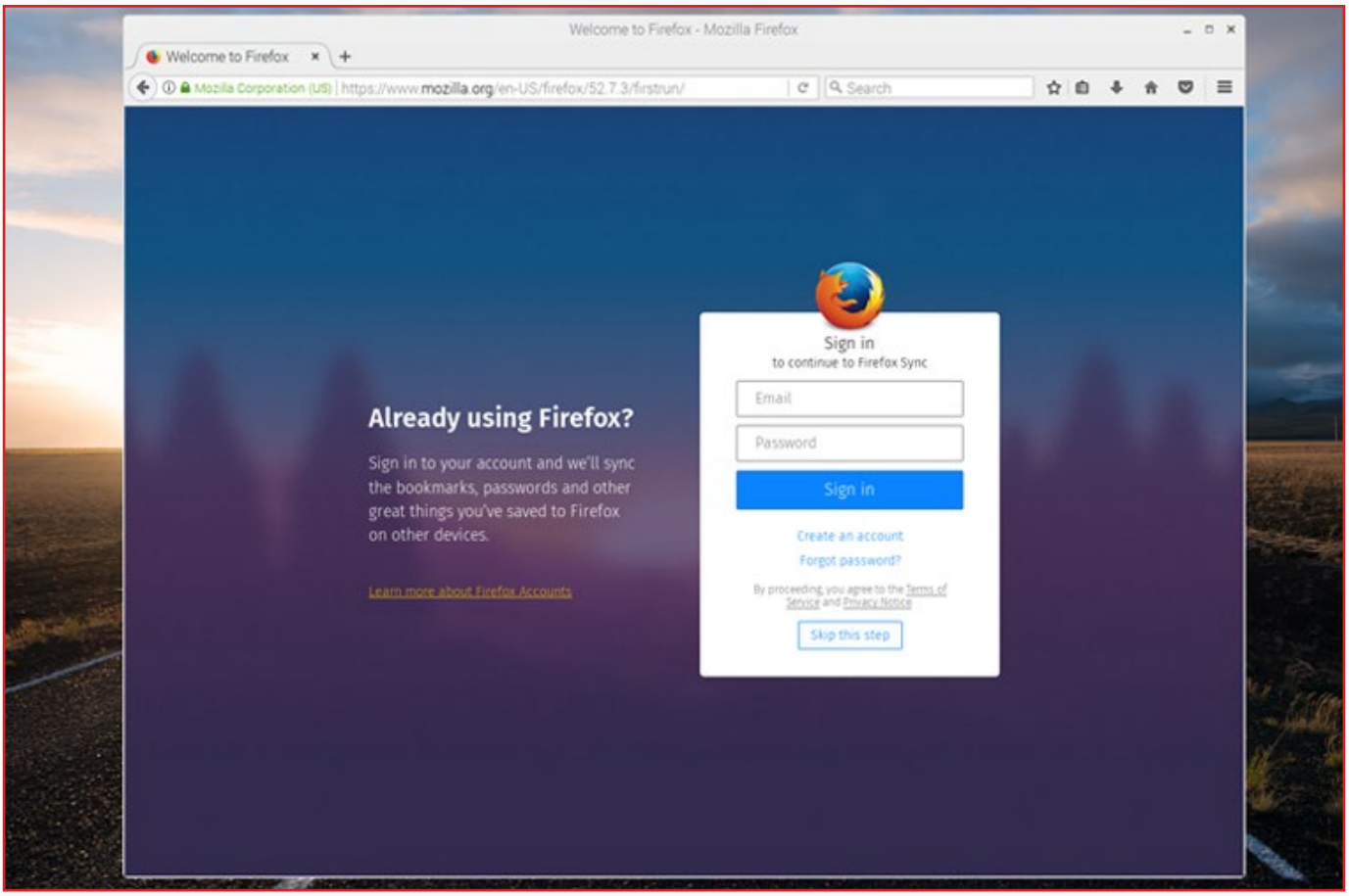
Finally click "**Firefox ESR**" to load up the Firefox Web Browser.



How to install Firefox

That's all you need to do, and the browser will work on most websites with a few exceptions.

It's pretty much perfect if you're looking for a replacement over the native Chromium browser on the Pi.



By the end of this guide we hope that you now have Firefox up and running on your device with it working flawlessly.


GPIO
Made in PRC
DISPLAY
BROADCOM®
BCM2837RIFBG
TN1723 P11
047-15 N3 W
FCC ID: 2ABCB-RPI32
IC: 20953-RPI32
HDMI
CAMERA
PWR IN
ACT PWR

Installing & Using Vivaldi

Project Description

In this guide we will be showing you how you can install Vivaldi to your Raspberry Pi while also walking you through some of its key features and why you would want to use it.

If you have never heard of Vivaldi that's because it is only relatively new, with the first release coming out in 2016.

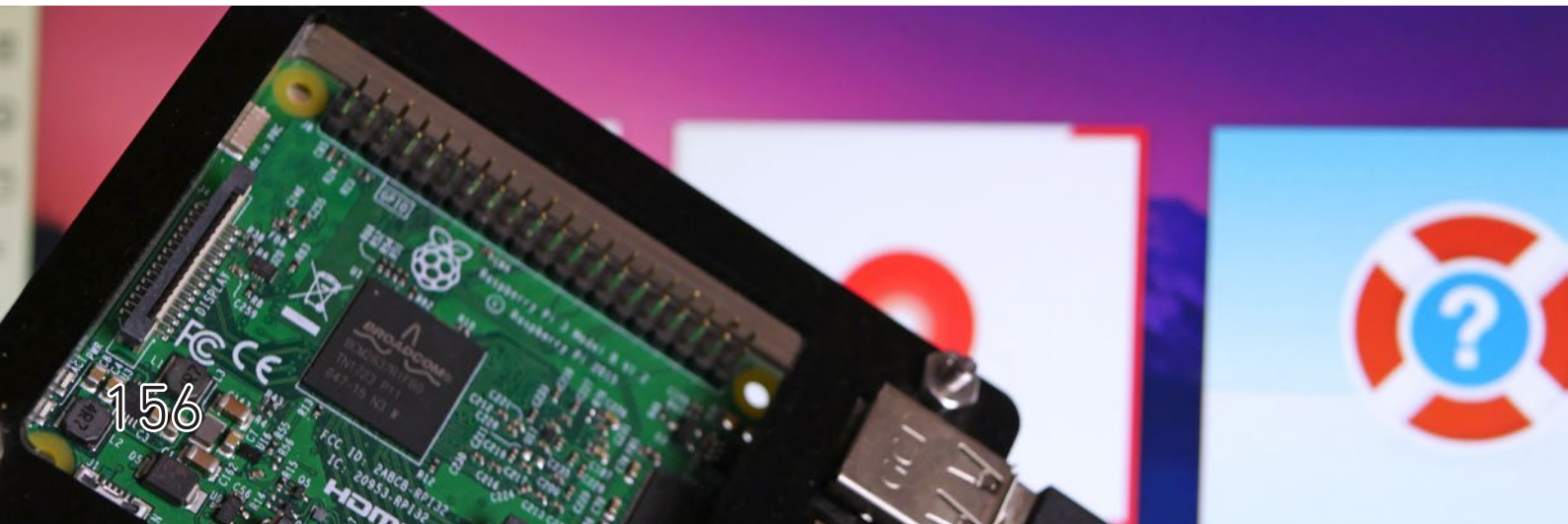
The company who develops the Vivaldi browser was founded by Jon Stephenson Von Tetzchner, the former CEO at Opera. The browser has more than a million users and is steadily growing.

If you are looking for an alternative to the default Chromium web browser that comes installed on Raspbian, then this browser is one of the better ones you can install on the Raspberry Pi that actually has proper support for the ARM processor.

Equipment

Required
Raspberry Pi
Network connection

Optional
Raspberry Pi Case (optional)



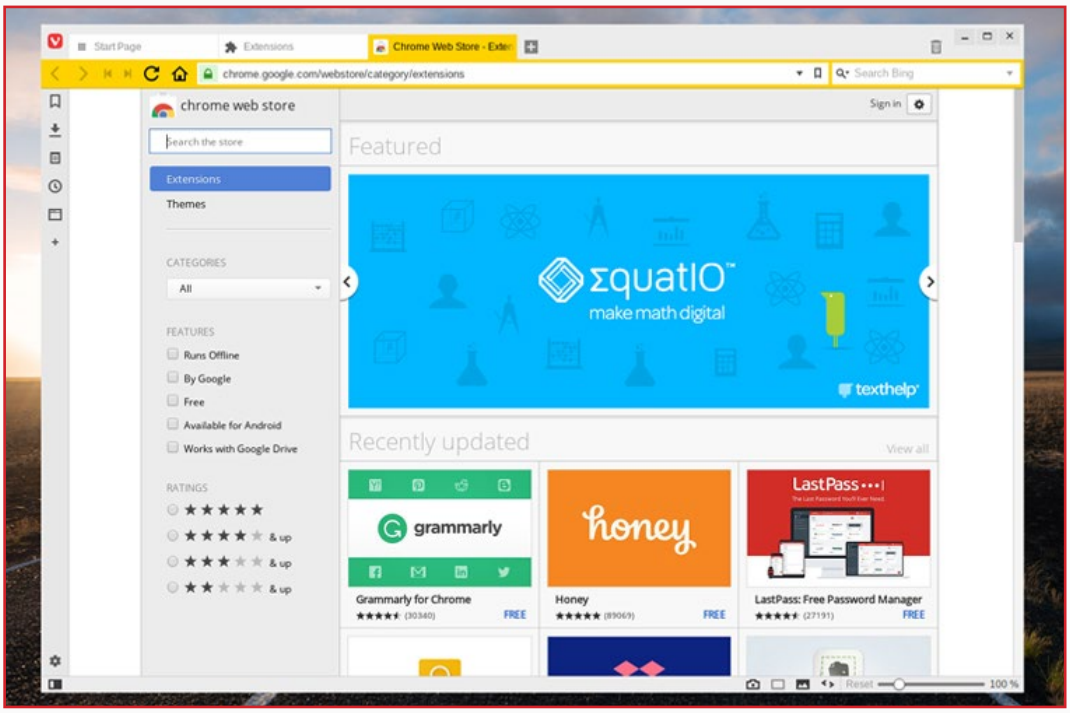
Vivaldi's Coolest Features

I will quickly highlight some of the coolest features of the Vivaldi and why you might want to download it and use it rather than the default Chromium.

Extensions

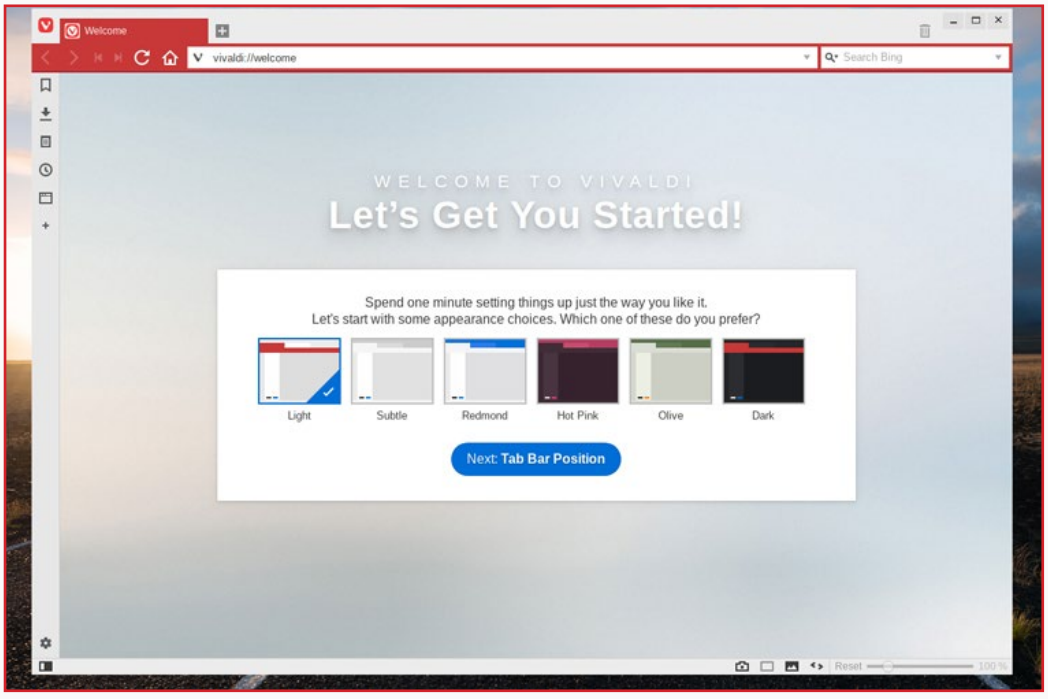
You can still get all your favorite Chrome extensions on Vivaldi with no extra configuration required. Perfect if you love having a vast range of extensions in your browser.

Remember some extensions can be quite performance heavy and may impact on your overall experience with Vivaldi on the Raspberry Pi.



Styling

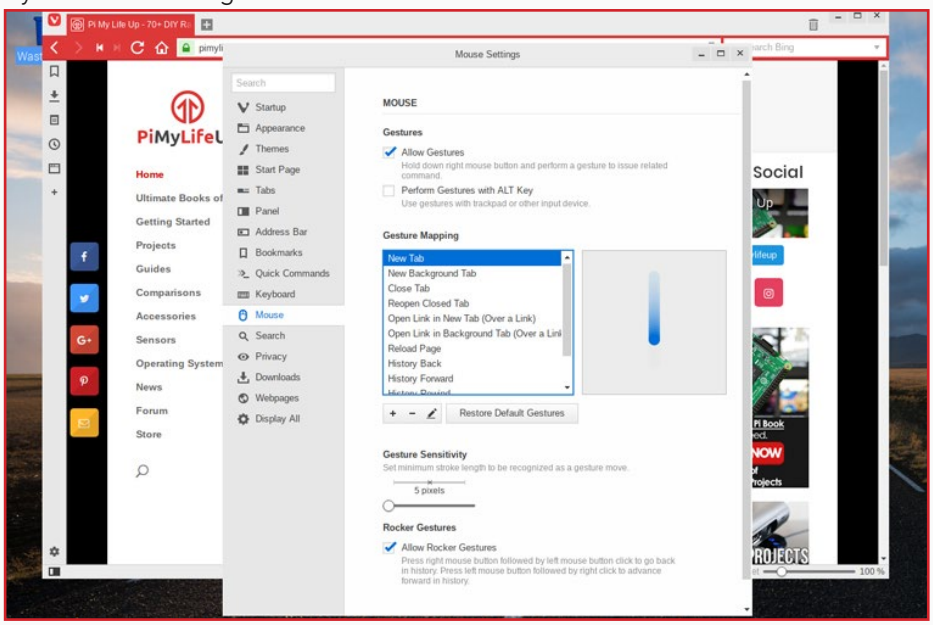
One of the biggest things I like about Vivaldi is the styles that you're able to pick from. There is a lot of different customization that you can do so you can best personalize the browser for you.



Mouse Gestures

If you're a fan of mouse gestures, then you will be please Vivaldi supports using your mouse to do basic commands.

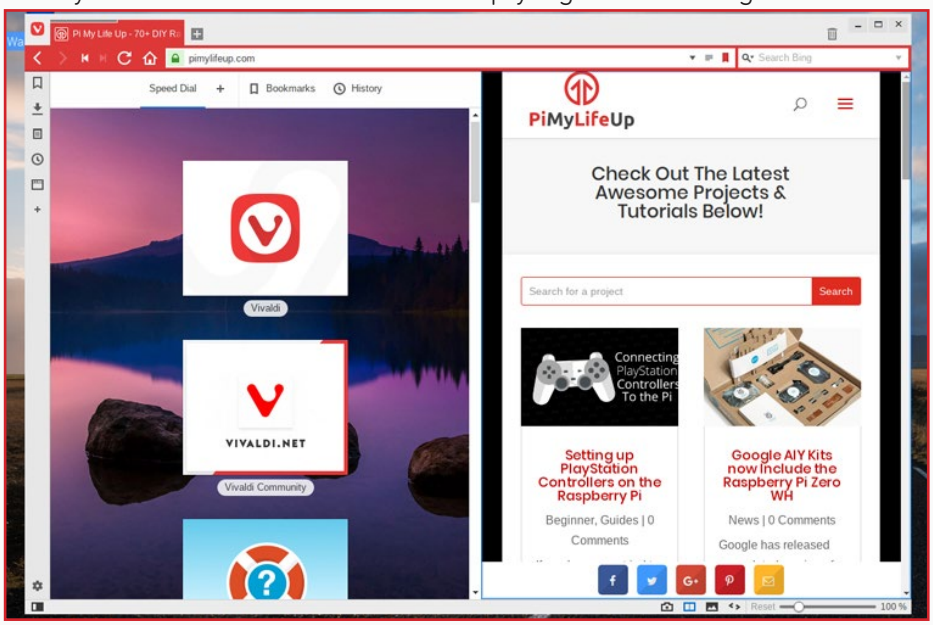
This feature is perfect if you're a bit of a power user and love the ability to shortcut a lot of typical actions you do when you're browsing.



Side by Side Browsing

Side by side browsing is extremely handy if you find the need to open two or more websites side by side. This feature is an extension of the tabbing feature where you can have multiple pages or sites under the one tab.

To move multiples sites onto a single tab simply drag down the tab down then onto the tab you want. To view the tabs side by side in the same window simply right click and go "Tile Tab Stack".



There is much more to the Vivaldi browser, and these are just some of the highlights.

There are tons more extras such as note taking, in-depth history statistics, quick commands and much more.

How to install Vivaldi on the Raspberry Pi

The most important part of this Raspberry Pi Vivaldi tutorial is the installation. It's a pretty straightforward setup that I will go into detail below.

