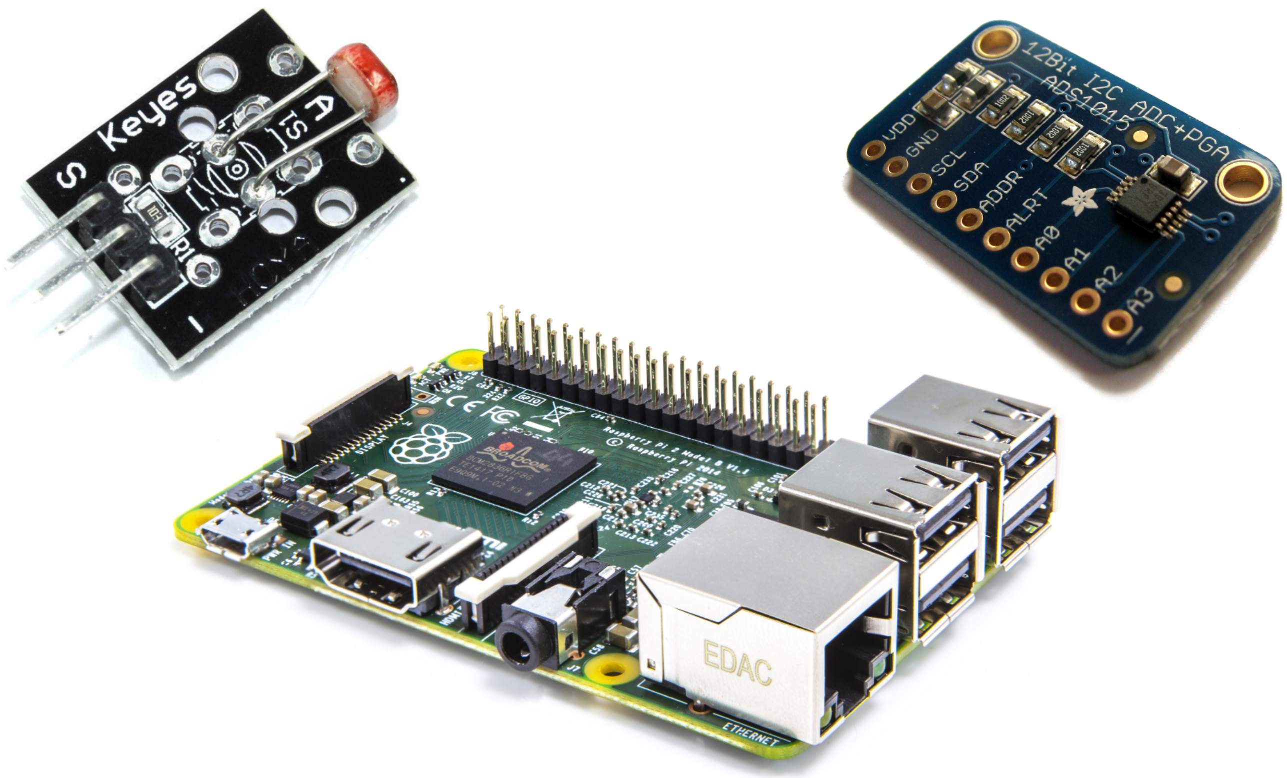
Raspberry Pi Computing

Analog Measurement



Contents

Introduction

Welcome!

Hi there. Congratulations on getting your hands on this book. You're interested in learning about connecting analog sensors to the Raspberry Pi. So, you've come to the right place.

This will be a journey of discovery for both of us. By experimenting with computers we will be learning about what is happening in the physical environment. Others have done this sort of thing, but I have an ulterior motive. I write books to learn and document what I've done. The hope is that by sharing the journey others can learn something from my efforts :-).

Ambitious? Maybe :-). But if you're reading this, I managed to make some headway. I dare say that like other books I have written (or are currently writing) it will remain a work in progress. They are living documents, open to feedback, comment, expansion, change and improvement. Please feel free to provide your thoughts on ways that I can improve things. Your input would be much appreciated.

You will find that I eschew a simple "Do this approach" for more of a story telling exercise. Some explanations are longer and more flowery than might be to everyone's liking, but there you go, that's my way :-).

There's a **lot** of information in the book. There's 'stuff' that people with a reasonable understanding of computers will find excessive. Sorry about that. I have gathered a lot of the content from other books I've written to create this guide. As a result, it is as full of usable information as possible to help people who could be using the Pi and coding for the first time. Please bear in mind, this is the description of *ONE* simple project. I could describe it in 5 pages but I have stretched it out into a *lot* more. If we need to recreate the project from scratch, this guide will leave nothing out. It will also form a basis for other derivative books (as books before this one have done). As the Raspberry Pi's and OS's improve, the descriptions will evolve.