1. To start off, we must first make sure that your Raspberry Pi is completely up to date.

We can do that by running the following commands on our Pi.

```
sudo apt-get update
sudo apt-get upgrade
```

2. Before we can install Vivaldi, we need to first download the Vivaldi Debian package file off of their website as it is not available with the default Raspbian package list.

To download this package file, just run the following command. Alternatively, you can go to Vivaldi's website and download it to your Raspberry Pi manually.

```
wget https://downloads.vivaldi.com/stable/vivaldi-stable_1.13.1008.34-1_armhf.deb
```

3. With Vivaldi's .deb file now downloaded to the Raspberry Pi, we can now install it.

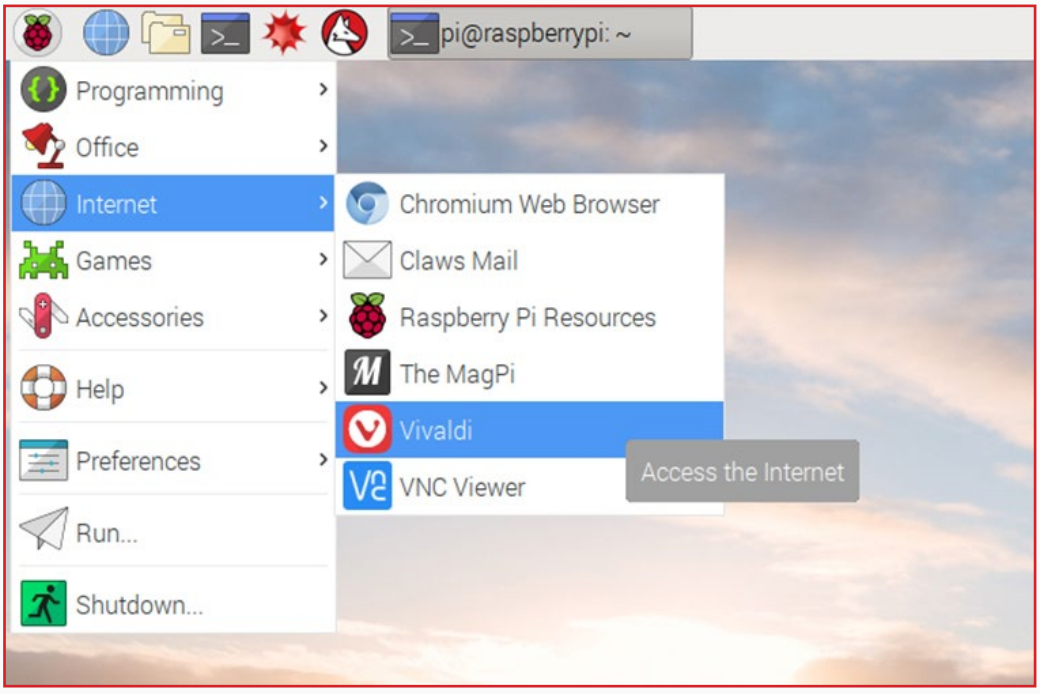
Install Vivaldi to the Raspberry Pi by just running the following command.

```
sudo apt install ./vivaldi-stable_1.13.1008.34-1_armhf.deb
```

4. Once Vivaldi's installation process has completed go to your Raspberry Pi's main screen.

On here you need to click the Raspberry Pi icon in the top left-hand corner to load up the drop-down menu, hover over "**Internet**" then find "**Vivaldi**" in the list there.

Click Vivaldi to launch it.



5. The browser should load up and you should now be able to play around with it without any huge issues.

Performance Enhancements

For a smoother performance of Vivaldi, it is recommended that you increase the swap space on the Raspberry Pi from 100mb to 2048mb. Before you increase the swap space, make sure that you have 2048mb in free space on your SD Card.

This change can easily be done by doing the following steps.

1. To increase the swap space available to the Pi we need to open up the **dphys-swapfile**.

Launch it up in the nano editor by running the following command in the terminal.

```
sudo nano /etc/dphys-swapfile
```

2. In here we need to find and edit the **CONF-SWAPSIZE** line, you can use **CTRL + W** from within the nano editor to find the line quickly.

Once you have found the line you need to change the value from 100 to 2048.

Find

```
CONF_SWAPSIZE=100
```

Replace With

```
CONF_SWAPSIZE=2048
```

3. Once done, we can save and exit by pressing **CTRL + X** then **Y** and finally **Enter**.

4. This increase in swap space should now help with the performance of the Vivaldi browser on the Raspberry Pi by allowing it to utilize more space on the SD Card to store its memory.

Please note that increasing the swap size may shorten the life of your SD Card due to the increased read and write.

Media Extensions

From my testing, most websites will work correctly by just using the basic installation method as detailed in this tutorial. If you want websites such as Spotify to work, then you will need to install some extra software.

Vivaldi has put together a very handy guide that touches on most issues that you might find with the browser running on a Pi. You can find this guide at: https://help.vivaldi.com/article/raspberry-pi/

I highly recommend checking it out if you're having any issues. Some things require a bit more tinkering to work, but hopefully, this guide has been able to get all the basics up and running.


DISPLAY
CAMERA
HDMI
FCC ID: 2ABCB-RP132
IC: 20953-RP132

Installing & Using Luakit

Project Description

In this Raspberry Pi Luakit tutorial, I take you through all the steps on how to install the Luakit browser. I assume you will be using Raspbian for this tutorial so if you're running something else, then the steps might differ slightly.

Luakit is a little different, so it may or may not take your fancy. For example, instead of the address bar being located at the top of the page it is instead located on the bottom. Several other changes make this browser stand out a bit that I go into detail on the next few pages.

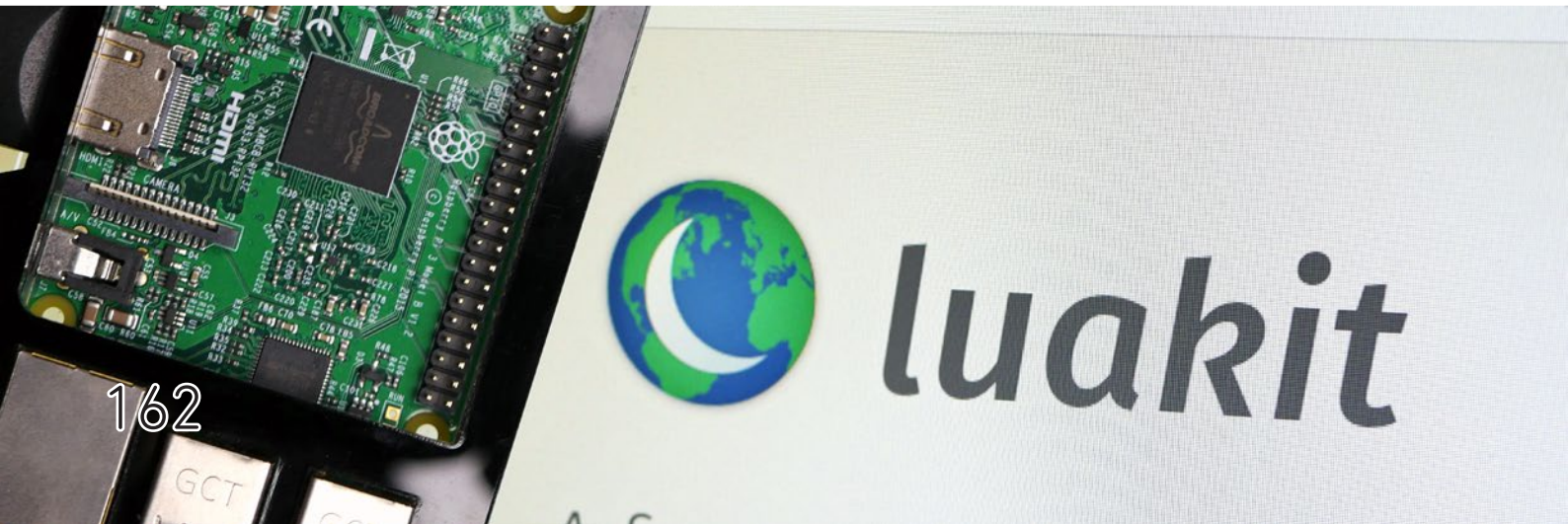
You can find out more about the browser on their GitHub page. You can also lodge issues if you find a bug with the browser.

The Luakit Wiki is also handy if you're a beginner, but at the time of writing some pages are only in Russian on the Wiki so you may need to use Google translate.

Equipment

Required
Raspberry Pi
Network connection

Optional
Raspberry Pi Case (optional)



Luakit Features

Luakit isn't nearly as feature-packed as other browsers such as Firefox or Vivaldi. However, the lack of features makes it very lightweight, fast and perfect for anyone who wants a slim browsing experience.

Extensible using Lua

You can extend the functionality of the browser with the use of the Lua programming language. It's perfect if you're already well versed in programming and want to customize the browser to your liking.

Bookmarks and Downloader Manager

You will be glad to hear that even though this browser is basic it still can bookmark web pages. You can also use the download manager to help manage any downloads you have.

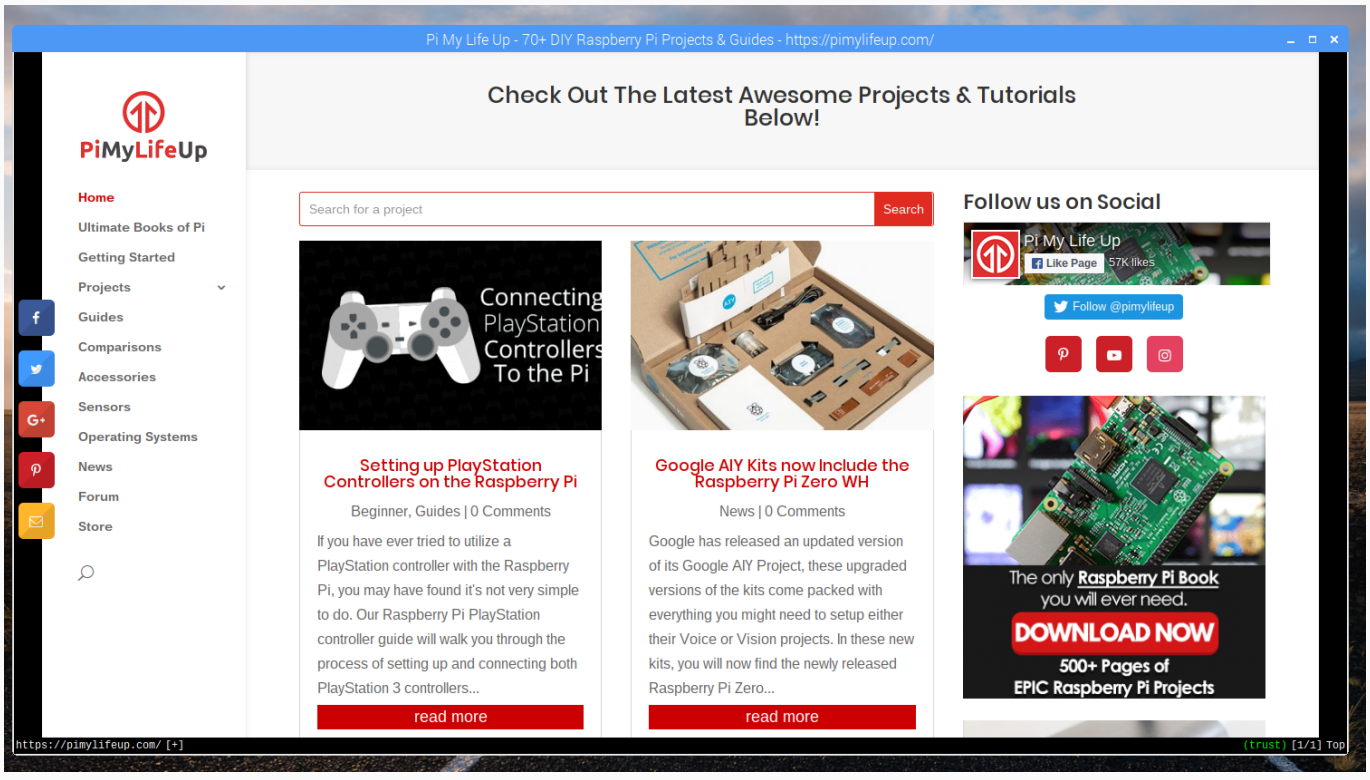
You can load up the bookmarks by either using the **:bookmarks** command in the URL bar or by using the shortcut (**g** followed by **b** or **shift+b** for a new tab).

Minimalistic UI

Most browsers you probably have used have an abundant number of buttons, menus and more to navigate.

Luakit strips right back to the bare minimum and relies on keyboard shortcuts for navigation. Perfect if you're a power user and prefer a minimal user interface.

You can find out more of the shortcuts over at the Arch Linux Luakit wiki.



There are more features that you might find attractive as these are just the top features that I found with the browser.

If you're looking for something that is super minimal but still powerful, then Luakit is undoubtedly a browser that you should consider.

How to install Luakit

The process of installing Luakit is extremely easy. If you're willing to give this lightweight browser a go then be sure to follow the few steps below.

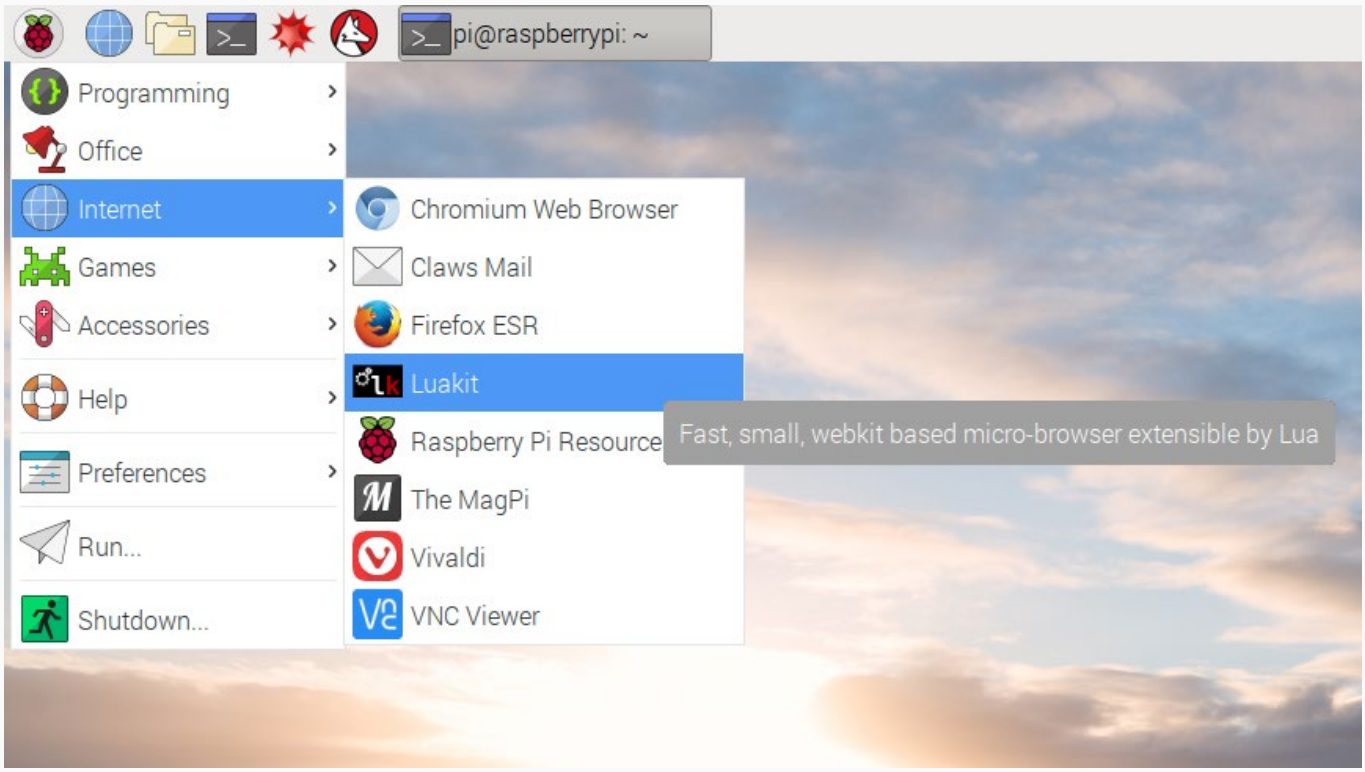
1. First, make sure your Raspberry Pi is up to date by running the following two commands.

```
sudo apt-get update
sudo apt-get upgrade
```

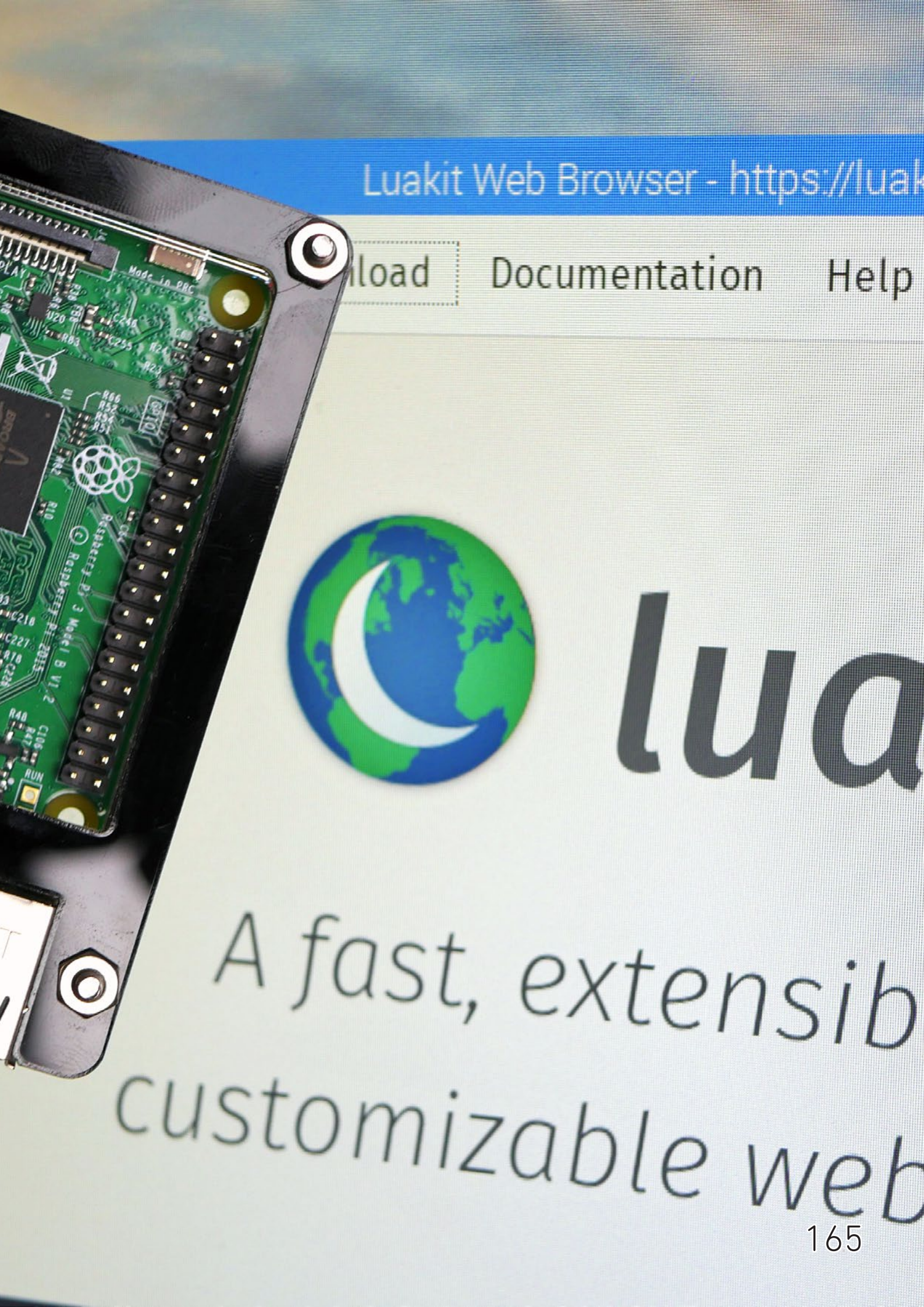
2. Next, to install Luakit, just run the following command.

```
sudo apt-get install luakit
```

3. Once done, you will find Luakit in the **Menu** -> **Internet** -> **Luakit**



I hope you have been able to get Luakit installed on your Raspberry Pi without any issues if you have some thoughts on this Raspberry Pi Luakit tutorial then be sure to let us know over on the forums.


Luakit Web Browser - https://luak
load
Documentation
Help
lua
A fast, extensib
customizable web

Midori

Project Description

In this tutorial we will be showing you through all the steps to install the Midori browser to your Raspberry Pi.

Midori is yet another lightweight web browser for the Raspberry Pi. It is very much like Luakit in that it is very bare bones and minimal.

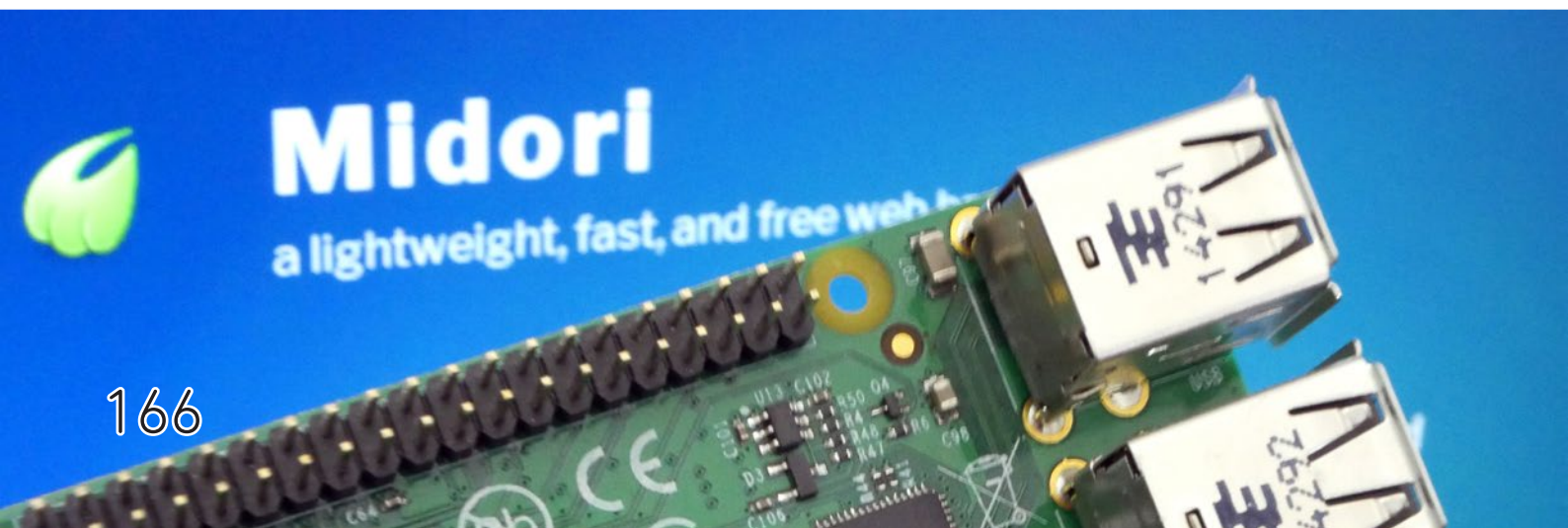
However what Midori may like in extensionability and functionality it makes up for in performance as it doesn't have to deal with as much stuff.

This makes it great for the Raspberry Pi's limited processing power and

Equipment

Required
Raspberry Pi
Network connection

Optional
Raspberry Pi Case (optional)



Midori Features

Midori might be lightweight, but it does contain a lot of the basics that you require with a modern browser. We will go through just a couple of the features, but you can find out more on their website.

Built for the Modern Web

Midori can handle most of the latest web technologies such as HTML5, CSS3 and much more. You will be able to have the full experience on most modern websites.

Some websites however may require further configuration to work on the Raspberry Pi such as Spotify due to DRM requirements.

You can find out more about the required chagnes by visiting Midori's wiki page at the following URL: wiki.xfce.org/midori/faq

Built-in Privacy Tools

Another must-have in a modern browser is the ability to clear any data that you don't want lingering around.

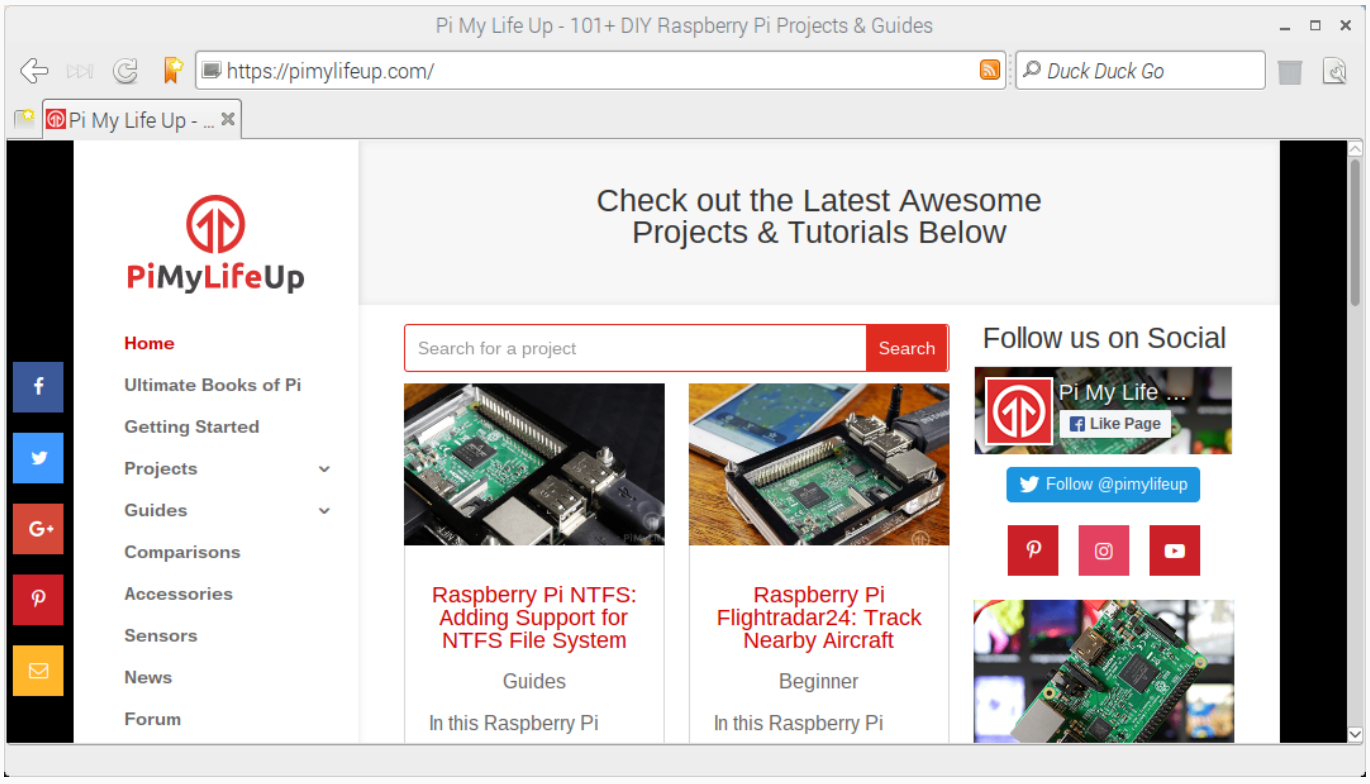
Midori also has the ability to disable scripts, third-party cookie blocking, and strip referrer details.

You can also set up automatic history clearing.

Open Source

If you love open source software, then you will be pleased to see that the Midori browser is open source.

You can obtain the source code of Midori by visiting https://wiki.xfce.org/midori/contribute, in addition you can also see how its possible to browse through their various branches of development.



Installing Midori to the Raspberry Pi

Installing Midori to the Raspberry Pi is super easy much like the rest of the browsers. This process is for Raspbian, but it should work on most other operating systems as well.

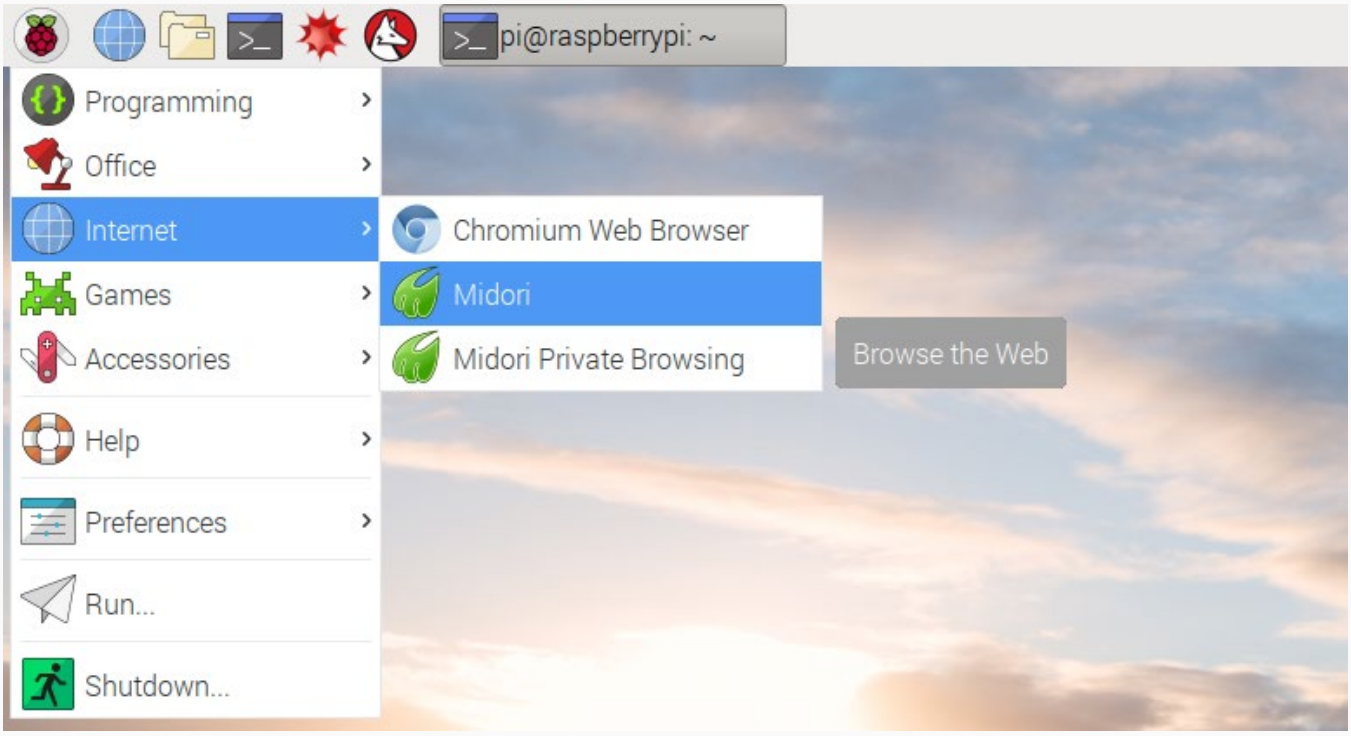
1. First, ensure that Raspbian and its packages are up to date by running the following commands.

```
sudo apt-get update
sudo apt-get upgrade
```

2. Next, let's install Midori to our Raspberry Pi, to do this you need to simply run the following command as Midori is easily available through Raspbian package repositories.

```
sudo apt-get install midori
```

3. Once installed, you will be able to find the brwoser by going to **Menu** -> **Internet** -> **Midori**, as shown in our screenshot below.




Midori
a lightweight
free web bro

Epiphany Browser

Project Description

In this tutorial we will be showing you how to install the very simple web browser known as Epiphany to your Raspberry Pi.

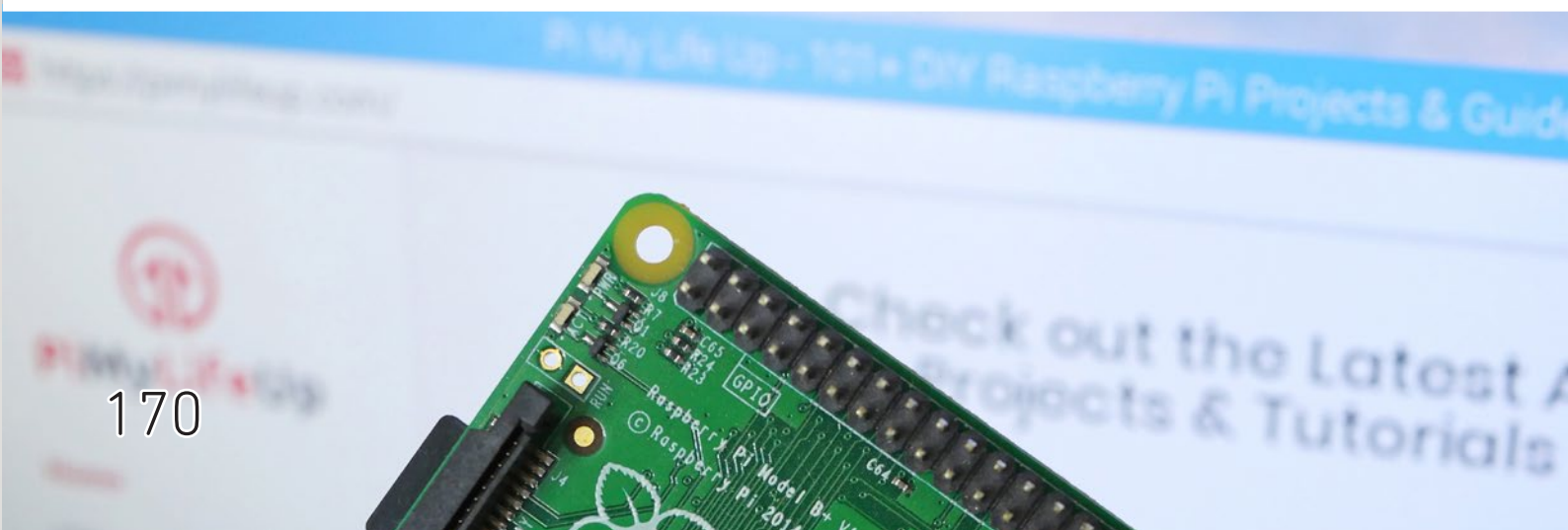
Many people will know the Epiphany Web Browser from its more common name as "Gnome Web" as it is usually intergrated with the Gnome desktop interface

Epiphany is a very straightforward and bareboned web browser, while not as light as browsers like Luakit and Midori it still offers a much more slimmed down experience to the more heavyweight and demaning Chromium.

Equipment

Required
Raspberry Pi
Network connection

Optional
Raspberry Pi Case (optional)



What's the Epiphany Browser

Epiphany was previously the primary web browser for the Raspberry Pi until it was replaced with the Chromium browser.

You will find that Chromium is much more feature packed and easier to extend. However, if you don't like the Chromium browser, then this is a strong contender as an alternative.

Unfortunately, the Epiphany package for the Raspberry Pi isn't at feature parity with the official version. You will find a lot of the newer features are missing from the Raspberry Pi version of the browser.

Installing Epiphany Browser on the Raspberry Pi

Installing the Epiphany browser to the Raspberry Pi is incredibly easy. It involves installing one package but before we do that we should update our package list and packages.