I'm sure most authors try to be as accessible as possible. I'd like to do the same, but be warned… There's a good chance that if you ask me a technical question I may not know the answer. So please be gentle with your emails :-).

Email: d3noobmail+analog@gmail.com

Cover photo via Good Free Photos[1] and fluxworkshop[2].

[1]https://www.goodfreephotos.com
[2]https://www.ebay.com/usr/fluxworkshop

What are we trying to do?

Put simply, we are going to examine the wonder that is the Raspberry Pi computer and use it to accomplish something.

In this specific case we will be connecting an analog sensor (specifically a **L**ight **D**ependent **R**esistor (LDR)) to an **A**nalog to **D**igital **C**onverter (ADC) which will be connected to the Pi. We'll be measuring the values that it returns, recording them in a database and then making those values available via web a interface!

Along the way we'll;

- Look at the Raspberry Pi and its history.
- Work out how to get software loaded onto the Pi.
- Learn about networking and configure the Pi accordingly.
- Install and configure a web server and a database.
- Write some code to interface with our ADC and our LDR.

Who is this book for?

You!

By getting hold of a copy of this book you have demonstrated a desire to learn, to explore and to challenge yourself. That's the most important criteria you will want to have when trying something new. Your experience level will come second place to a desire to learn.

It may be useful to be comfortable using the Windows operating system (I'll be using Windows 7 for the set-up of the devices). You should be aware of Linux as an alternative operating system, but you needn't have tried it before. Before you learn anything new, it pretty much always appears indistinguishable from magic. but once you start having a play, the mystery falls away.

What will we need?

Well, you could just read the book and learn a bit. By itself that's not a bad thing, but trust me when I say that actually experimenting with physical computers is fun and rewarding.

The list below is flexible in most cases and will depend on how you want to measure the values.

- A Raspberry Pi (I'm using a Raspberry Pi Model B 2 / 3)
- Probably a case for the Pi
- A MicroSD card
- A power supply for the Pi
- A keyboard and monitor that you can plug into the Pi (there are a few options here, read on for details)
- A remote computer (like your normal desktop PC that you can use to talk to connect to the Pi). This isn't *strictly* necessary, but it makes the experience *way* cooler.

- A Keyes KY-018 LDR[3]. They are available from lots of places for around $2 US.
- An ADS1015 ADC from Adafruit[4]. The ADS1015 has a 12bit resolution giving it the ability to convert an analog signal into one of 4096 discrete levels.
- Some 2.54mm header pins for the ADC module (these are widely available) and some soldering equipment (you could solder directly, but that's not as flexible).
- Some dupont connectors (that's what I used, but you could connect to the Pi and the modules in different ways).
- An Internet connection for getting and updating the software.

As we work through the book we will be covering off the different parts required and you should get a good overview of what your options are in different circumstances.

Why on earth did I write this rambling tome?

That's a really good question. This is another project that I wanted to update from an earlier book (Raspberry Pi: Measure, Record, Explore[5]) and to be brutally hones I picked it at random over other options. Writing the previous books in this series[6] was an enjoyable process, so I thought that I'd carry on and continue to adapt the book for subsequent projects. This is book three in this series, so I suppose it's a 'thing' by now. Will this continue? Who knows, stay tuned…

Included is a bunch of information from my books on the Raspberry Pi, Linux and d3.js. I hope you find it useful.

Where can you get more information?

The Raspberry Pi as a concept has provided an extensible and practical framework for introducing people to the wonders of computing in the real world. At the same time there has been a boom of information available for people to use them. The following is a far from exhaustive list of sources, but from my own experience it represents a useful subset of knowledge.

raspberrypi.org[7]

Google+[8]

reddit[9]

Raspberry Pi Stack Exchange[10]

[3]https://www.google.co.nz/search?q=Keyes+KY-018+LDR
[4]http://www.adafruit.com/products/1083
[5]https://leanpub.com/RPiMRE
[6]https://leanpub.com/b/rpc
[7]https://www.raspberrypi.org/
[8]https://plus.google.com/u/0/communities/113390432655174294208
[9]https://www.reddit.com/r/raspberry_pi/
[10]https://raspberrypi.stackexchange.com/questions?sort=newest