1. First, ensure that Raspbian and its packages are up to date by running the following commands.

```
sudo apt-get update
sudo apt-get upgrade
```

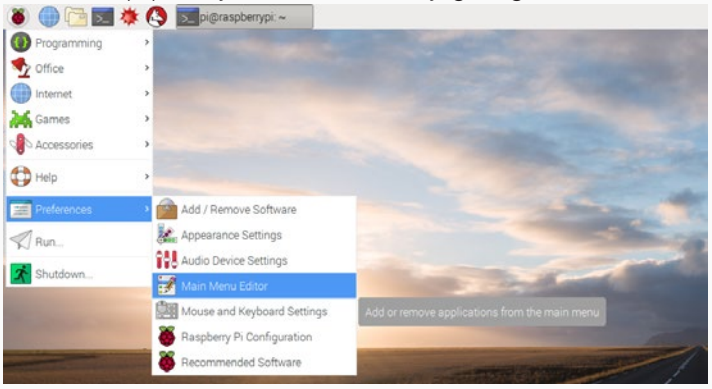
2. First, ensure that Raspbian and its packages are up to date by running the following commands.

```
sudo apt-get install epiphany-browser
```

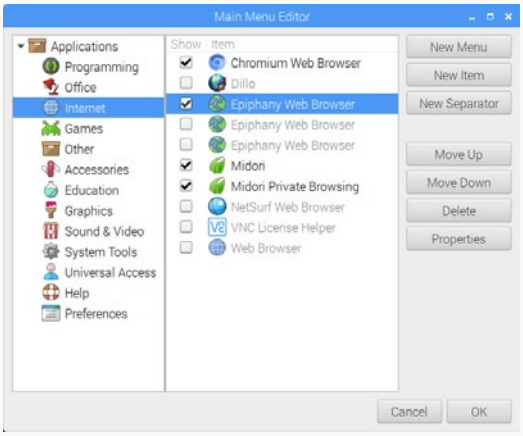
3. Now to launch epiphany we can utilize the following command within a terminal session.

```
epiphany
```

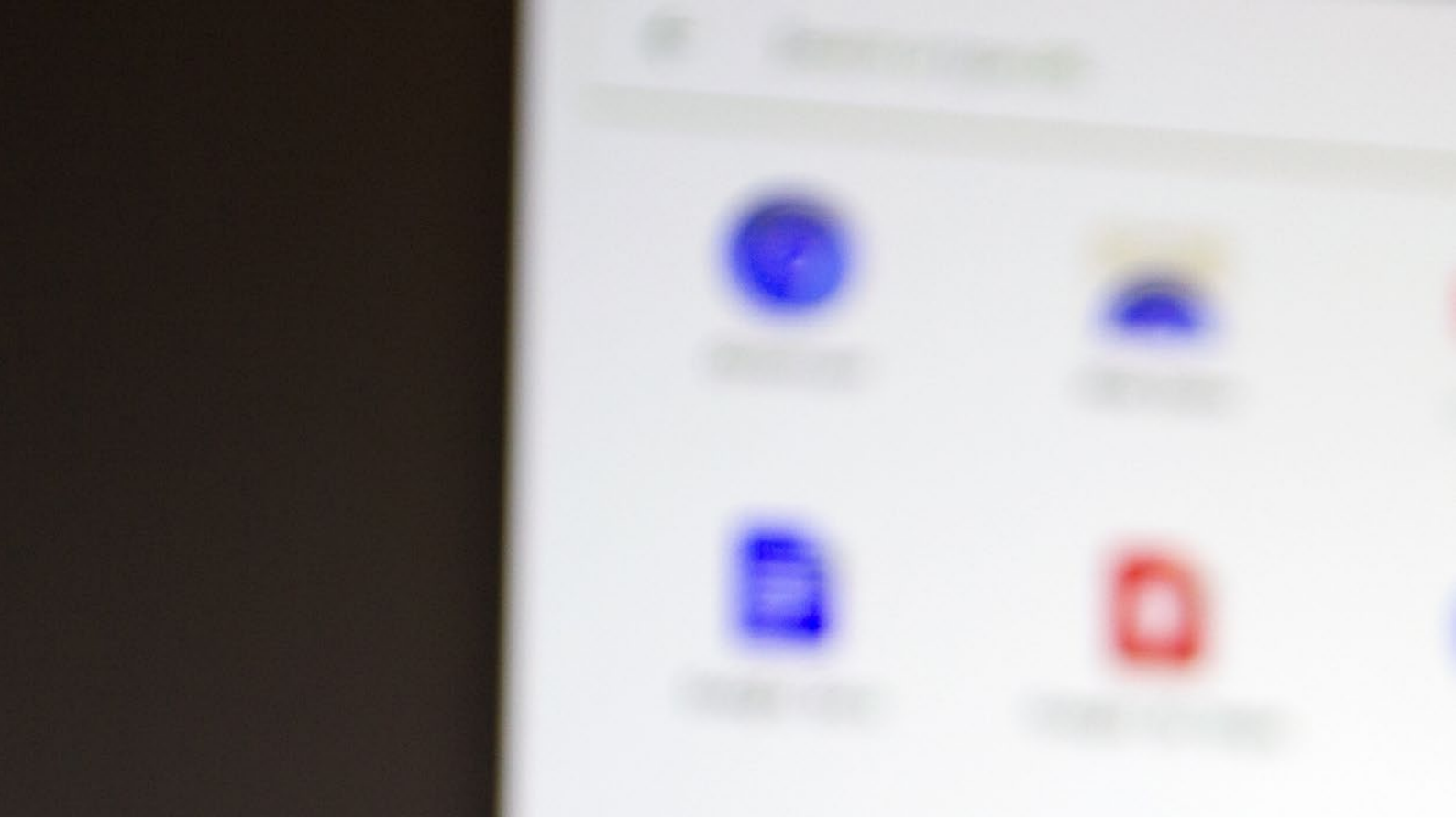
4. Alternatively you can also add Epiphany to the menu by going to **Main Menu** -> **Main Menu Editor**.



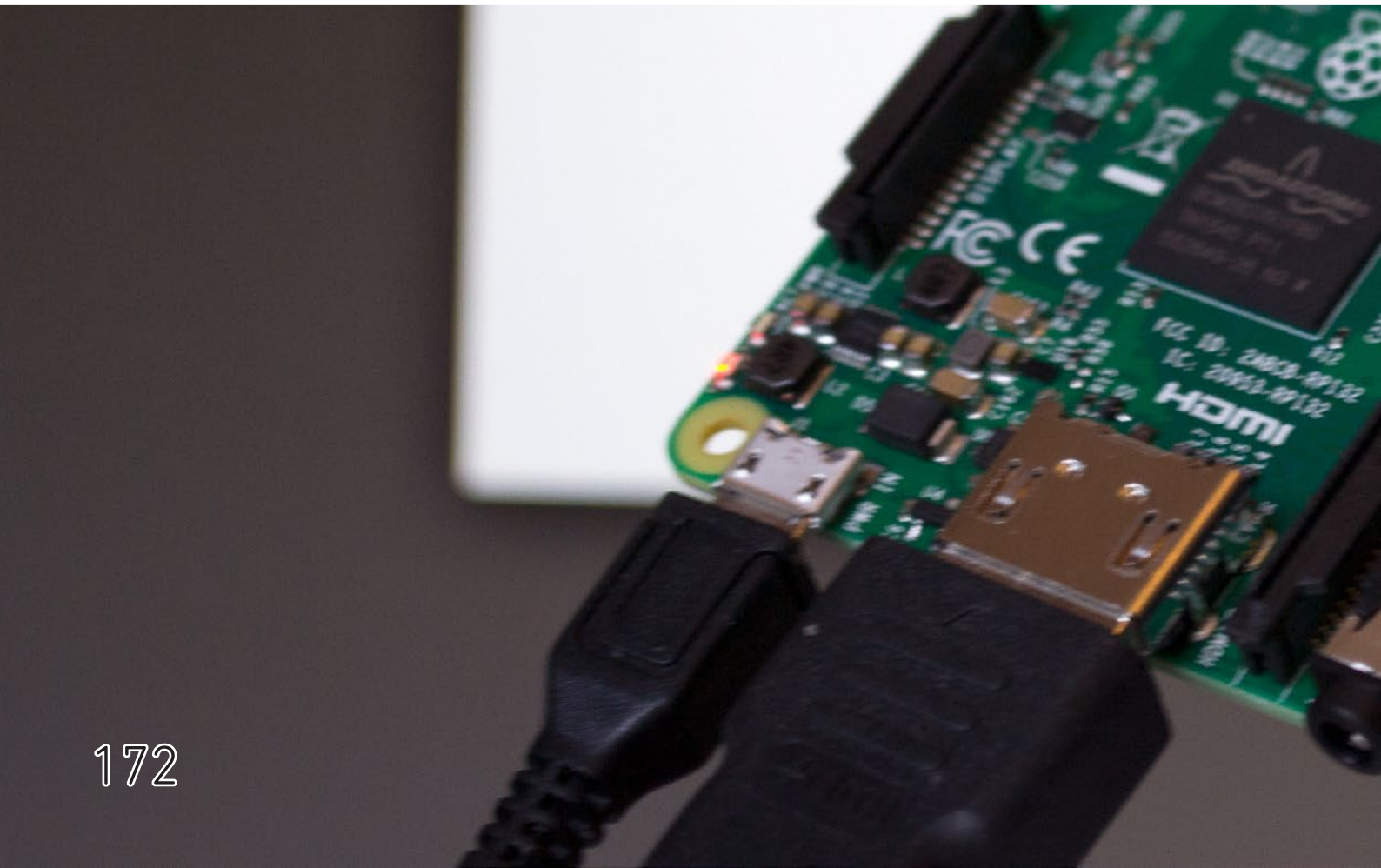
5. Within this application, click "**Internet**" in the left hand column and tick any one of the **Epiphany Web Browser** entries.

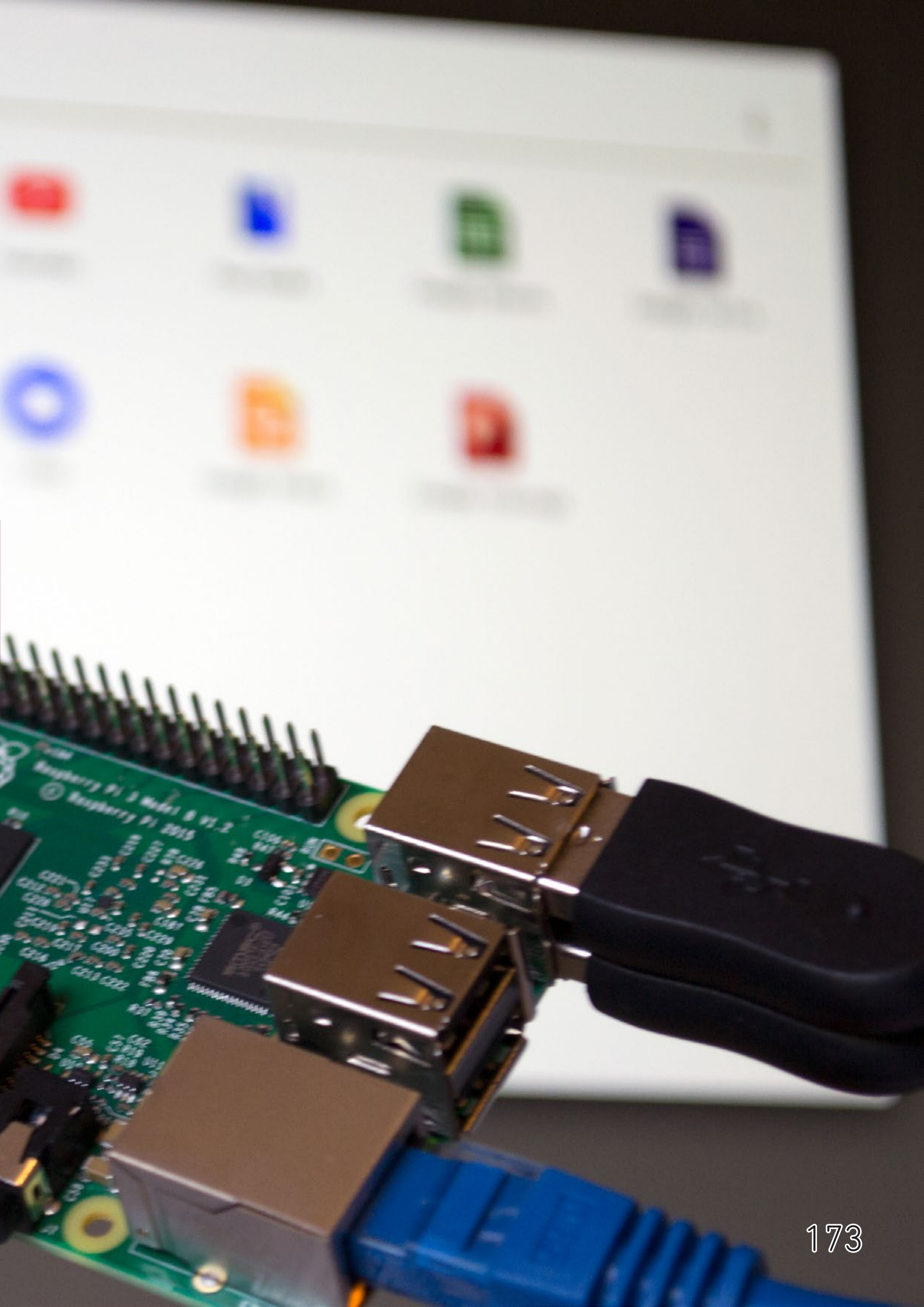


6. You should now be able to find it within the menu under the "Internet" category. It will likely be referenced under the name "**Web**".



Operating Systems





How to install Flint OS

Project Description

In this tutorial, we go through the steps to installing Raspberry Pi Flint OS. I also do a brief overview of what exactly the operating system (OS) is and any setup that you will need to do on first boot.

Flint OS is perfect if you only need a basic operating system that can run a popular range of applications such as Google docs or even the web versions of Microsoft Office apps. You're able to access the chrome web store, so you have a wide range of extensions, applications, themes and much more to install.

Not all of the Chromium OS apps will run on the Raspberry Pi as some of them are quite intensive and need a fair bit of resources. For example, "Cut the Rope" will run the first mission and that's as far as you will get.

It's important to note that if stability is hugely important to you, then I do not recommend installing this operating system until it has been worked on a little bit more. However, if you love experimenting and trying out new things, then you should try this out.

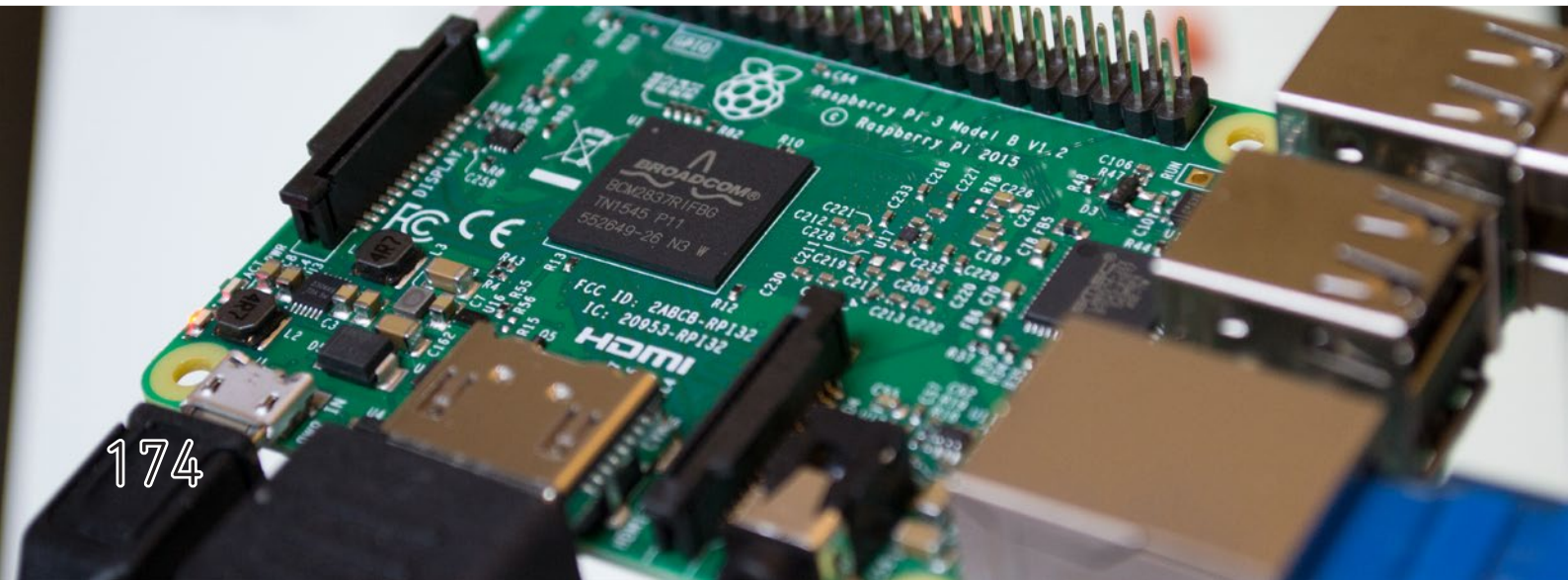
Equipment

Required

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- USB Keyboard
- USB Mouse

Optional

- Raspberry Pi Case



Step 1 - Formatting your SD Card

First off you should do when installing a new image to an SD card is format it correctly.

The following steps will show you the best way to format your SD Card for the Raspberry Pi:

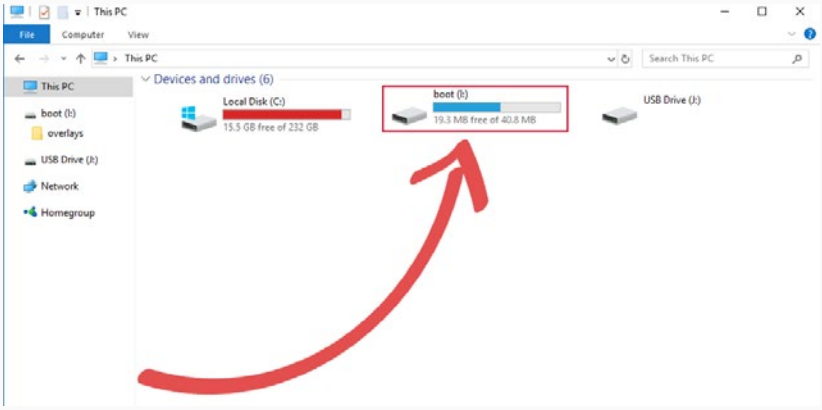
1. To begin you need a tool to format your SD Card. You can obtain one of the most robust ones from here: **https://www.sdcard.org/downloads/formatter_4/**

Once you are on the website will find a download link for the SD Formatter.

2. Download and install the SD Card Formatter from the SD Card association.

3. With the SD Formatter installed to your computer, insert your SD card and check the drive letter that is allocated to it, e.g., **G:/**.

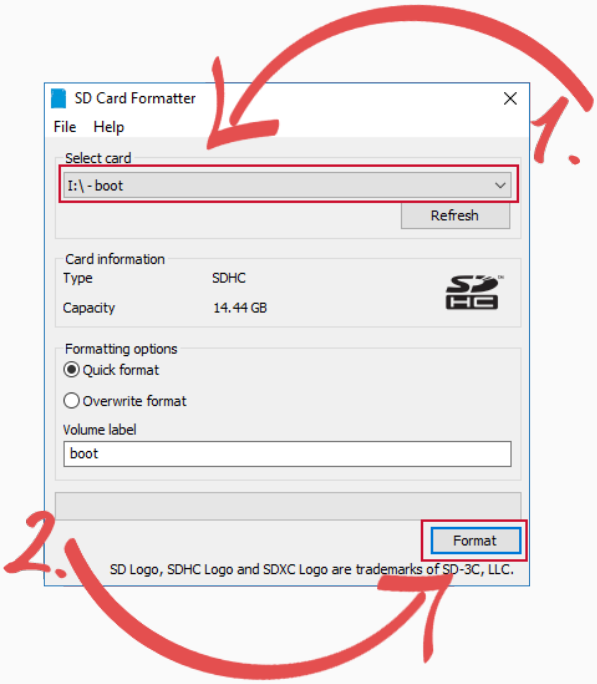
Make sure that you have the correct drive letter otherwise you could accidentally format the wrong device.



4. With your SD Card now plugged into your computer, launch the SD Formatter software that we installed earlier.

First set the drive letter to the correct one **(e.g., G:/)**.

Once you have everything set correctly, press the "**FORMAT**" button.



Step 2 - Installing Flint OS

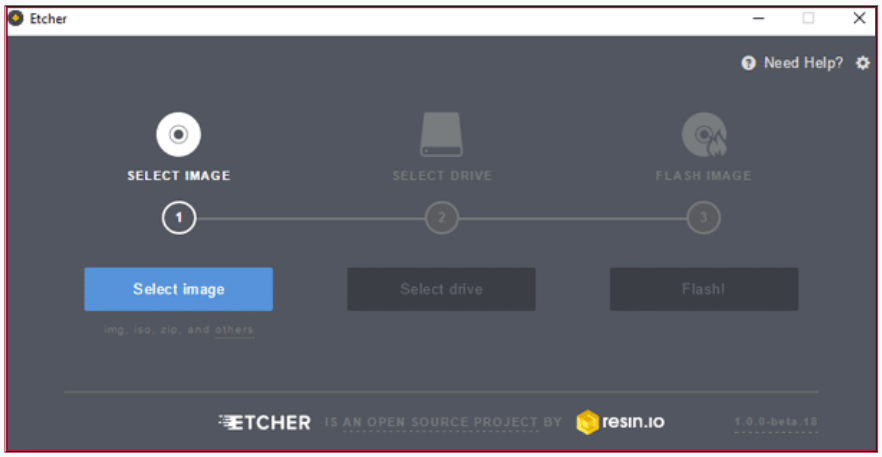
1. First head over to the Flint OS website at https://flintos.io/download/ and download the latest zip file.

Once downloaded, unzip the file so, you are left with a **.img** file.

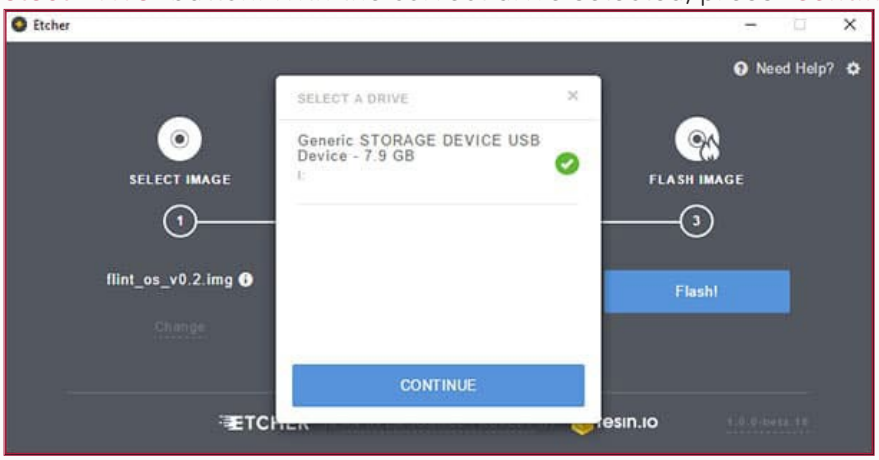
2. Next, you will need some tools to be able to write the image to the SD card.

For this we will use a tool called Etcher, you can download at https://etcher.io/.

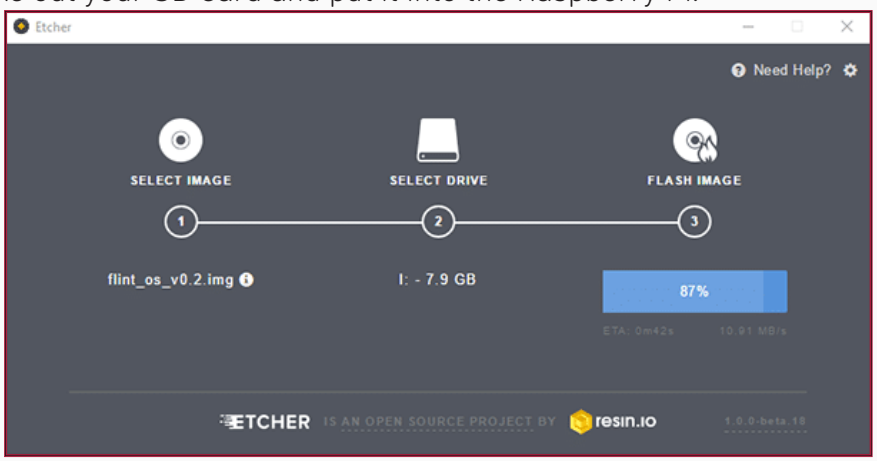
3. Open Etcher and press the "**Select Image**" button. Browse to the image you downloaded in **Step 1** and select it.



4. Next click the "**Select Drive**" button. With the correct drive selected, press "**Continue**".



5. Finally, click the "**Flash!**" button to begin the flashing process. This can take some time. Once completed, take out your SD Card and put it into the Raspberry Pi.

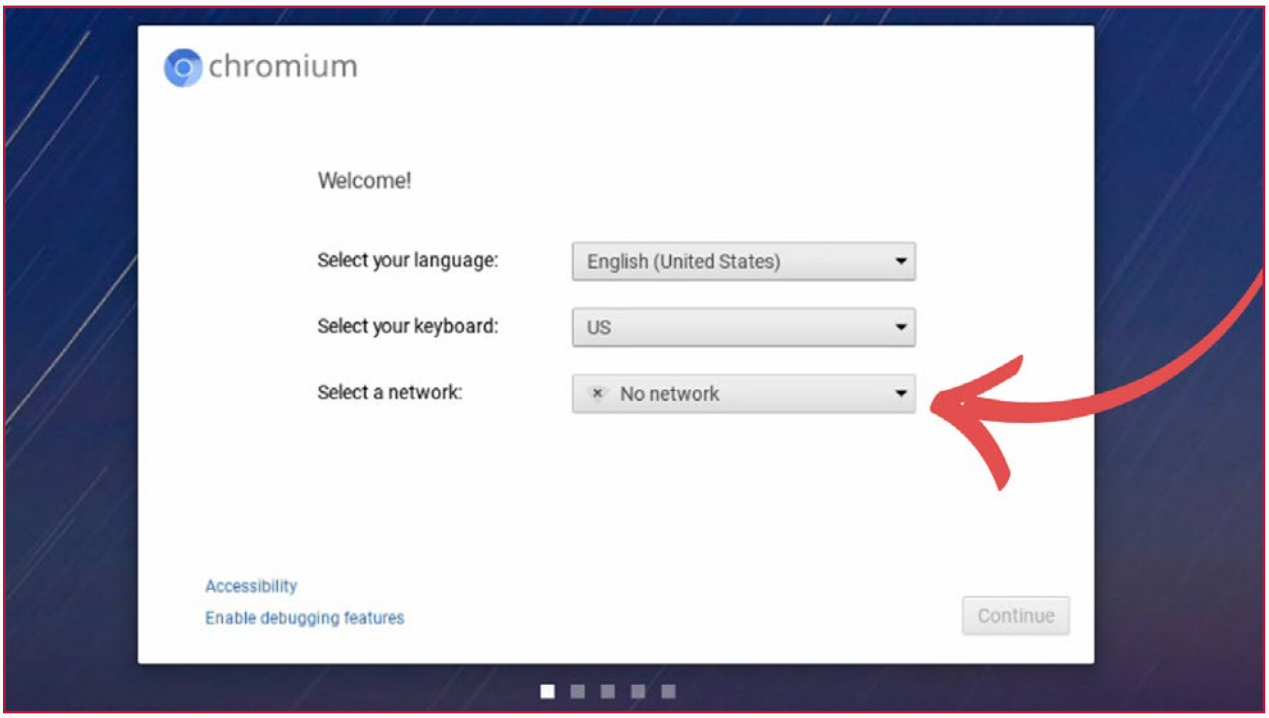


First Boot of Flint OS

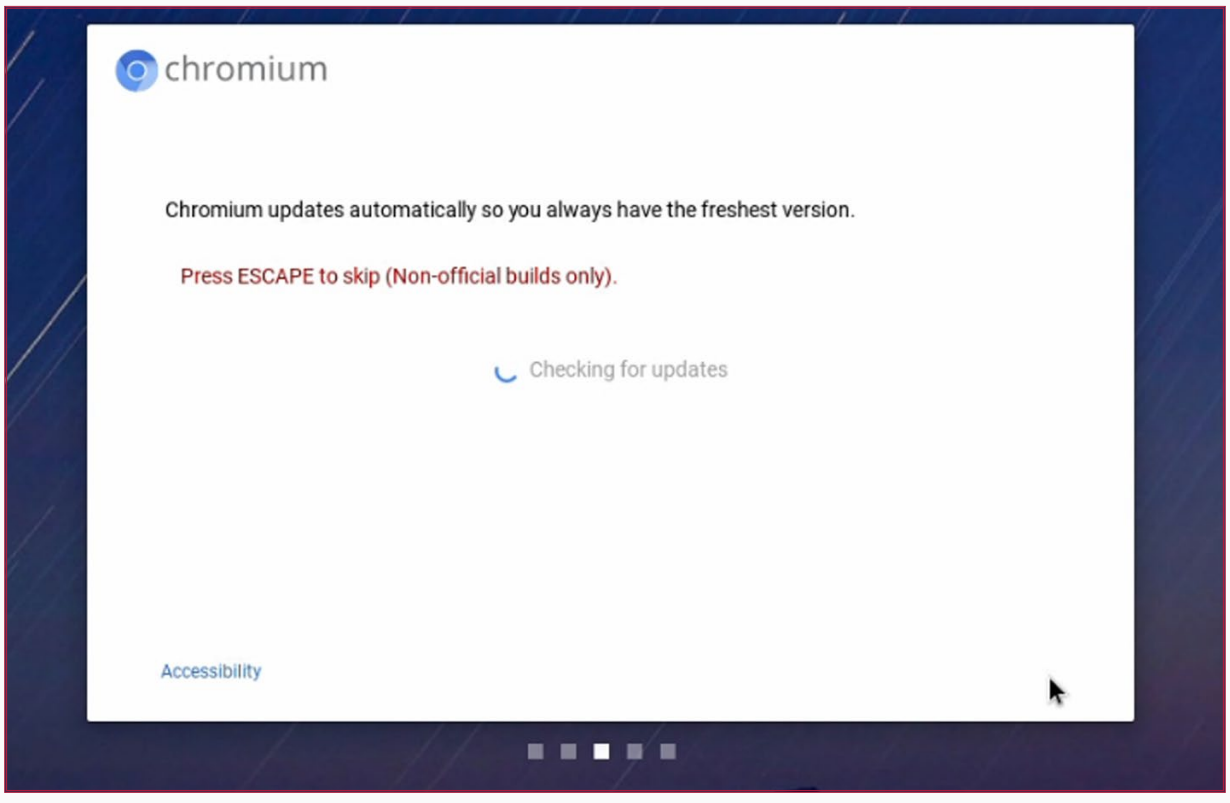
1. First, you will need to connect to a network, Chromium makes this a simple process.

Simply just click the "**Select a network**" dropdown box to pick the network you want to connect to.

If you have a Raspberry Pi 3, then the onboard Wi-Fi is supported in the latest version. Dongles and Ethernet should also work.



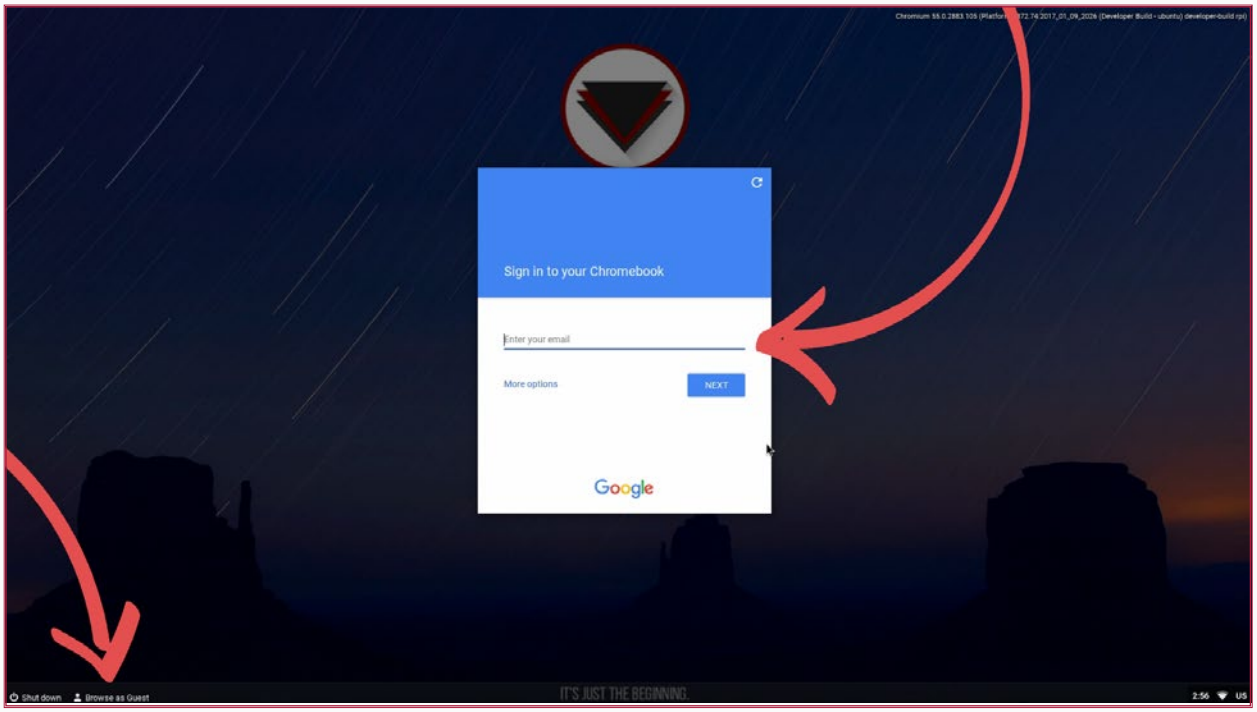
2. The Flint OS installation will now utilize your internet connection to check for any updates, just wait a few minutes for this to complete.