The History of the Raspberry Pi

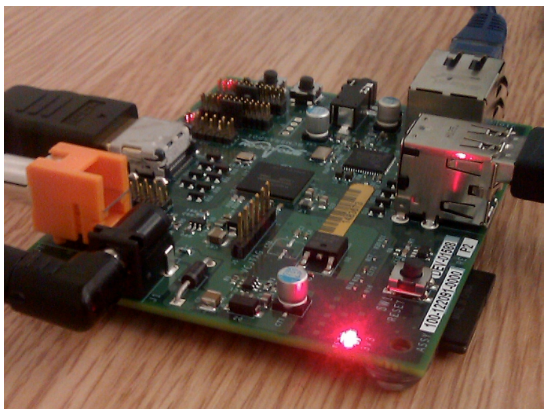
The story of the Raspberry Pi starts in 2006 at the University of Cambridge's Computer Laboratory. Eben Upton, Rob Mullins, Jack Lang and Alan Mycroft became concerned at the decline in the volume and skills of students applying to study Computer Science. Typical student applicants did not have a history of hobby programming and tinkering with hardware. Instead they were starting with some web design experience, but little else.

They established that the way that children were interacting with computers had changed. There was more of a focus on working with Word and Excel and building web pages. Games consoles were replacing the traditional hobbyist computer platforms. The era when the Amiga, Apple II, ZX Spectrum and the 'build your own' approach was gone. In 2006, Eben and the team began to design and prototype a platform that was cheap, simple and booted into a programming environment. Most of all, the aim was to inspire the next generation of computer enthusiasts to recover the joy of experimenting with computers.

Between 2006 and 2008, they developed prototypes based on the Atmel ATmega644 microcontroller. By 2008, processors designed for mobile devices were becoming affordable and powerful. This allowed the boards to support an graphical environment. They believed this would make the board more attractive for children looking for a programming-oriented device.

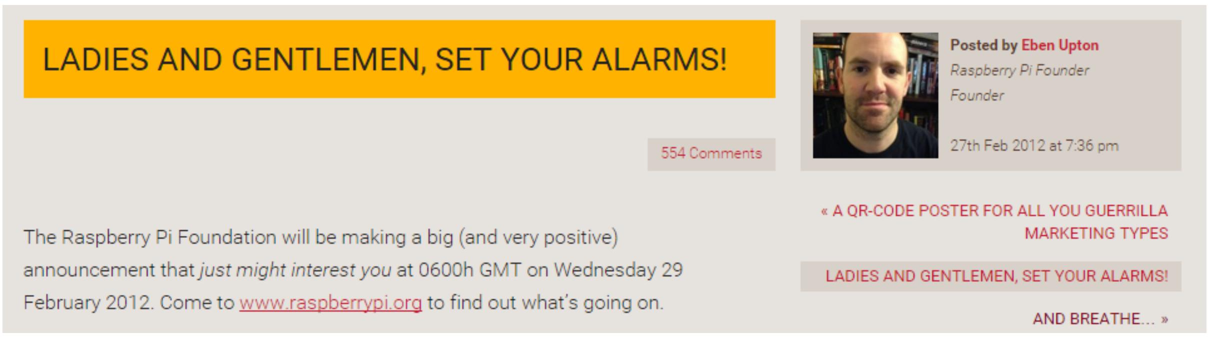
Eben, Rob, Jack and Alan, then teamed up with Pete Lomas, and David Braben to form the Raspberry Pi Foundation. The Foundation's goal was to offer two versions of the board, priced at US$25 and US$35.

50 alpha boards were manufactured in August 2011. These were identical in function to what would become the model B. Assembly of twenty-five model B Beta boards occurred in December 2011. These used the same component layout as the eventual production boards.


Early Alpha Board (Credit: Paul Downey)

Interest in the project increased. They were demonstrated booting Linux, playing a 1080p movie trailer and running benchmarking programs. During the first week of 2012, the first 10 boards were put up for auction on eBay. One was bought anonymously and donated to the museum at The Centre for Computing History in Suffolk, England. While the ten boards together raised over 16,000 Pounds (about $25,000 USD) the last to be auctioned (serial number No. 01) raised 3,500 Pounds by itself.

The Raspberry Pi Model B entered mass production with licensed manufacturing deals through element 14/Premier Farnell[11] and RS Electronics[12]. They started accepting orders for the model B on the 29th of February 2012. It was quickly apparent that they had identified a need in the marketplace. Servers struggled to cope with the load placed by watchers repeatedly refreshing their browsers. The official Raspberry Pi Twitter account reported that Premier Farnell sold out within few minutes of the initial launch. RS Components took over 100,000 pre orders on the first day of sales.