First Boot of Flint OS

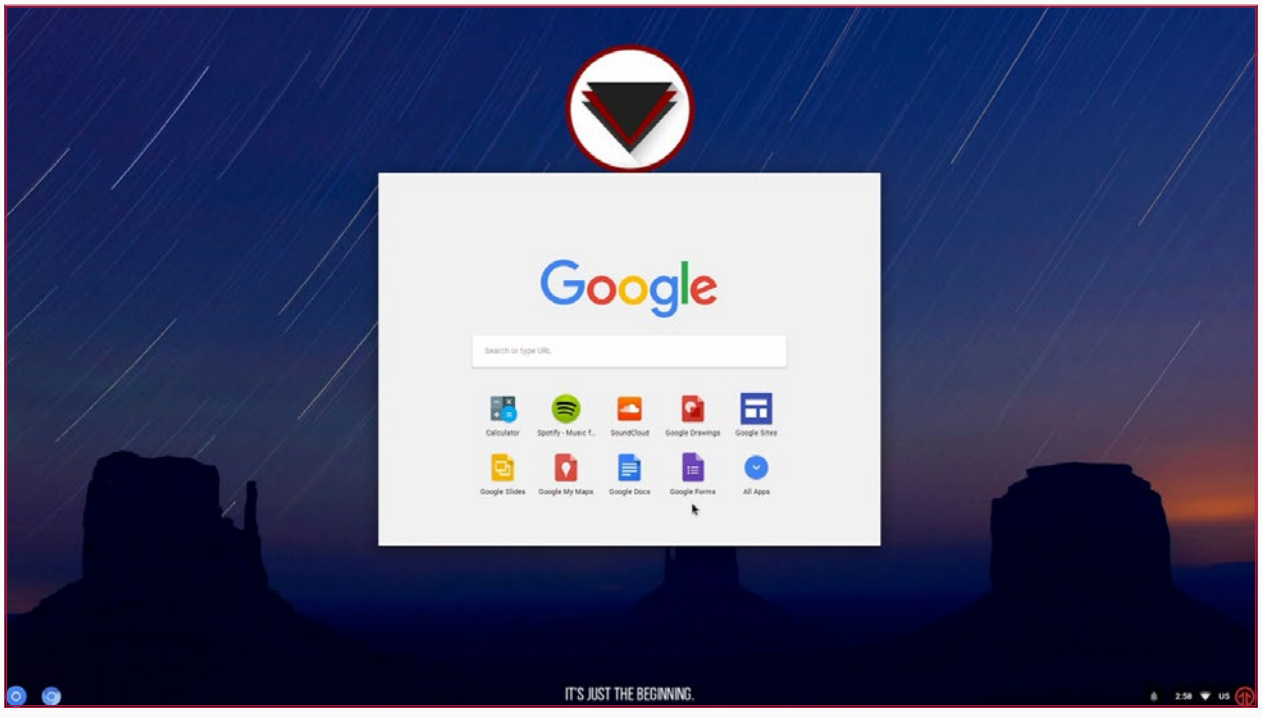
3. You will now be asked to log in to your Google account to continue on using Flint OS (Chromium) The operating system uses this to enhance your overall experience and automatically log you in to all Googles services.

Alternatively, if you would prefer not to use your Google Account you can continue as a guest.



4. Once done you will now be logged in to the operating system and it will start to grab any pre-existing information you might have from a previous usage of a Chormium based operating system.

You're now ready to use Flint OS!



Troubleshooting

At the time of writing this tutorial, Flint OS has a few issues here and there but is in much better shape than Chromium OS for SBC. Flint OS is also still in active development, so a lot of the issues are likely to be fixed in the future.

If you find your screen has frozen wait a minute as sometimes it lags and takes a bit of time to catch up. If it's still not doing anything, then you will need to reset the Pi.

Most of the basics work just fine such as profile changing, installation of apps, extensions from the web store and much more.

The Flint OS subreddit is probably the best place to go to talk about issues, development, and anything else related to the operating system.

You can find this subreddit at https://www.reddit.com/r/FlintOS/

Installing Windows 10 IoT

Project Description

In this tutorial, I will be going through the process of installing and setting up Windows 10 IoT Core for Raspberry Pi.

For those who don't know Windows 10 IoT Core is a version of the Windows 10 operating system built just for IoT devices such as the Raspberry Pi. Windows 10 IoT Core is perfect if you plan on utilizing something like UWP to write your application, it also gives you access to Windows's 10's core, and its various features.

I very briefly go into coding and pushing applications to the device. If you need to learn more about how to do things, then I highly recommend looking at some of Microsoft's documentation as it is comprehensive.

Please note to complete this tutorial you will need either a Raspberry Pi 2 or a Raspberry Pi 3. Windows 10 IoT Core is unsupported on other versions of the Raspberry Pi.

Equipment

Required

- Raspberry Pi
- SD Card (8 GB+ Recommended)
- Network Connection
- USB Keyboard
- USB Mouse
- HDMI Connection

Optional

- Raspberry Pi Case

Installing Windows 10 IoT on your Raspberry Pi

1. We will first need to download and install the Windows 10 IoT Core Dashboard. To download this, go to the following link: https://developer.microsoft.com/en-us/windows/iot/Downloads.

This piece of software is what will download the correct system for our Raspberry Pi and format it.

2. Insert your SD card into the computer or laptop's SD card reader and check the drive letter allocated to it, **e.g., G:/**. You will need to know this to ensure that you are formatting the correct drive, as you don't want to be doing this to any critical data.

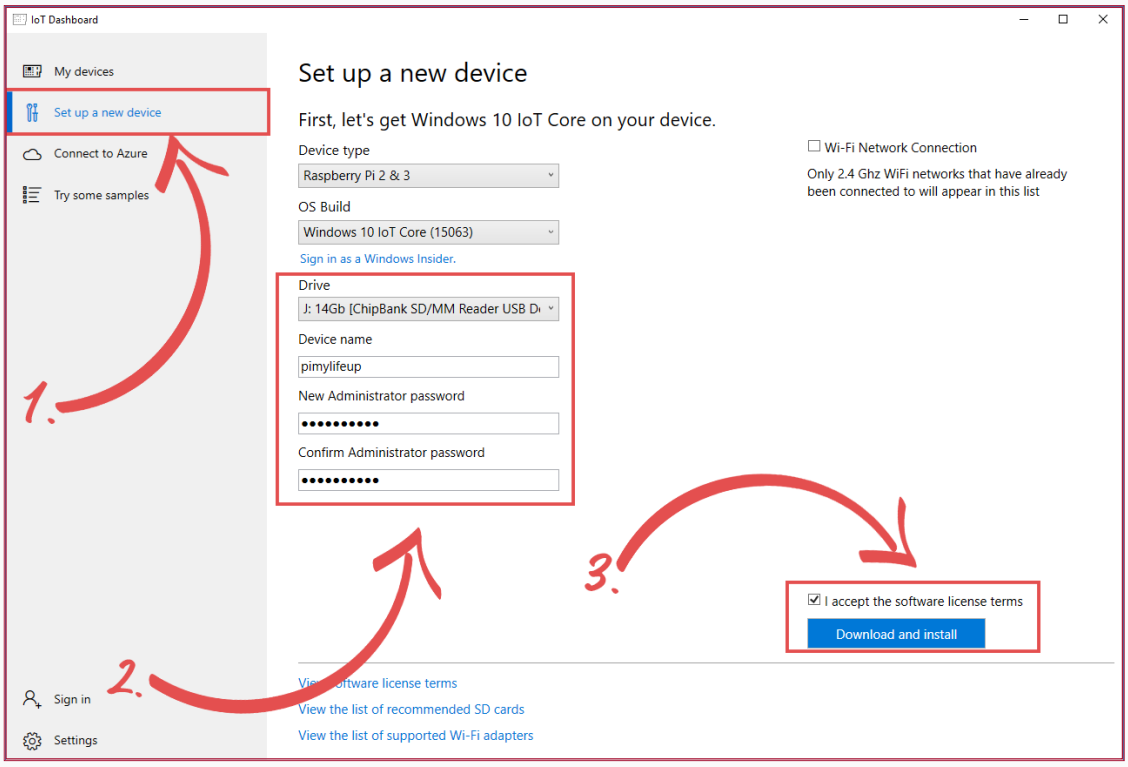
3. Now that you have inserted your SD Card into your computer/laptop, we will need to run the "**Windows 10 IoT Core Dashboard**" software. If you can't find this easily after installing it, then try running a search.

With the software loaded up, we need to go into the "**Set up a new device**" (**1.**) screen as shown below.

On here you will want to set your "**Device name**" and set the "**New Administration password**". Make sure that you set the password to something you can remember easily, but is secure, as this password is what you will use to connect to your Raspberry Pi remotely (**2.**).

Before we continue, make sure that "**Drive**" is set to the correct drive, make sure that the drive letter is the same as the SD Card that you inserted in step 2.

Once you have filled in your information, you should go ahead and tick the "**I accept the software license terms**" and then press the "**Download and install**" button (**3.**).



4. Once the software has finished downloading and installing Windows 10 IoT Core for Raspberry Pi, we can proceed on with this tutorial. Now safely to out your Micro SD card from your computer so you can put it into your Raspberry Pi.

Booting and setting up your Win 10 IoT device

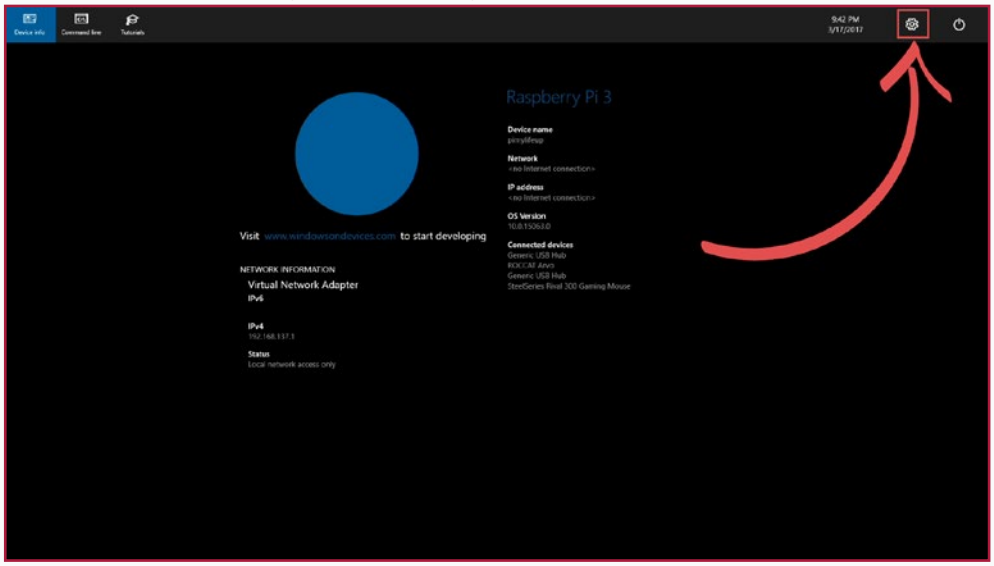
1. Now that we have successfully downloaded and written the image to our Raspberry Pi's Micro SD card we can insert the SD Card back into the Raspberry Pi.

2. Before we power back on the device, make sure that you plug in an HDMI cable and a mouse and keyboard, we will need all 3 of these if you intend on utilizing Wi-Fi on your Raspberry Pi Windows 10 IoT device.

Once done you can plug your Raspberry Pi back into power and allow it to start booting up.

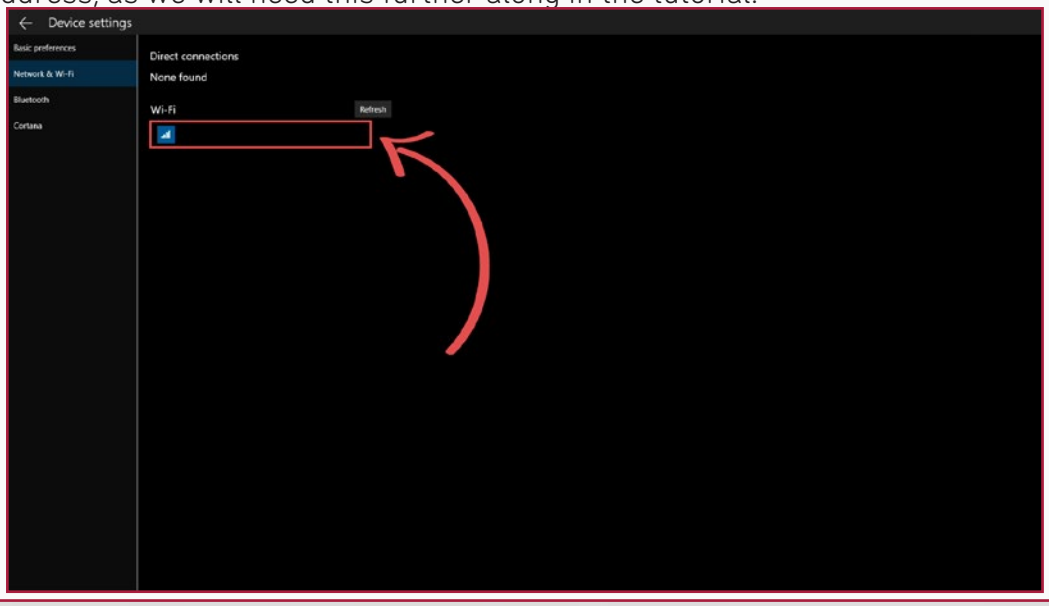
3. Now is the long wait for your Raspberry Pi to start up, when I did this it took a fair while for the Raspberry Pi to start up on boot, don't be afraid if you think it may have frozen it just takes some serious time to do the initial setup and startup.

4. Once it has finished starting up, you will be greeted with a screen like below. Now to set up a WiFi connection, we need to click the cog in the top right-hand corner.



5. Now in the next menu, we need to go to "**WiFi and Network**" and select the WiFi access point you want to connect to, you will receive a prompt asking you to enter your network password.

Once you have connected to your WiFi network, you can return to the main screen to grab your Raspberry Pi's IP Address, as we will need this further along in the tutorial.



Connecting to Your Device - Web Browser

There are three major ways of connecting to your Raspberry Pi Windows 10 IoT Device, all 3 of which we will quickly go over in this section.

Web Browser

First off is utilizing your web browser to talk with the Raspberry Pi, it is probably the easiest out of the three main ways to deal with.

All you need to do is point your Web Browser to your Raspberry Pi's IP Address on port 8080.

For example, my Raspberry Pi's local IP address is **192.168.0.143**, so in my favorite web browser I would type in **http://192.168.0.43:8080**

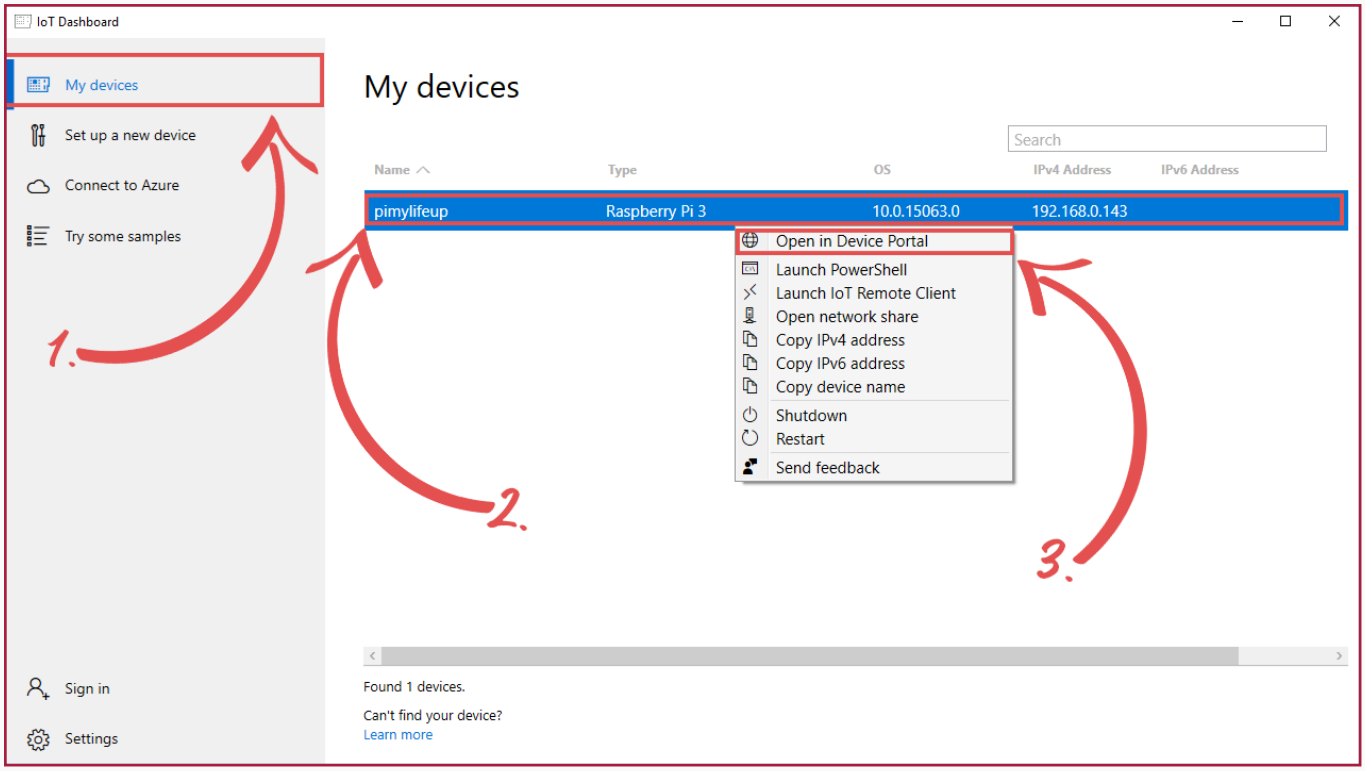
You can also use the "**Windows 10 IoT Core Dashboard**" tool to be able to click to get to the devices web page as well. Below we will run through the simple steps to using the tool to get to the device page.

1. To begin just simply load up the "**Windows 10 IoT Core Dashboard**" tool.

2. Go to the "**My Devices**" (**1.**) tab in the left sidebar.

3. **Right-click** (**2.**) on the device you want to make a connection to.

4. Finally click "**Open in Device Portal**" (**3.**) to take you to your Raspberry Pi's device portal.



Connecting to Your Device - Web Browser

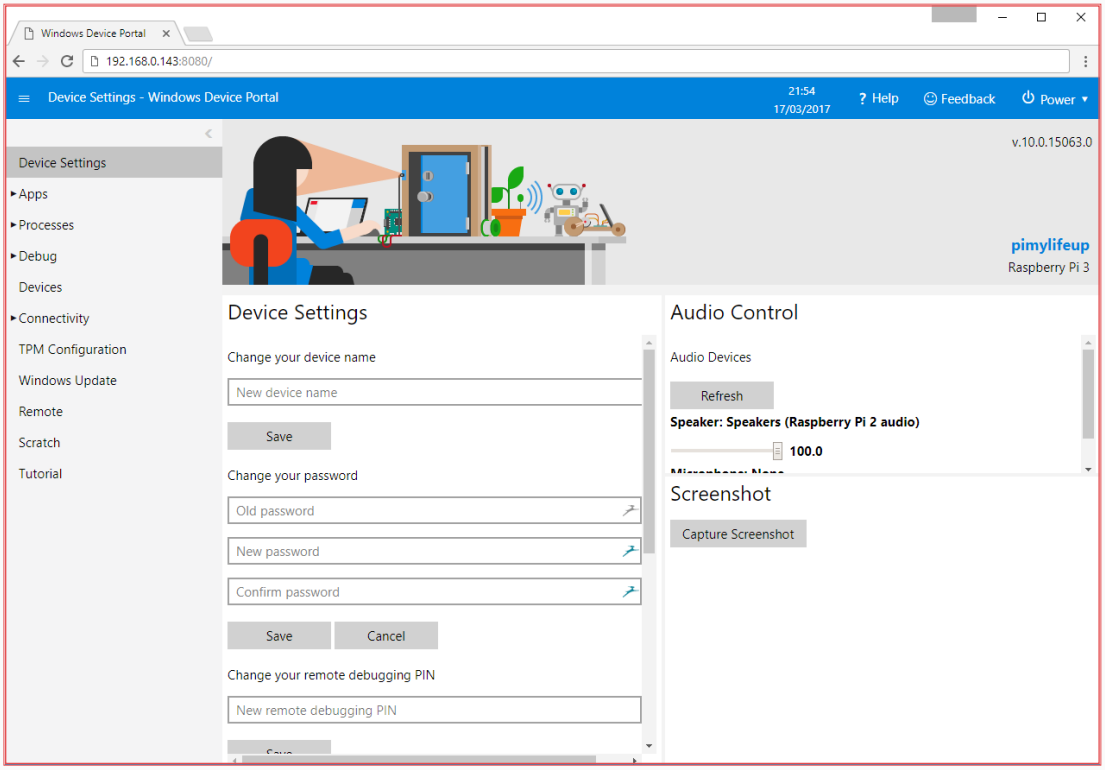
Upon either going to your Raspberry Pi's IP Address or using the Windows 10 IoT Core Dashboard tool, you will be first asked to log in.

Make sure you use **administrator** as the username, and the password you set at the beginning of this tutorial as the password.

Upon successfully logging in you should be greeted with the screen below.

We recommend exploring around as the web tool does offer a fair bit of access and incite to your device.

You can debug and see real-time performance through this interface which is incredibly helpful to look at what your Raspberry Pi is doing.



Connecting to Your Device - Powershell

PowerShell

PowerShell is not a tool that many will be too familiar with, but it is Microsofts more advanced version of command prompt giving you access to a wealth of tools including the ability to administer remote systems, a feature we will be making use of shortly.

PowerShell makes it rather simple to interact with your Raspberry Pi Windows 10 IoT device as we will show shortly.

There are two ways of connecting to your device through PowerShell.

The easier method relies on the "**Windows 10 IoT Core Dashboard**" tool (**Steps 1a+**), the other way is utilizing PowerShell to do everything (**Steps 1b+**).

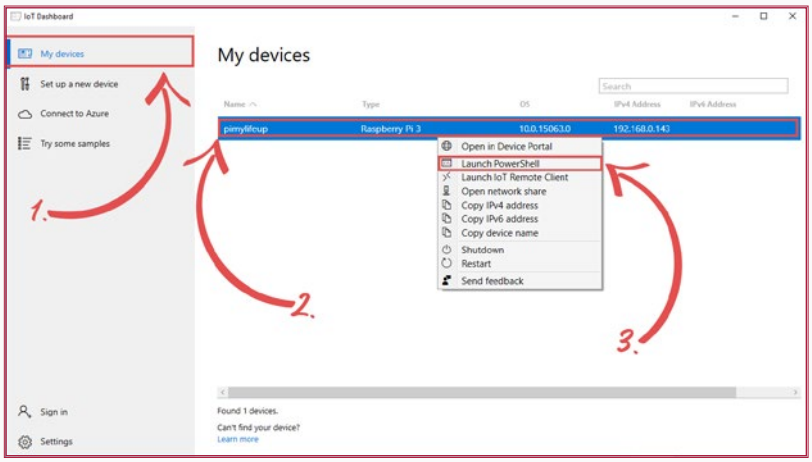
Connecting to Your Device - Powershell

1a. First off, we will explain the easy way, load up the "**Windows 10 IoT Core Dashboard**".

With the application open, go to the "**My Devices**" (**1.**) tab in the sidebar.

Right-click (**2.**) on the device you want to connect to.

Finally click "**Launch PowerShell**" (**3.**) in the pop up menu.



2a. This will launch a PowerShell session that will automatically begin to connect to your Pi.

When prompted enter the password we set at the start of this tutorial. You should be greeted with a PowerShell window like shown below when you have been successfully connected.

1b. The second way of connecting to your Raspberry Pi is slightly more complicated and utilizes PowerShell completely.

To open PowerShell on Windows 10, **right-click the windows Icon** and select "**Windows Powershell (Admin)**".

2b. In here we want to type in the following command, this adds our Raspberry Pi as a trusted device for Powershell to connect to.

Make sure you replace **[YOUR _PI_IP_ADDRESS]** with your Raspberry Pi's local IP address.

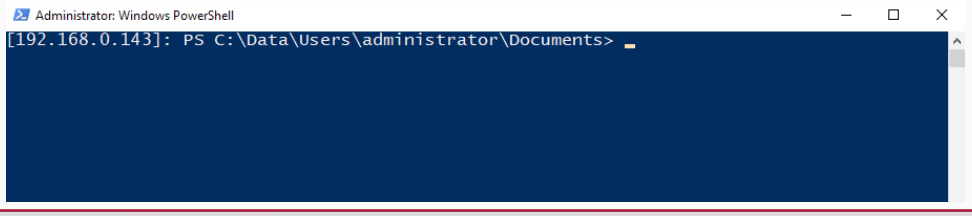
```
Set-Item WSMan:\localhost\Client\TrustedHosts -Value [YOUR _PI_IP_ADDRESS]
```

3b. With that done, we can now start a PowerShell session with our Raspberry Pi Windows 10 IoT device. To do this enter the command below into PowerShell, make sure that you replace **[YOUR _PI_IP_ADDRESS]** with your Raspberry Pi's local IP address.

```
Enter-PSSession -ComputerName [YOUR _PI_IP_ADDRESS]
-Credential [YOUR _PI_IP_ADDRESS]\Administrator
```

4b. You will be asked to enter the password you set earlier. Enter that to continue.

5b. After about 30 seconds, PowerShell should have now successfully made the connection, and you should see a screen like below.



SSH

The third way of interacting with your Raspberry Pi Windows 10 IoT device is to utilize SSH.

The main advantage of this is that it is a widely available protocol and is something most users of the Raspberry Pi will be thoroughly familiar with.

1. To start off make sure you have an SSH client installed, on Windows I highly recommend using either Putty or MobaXterm.

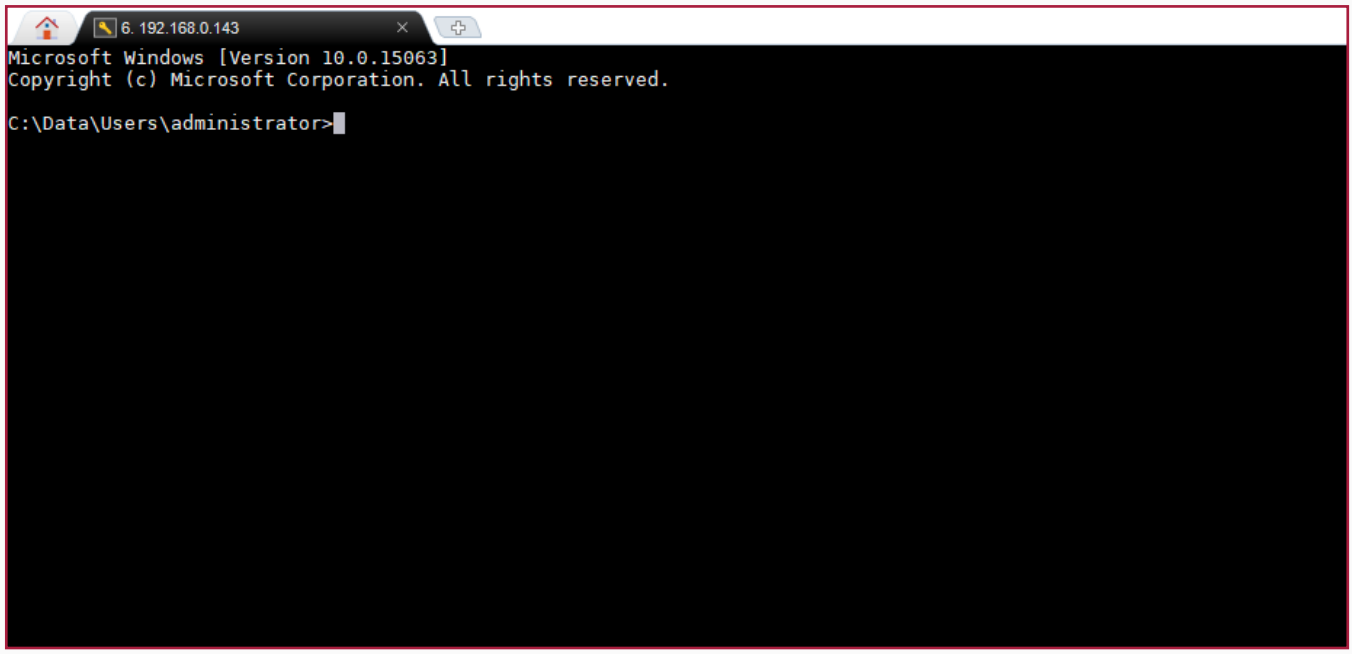
2. Now in your SSH Client connect to your Raspberry Pi's IP Address on port 22 (The default SSH port).

3. When asked to enter the username you want to log in with, make sure you use **Administrator**, as this is the default login username for Windows 10 IoT Core.

4. You will now be asked to enter the password associated with the account.

The password you want to use is the one you would have set within the **Windows 10 IoT Core Dashboard** at the start of this tutorial.

5. You should now be successfully logged into your Raspberry Pi Windows 10 IoT Core device and should be greeted with a screen like what is shown below.



If you want to learn more about utilizing SSH and some of the commands you can use within the session then make sure you take a look at Microsoft's own documentation at:
https://developer.microsoft.com/en-us/windows/iot/Docs/CommandLineUtils

Setting up Visual Studio for Windows 10 IoT Core

Lastly, you are most likely going to want to setup Visual Studio Community. The reason for this is so that you can start developing your applications for Windows 10 IoT Core.

Installation

1. First, we must download and install Visual Studio Community.

Luckily this is readily available on Microsoft's website, go through to their website by following this link: https://www.visualstudio.com/vs/community/

Be warned that the download and installation of Visual Studio Community can take some time especially on slow internet connections.

2. Once the installation process has completed, you can continue with the tutorial.

Start by launching up Visual Studio Community. It will ask you to do some configuration. It should be fine just to use the default settings.

3. Now one of the things you will find that is currently missing is any project templates for Windows 10 IoT Core.

We can grab and install these by going to the following link:
https://marketplace.visualstudio.com/items?itemName=MicrosoftIoT.WindowsIoTCoreProjectTemplates

4. Once you have downloaded and installed the templates, close and re-open Visual Studio, you need to do this for Visual Studio to load them in.

5. Now when creating a new project, you should be able to see that there is now Windows 10 IoT templates in the list. Select the one you wish to use.

6. Upon creating the new project, you will be prompted to activate developer mode on your Windows 10 device. Just follow the prompts provided to activate it.

7. Everything should now be ready for you to code your new application. You can find documentation on certain features of Windows 10 IoT Core by going to their documentation page at:
https://developer.microsoft.com/en-us/windows/iot/docs.

You can also find a document that explains how to utilize the GPIO pins from within Windows 10 IoT by going to https://developer.microsoft.com/en-us/windows/iot/docs/pinmappingsrpi.

Pushing code to the device.

1. Once you have your new application in a state in which you want to deploy it to your Raspberry Pi Windows 10 IoT device, go up to the tab that has a green arrow in it.

2. Click the black drop-down arrow, and select "**Remote machine**".

3. Now in here, you should be able to select your Raspberry Pi underneath automatic configuration, however, in some cases this will not function correctly, and you will have to enter the IP of your Raspberry Pi manually.

4. You should now be able to push code/applications to your Windows 10 IoT Raspberry Pi.

Overclocking
Your Raspberry Pi

Overclocking the Raspberry Pi

Be aware that overclocking may **reduce the lifetime of your Raspberry Pi**. If overclocking at a certain level causes system instability, try a more modest overclock.

Hold down shift during boot to temporarily disable the overclock.

There aren't any real "best settings" for overclocking your Raspberry Pi, and before we get started we must warn that overclocking does void the warranty of your Pi, it can also create various problems with the device, so use at your own risk.

Every Raspberry Pi's processor has a different upper limit on how far they can be pushed before problems will start occurring.

This difference in limits is due to the way CPU's are manufactured. CPU's are fabricated in large wafers.

This wafer contains many CPU's, not all of them come out the same during this manufacturing process.

Some will be much more capable of higher clock rates. Some will be best suited to sit at the default clock rate. The default factory clock rate is the guaranteed performance of the processors that come out of the wafer.

To be able to achieve the best possible overclocking performance for your Raspberry Pi, it is highly recommended to apply a heatsink to the processor.

This need for a heatsink is due to that as you overclock a processor, it drives more power into it, which increases the heat generated by the processor. If this heat isn't dissipated fast enough, it can cause some severe damage to the processor.

A heat sink works by increasing the total surface area that the heat is transferred to, while also being designed in a way air can quickly travel between it and cool the heatsink back down.

You must adequately test your overclocking settings to see how far you can push the Raspberry Pi before it starts to hit its limit and freeze. We will go into a method that you can use to see how capable your Raspberry Pi is at overclocking before it will start to fail over our next few pages.

We also highly recommend that you make sure that your power supply is sufficient (5V 2A), you will find most USB chargers don't provide a stable enough power supply to handle the Raspberry Pi while its overclocked.

Frequency Overclocking Options

arm_freq –
This setting dictates the frequency of the ARM processor. The higher the number, the faster it will try to run the processor.

gpu_freq –
This sets every GPU related frequency to the same value, so core_freq, h264_freq, isp_freq and v3d_freq will be set to this value.

core_freq –
GPU core frequency, On the Raspberry Pi 1 his has an impact on ARM performance since it drives L2 cache. This issue is no longer a problem with the Raspberry Pi 2 and later.

h264_freq –
The frequency of the hardware video block that renders the h264 codec.

isp_freq –
The frequency of the image sensor pipeline block GPU, this handles thing such as the camera.

v3d_freq –
The frequency of the 3D rendering block of the GPU.

sdram_freq –
The frequency of the SDRAM chip. This setting controls the speed at which the RAM of the Raspberry Pi runs.

Voltage Overclocking Options

(!) The voltages can't be chosen individually. They will all be set to the lowest voltage (!)
(!) The minimum is -16 the maximum 8, every step is 0.025 volt. The default is 0, with 1.2 volts (!)

over_voltage –
ARM and GPU core voltage adjustment, default

over_voltage_sdram –
sets all other SDRAM voltages together

over_voltage_sdram_c –
SDRAM controller voltage adjustment

over_voltage_sdram_i –
SDRAM I/O voltage adjustment

over_voltage_sdram_p –
SDRAM phy voltage adjustment

Getting into the Overclock Menu

Before you get started overclocking your Raspberry Pi, you first need to boot up **raspi-config**.

To do this, you need to first go into **Terminal**. There is an icon located on the Raspbian desktop. If you are already in a Terminal session, then all you need to do enter is the following command into it to get into the **raspi-config**.

```
sudo raspi-config
```

Upon typing in this command, you will be greeted with a screen like below. The options that we are interested in is Number **6 Overlocking**.

Navigate down to it using the **up** and **down** arrow keys.

Once you have highlighted the menu, you are after. Press **Enter** to select it.

```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password      Change password for the current user
2 Network Options           Configure network settings
3 Boot Options              Configure options for start-up
4 Localisation Options      Set up language and regional settings to match your location
5 Interfacing Options       Configure connections to preipherals
6 Overclock                 Configure overclocking for your Pi
7 Advanced Options          Configure advanced settings
8 Update                    Update this tool to the latest version
9 About raspi-config        Information about this configuration tool

          <Select>                <Finish>
```

After pressing Enter you will see the following warning, if you are worried about any problems occurring with your Raspberry Pi here is an excellent spot to stop proceeding with the guide.

```
Be aware that overclocking may reduce the lifetime of your
Raspberry Pi. If overclocking at a certain level causes
system instability, try a more modest overclock. Hold down
shift during boot to temporarily disable overclock.
See http://elinux.org/RPi_Overclocking for more information
```

Raspberry Pi Overclocking Options

The various overclocking options that are available to you will purely depend on what Raspberry Pi version you are running. Both the Raspberry Pi Zero and the Raspberry Pi 3 both aren't supported within the **raspi-config** tool as of publication of this book.

We will quickly run through the available overclocking options for both the Raspberry Pi 1 and Raspberry Pi 2

Raspberry Pi 1 Overclocking

```
Choose overclock preset

None      700MHz ARM, 250MHz core, 400MHz SDRAM, 0 overvolt
Modest    800MHz ARM, 250MHz core, 400MHz SDRAM, 0 overvolt
Medium 900MHz ARM, 250MHz core, 450MHz SDRAM, 2 overvolt
High      950MHz ARM, 250MHz core, 450MHz SDRAM, 6 overvolt
Turbo    1000MHz ARM, 500MHz core, 600MHz SDRAM, 6 overvolt
```

The Raspberry Pi 1 is the most open Raspberry Pi to being overclocked. This is largely due to its modest default clock rate of 700MHz.

It also only had the single ARM core, so there was a much larger payoff for users to overclock to run more intensive tasks compared to the Raspberry Pi 2 and Raspberry Pi 3 that both boasted quad-core processors and higher default clocked cores.

Any overclocking that requires overvolt means you will need to have a very stable power supply that can supply the right amount of voltage and amperage to the Raspberry Pi.

If you would like to overclock the Raspberry Pi 1 with little to no consequences, Modest should be a safe option as it requires no extra voltage to the processor.

Advanced Raspberry Pi 1 Overclocking

There is a more advanced option for overclocking the Raspberry Pi 1. This option requires modifying the config file that stores all the overclocking values and tweaking each one individually.

This config file allows you to overclock the Raspberry Pi 1 further than the **raspi-config** tool allows, as you can tweak each value. Editing these values requires some understanding of overclocking, so be careful with fiddling with the values.

The **Raspi-config** tool already offers quite an extensive list of options for overclocking the Raspberry Pi 1, so we highly recommend utilizing the options available in the tool before going and fiddling with the values yourself.

Modifying the file is recommended for **advanced** users only.

To achieve this, begin by editing the config file present on the SD Card at **/boot/config.txt**

Raspberry Pi 2 Overclocking

```
Choose overclock preset

None      900MHz ARM, 250MHz core, 400MHz SDRAM, 0 overvolt
High     1000MHz ARM, 500MHz core, 500MHz SDRAM, 2 overvolt
```

The Raspberry Pi 2 boasted a massive increase in the Raspberry Pi's power, bringing with it a 900MHz default clock, and 4 cores. The Pi 2 offers a significant improvement over the Raspberry Pi 1's 700MHz, single core processor.

However, despite the performance increase, there is always more to be squeezed out. Unlike the Raspberry Pi 1, the Raspberry Pi 2 only has one extra overclocking option in **raspi-config**.

That is the "**High**" overclocking option, while only being a somewhat slight overclock it does require you to have a stable power supply.

Please note, using the raspi-config tool is not the only way to overclock a device. You can also overclock by modifying **/boot/config.txt**,but that is only for advanced users who know what tweaking each value does.

Advanced Raspberry Pi 2 Overclocking

There is a more advanced option for overclocking the Raspberry Pi 2. This option requires modifying the config file that stores all the overclocking values and tweaking each one individually.

Modifying this file allows you to overclock the Raspberry Pi 2 further than the raspi-config tool allows, as you can tweak each value. Changing this file requires some understanding of overclocking, so be careful with fiddling with the values. We explain each configurable variable on **Page 23**.

Changing these values are recommended for **advanced** users only.

To achieve this, begin by editing the config file present on the SD Card at **/boot/config.txt**.

Below we have some tested overclocks for the Raspberry Pi 2. Please note these can cause some issues so if you are interested in **advanced** overclocking, it is worth using these as a starting base.

```
arm_freq=1050
gpu_mem_512=384
core_freq=500
h264_freq=0
isp_freq=0
avoid_pwm_pll=1
sdram_freq=550
over_voltage=6
avoid_safe_mode=1
force_turbo=0
```

If the overclocks cause any issues, you can try holding **Shift** during boot up to revert to default clocks.

Or modify the **/boot/config.txt** file to remove any overclocks.

Raspberry Pi 3 Overclocking

The Raspberry Pi 3 along with the Raspberry Pi Zero are the more complicated Raspberry Pi's to overclock. At publication, the **raspi-config** tool had no support for overclocking the Raspberry Pi 3.

Instead of relying on the **raspi-config** tool to overclock the Raspberry Pi 3 you will have to tweak the values in **/boot/config.txt** manually.

Having to edit the values manually makes the device an awful lot more complicated to overclock, as instead of using a pre-tested and pre-thought out overclock you must tweak all the values individually.

On the next page, we provide a method for testing how stable the overclock is for your Raspberry Pi.

The script will be incredibly useful for overclocking the Raspberry Pi 3 and testing how the previous Raspberry Pi's handle an overclock.

Two things to note about the Raspberry Pi 3, is that even at its base clock rate it generates **a lot** of heat. So before you get started overclocking the Raspberry Pi 3 it is a requirement to have a heatsink placed onto the Processor of the Raspberry Pi 3 to ensure any overclock will remain stable.

It is also a requirement for the Raspberry Pi 3 to have a **sufficient power supply**. Ensure that the adapter you are using for it provides a guaranteed **5v 2500 mA**.

Try to steer away from cheap power supplies or USB chargers as they often don't provide the right amount of power, which will cause the overclock to be unstable.

Advanced Raspberry Pi 3 Overclock

This option requires some understanding of overclocking, so be careful with fiddling with the values. We explain each configurable variable earlier on in the book.

This option is recommended for **advanced** users only.

To achieve this, begin by editing the config file present on the SD Card at **/boot/config.txt**.

Below we have some tested and somewhat stable settings for you to start off utilizing. They should provide a decent speed increase, but again you should be careful as this can lower the lifetime of the device. Use the script located over the page to test the overclock.

```
#Overclock Settings
arm_freq=1350
over_voltage=6
temp_limit=80
core_freq=500

#GPU Based
h264_freq=333
avoid_pwm_pll=1
gpu_mem=450
v3d_freq=500

#Ram Overclock
sdram_freq=588
sdram_schmoo=0x02000020
over_voltage_sdram_p=6
over_voltage_sdram_i=4
over_voltage_sdram_c=4

#Sound Fix
hdmi_drive=2
```

Testing your Raspberry Pi Overclock

After overclocking, any of the Raspberry Pi's you will want to know whether it is stable.
While some overclocks will appear to function correctly it is not a guarantee. The script we utilize below pushes the Raspberry Pi to its limit.

This script intends to do multiple things. The first is to max out the CPU utilization. This script will force it to hit your overclocked speed, heating up the Raspberry Pi and putting the power supply under load.

Secondly, the script reads the entire contents of the SD Card. This script tests the ability of the RAM to sustain its clock rate, it also ensures that the I/O interface of the Raspberry Pi isn't suffering from any issues.

Finally, it tests to ensure that the SD Card won't be corrupted by the overclock, it achieves this by writing a 512mb file to the SD Card multiple times. This script will help detect whether the overclock is causing any corruption that will cause significant issues when using your Raspberry Pi every day.

First off, we need to write the bash script to the Raspberry Pi for you to utilize it. Using your favorite text editor such as **nano** or **vi**, create a new file at **/home/pi/stresstest.sh** and write the following script to the file.

```
#!/bin/bash
#Simple stress test for system. If it survives this, it's probably stable.
#Free software, GPL2+

echo "Testing overclock stability..."

#Max out the CPU in the background (one core). Heats it up, loads the power-supply.
nice yes >/dev/null &

#Read the entire SD card, 2x. Tests RAM and I/O
for i in {1..2}; do echo reading: $i; sudo dd if=/dev/mmcblk0 of=/dev/null bs=4M; done

#Writes 512 MB test file,  5x.
for i in {1..5}; do echo writing: $i; dd if=/dev/zero of=deleteme.dat bs=1M count=512; sync;
done

#Clean up
killall yes
rm deleteme.dat

#Print summary. Anything nasty will appear in dmesg.
echo -n "CPU freq: " ; cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
echo -n "CPU temp: " ; cat /sys/class/thermal/thermal_zone0/temp
dmesg | tail

echo "Not crashed yet, probably stable."
```

Testing your Raspberry Pi Overclock

Once you are certain you, have written the script correctly. Save it.

Before we go running this script though, we must give the script execution privileges otherwise it will fail to run correctly.

We achieve this by typing the following command into the terminal.

```
sudo chmod +x /home/pi/stresstest.sh
```

Now, all we must do is run the script. Please note this has a high chance of causing your Raspberry Pi to either crash entirely or freeze up. If it does, this indicates that the overclock you have applied to your Raspberry Pi is unstable, and you should try dialing it back.

The script will also show errors from the command **dmesg**, if any errors show, it makes an indication that the overclock is NOT stable. Again, as before, try dialing back the overclocks.

Any SD Card issues are caused by an overclock to the **core_freq**. You should attempt to dial this back to reduce the chance of the SD Card being corrupted.

Once you are ready, run the following command to start the stress test. Be prepared to give this a few minutes to complete.

```
bash /home/pi/stresstest.sh
```

Once completed successfully this should give you a few bits of information back.
It will give you the current running speed of the processor, which after the test should be about the same as you overlock.

The second value it gives back is the current temperature of the processor, the two front digits of this number are the ones you want to pay the most attention to. For instance, 43850 will be 43°C. Anything above 63°C is highly likely to cause issues with your Raspberry Pi.

There are a few ways to solve a high temperature on your Raspberry Pi, firstly is to install a heatsink, secondly make use of a case that provides a good airflow design, and lastly, you can always try to install a small fan to the Raspberry Pi to try and cool the device down faster.
Of course, the simpler and cheaper option is just to dial the overclock back to reduce the extra strain on the processor.

After the processor speed and temperature, the dmesg will print the message buffer from the kernel. This message will be an important message to read. Pay attention to any errors here.

If there are no errors here, it means that you have successfully overclocked your Raspberry Pi.


1402
1402

References

Linux Cheat Sheet

File System

ls	Directory Listing
tree	Indented file/directory listing
cd	Change directory
pwd	Display current directory
mkdir newDir	Creates a directory
rmdir oldDir	Removes a directory
cp file newfilename	Copies a file to a new location
mv file.txt ./newDir	Moves a file to a new location
touch file	Sets the last modified timestamp
cat	List contents of a file
head file	Outputs first 10 lines of a file
tail file	Outputs last 10 lines of a file
chmod 777 file	CHMOD is used to alter the permissions of a file or files.
chmod u=rw file	You can use symbols or numbers depending on what you prefer
chown pi:root file	CHOWN will change the user and/or the group that owns a file.
unzip archive.zip	Unzip will extract the files and directories from a compressed zip archive.
tar -cvzf **tar** -xvzf	Compress and extract the contents of an archive in the tar format. -c to compress & -x to extract

Users Management

id	Get the id of a user or group
who	List users that are logged in
last	List of users recently logged in
groupadd name	Adds a new group
useradd name	Adds a new user
userdel name	Deletes a user & all files related to that user
deluser name	Deletes user with options to remove certain data
usermod	Modify a user account
passwd	Change your password or a password of another account

Process Management

ps	Show snapshot of current processes
top	Show real time processes
kill pid	Kill process with id pid
pkill name	Kill process with name
killall name	Kill processes with name

Networking

ping host	Pings a host
hostname	Hostname of the system
ifconfig	Shows network configuration details for the interfaces on current system
ssh	Connect to another computer using SSH
scp	Transfer files over SSH
wget {URL}	Downloads a file directly to the device
netstat -tupl	Active connections to/from device

Shortcuts

!!	Repeats last command
ctrl+z	Stops command, resume using **fg** for foreground or **bg** for background
ctrl+c	Halts the current command
ctrl+w	Deletes 1 word on the current line
ctrl+u	Deletes entire line
ctrl+r	Search previous commands
exit	log out of current session

General Commands

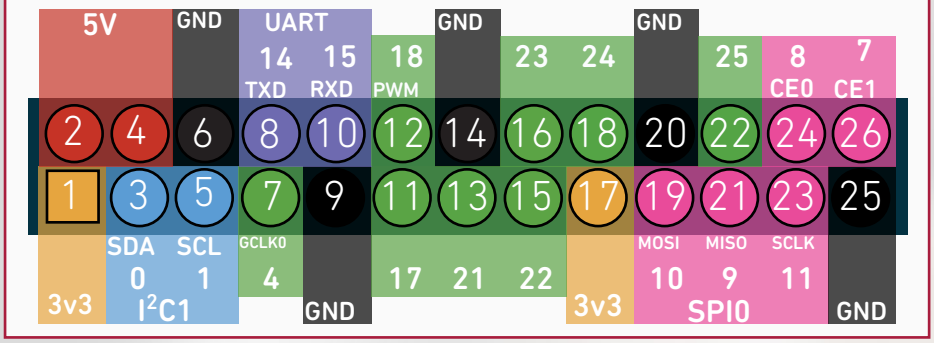
Man ls	Brings up the manual page for ls
ls **|** head	Pipes	allows you to direct output from one command into another
date	Shows system date
whoami	Shows your username
uptime	Shows uptime
uname -a	Show system & kernel

Searching

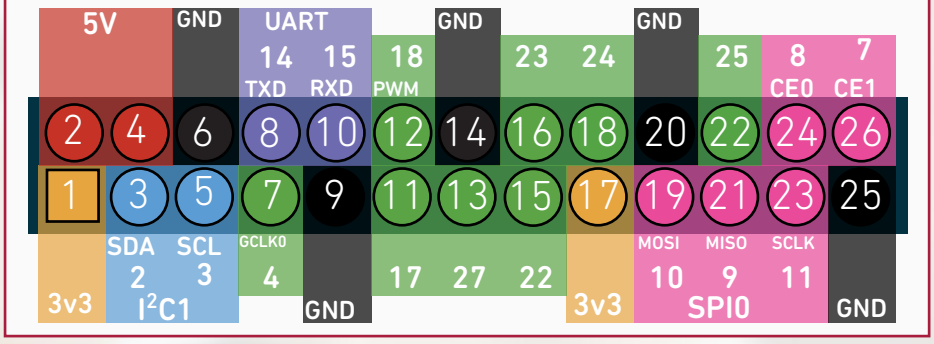
grep "search" *.txt	Search for a pattern inside files
find . -name 'help'	Will search for directories & files that match a pattern
whereis ls	Displays documentation, binaries & source files of a command

Raspberry Pi GPIO Pins

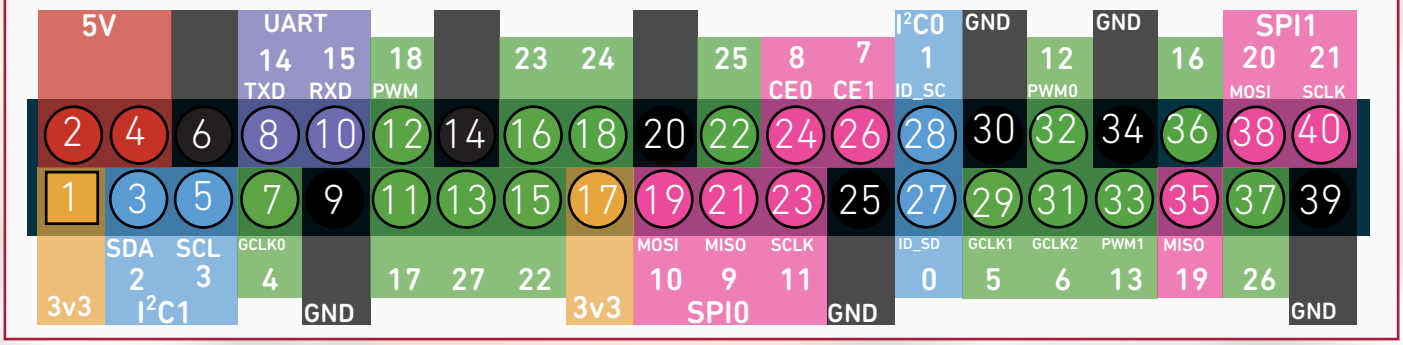
Raspberry Pi (Revision 1)



Raspberry Pi (Revision 2)



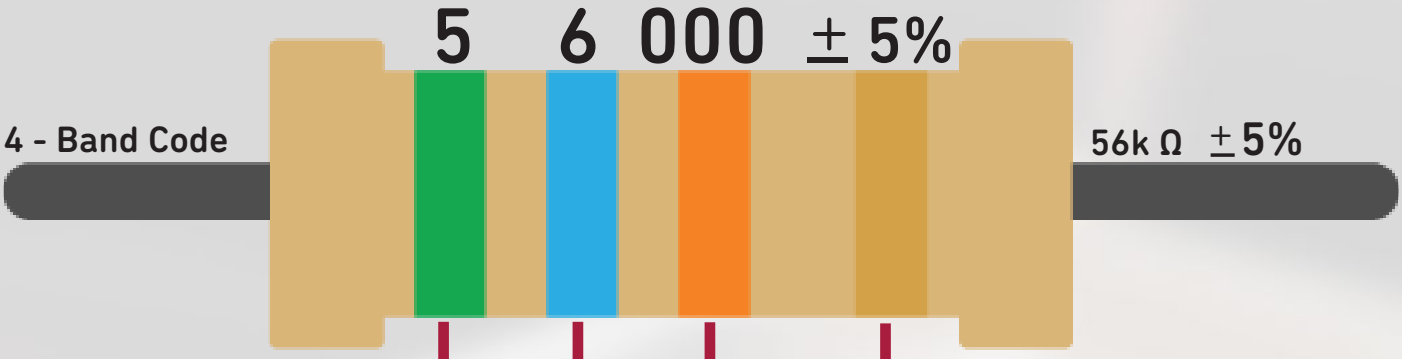
Raspberry Pi B+, 2, 3 & Zero



Key

- GPIO Pins
- Ground
- Serial Peripheral Interface (SPI)
- Universal asynchronous receiver/transmitter (UART)
- Inter-Intergrated Circuit (I²C)
- 5v Power Pins
- 3v3 Power Pins

Resistor Colour Guide



1st Digit	2nd Digit	3rd Digit	Multiplier	Tolerance
0	0	0	1	
1	1	1	10	1%
2	2	2	100	2%
3	3	3	1000	
4	4	4	10000	
5	5	5	100000	
6	6	6	1000000	
7	7	7		
8	8	8	0.1 Gold	5% Gold
9 White	9 White	9 White	0.01 Silver	10% Silver

Voltage Divider Equation

Calculating the Voltage Out

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

Calculating the R2 Value

$$R_2 = \frac{V_{out} * R_1}{V_{in} - V_{out}}$$