LADIES AND GENTLEMEN, SET YOUR ALARMS!

554 Comments

Posted by Eben Upton
Raspberry Pi Founder
Founder

27th Feb 2012 at 7:36 pm

The Raspberry Pi Foundation will be making a big (and very positive) announcement that *just might interest you* at 0600h GMT on Wednesday 29 February 2012. Come to www.raspberrypi.org to find out what's going on.

« A QR-CODE POSTER FOR ALL YOU GUERRILLA MARKETING TYPES
LADIES AND GENTLEMEN, SET YOUR ALARMS!
AND BREATHE… »

raspberrypi.org blog lights the fuse.

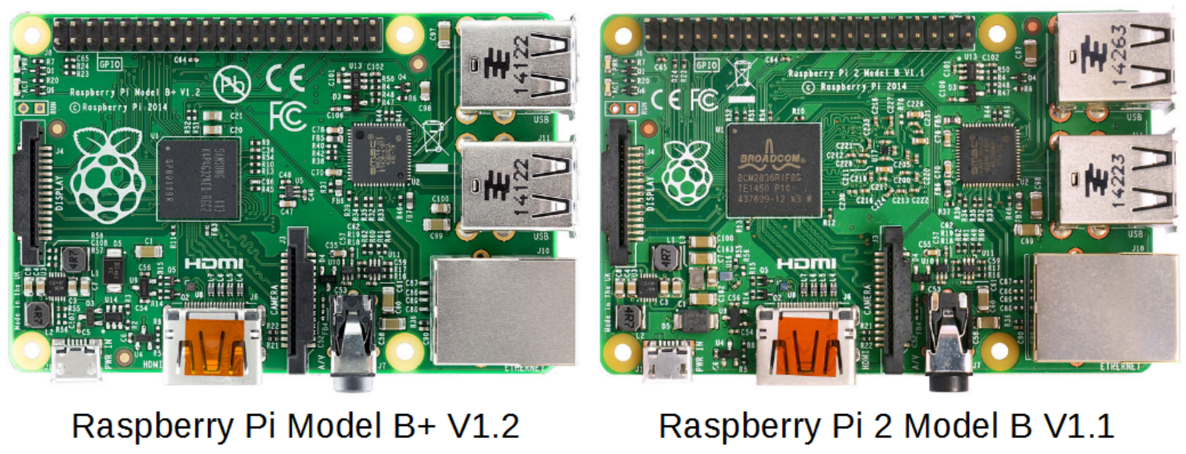
Within two years they had sold over two million units.

The the lower cost model A went on sale for $25 on 4 February 2013. By that stage the Raspberry Pi was already a hit. Manufacturing of the model B hit 4000 units per day and the amount of on-board ram increased to 512MB.

The official Raspberry Pi blog reported that the three millionth Pi shipped in early May 2014. In July of that year they announced the Raspberry Pi Model B+, "*the final evolution of the original Raspberry Pi. For the same price as the original Raspberry Pi model B, but incorporating numerous small improvements*". In November of the same year the even lower cost (US$20) A+ was announced. Like the A, it would have no Ethernet port, and just one USB port. But, like the B+, it would have lower power requirements, a micro-SD-card slot and 40-pin HAT compatible GPIO.

On 2 February 2015 the official Raspberry Pi blog announced that the Raspberry Pi 2 was available. It had the same form factor and connector layout as the Model B+. It had a 900 MHz quad-core ARMv7 Cortex-A7 CPU, twice the memory (for a total of 1 GB) and complete compatibility with the original generation of Raspberry Pis.

[11]http://element14.com/
[12]http://www.rs-components.com/index.html



Raspberry Pi B+ and Raspberry Pi B2

Following a meeting with Eric Schmidt (of Google fame) in 2013, Eben embarked on the design of a new form factor for the Pi. On the 26th of November 2015 the Pi Zero was released. The Pi Zero is a significantly smaller version of a Pi with similar functionality but with a retail cost of $5. On release it sold out (20,000 units) World wide in 24 hours and a free copy was affixed to the cover of the MagPi magazine.

The Raspberry Pi 3 was released in February 2016. The most notable change being the inclusion of on-board WiFi and Bluetooth.

In February 2017 the Raspberry Pi Zero W was announced. This device had the same small form factor of the Pi Zero, but included the WiFi and Bluetooth functionality of the Raspberry Pi 3.

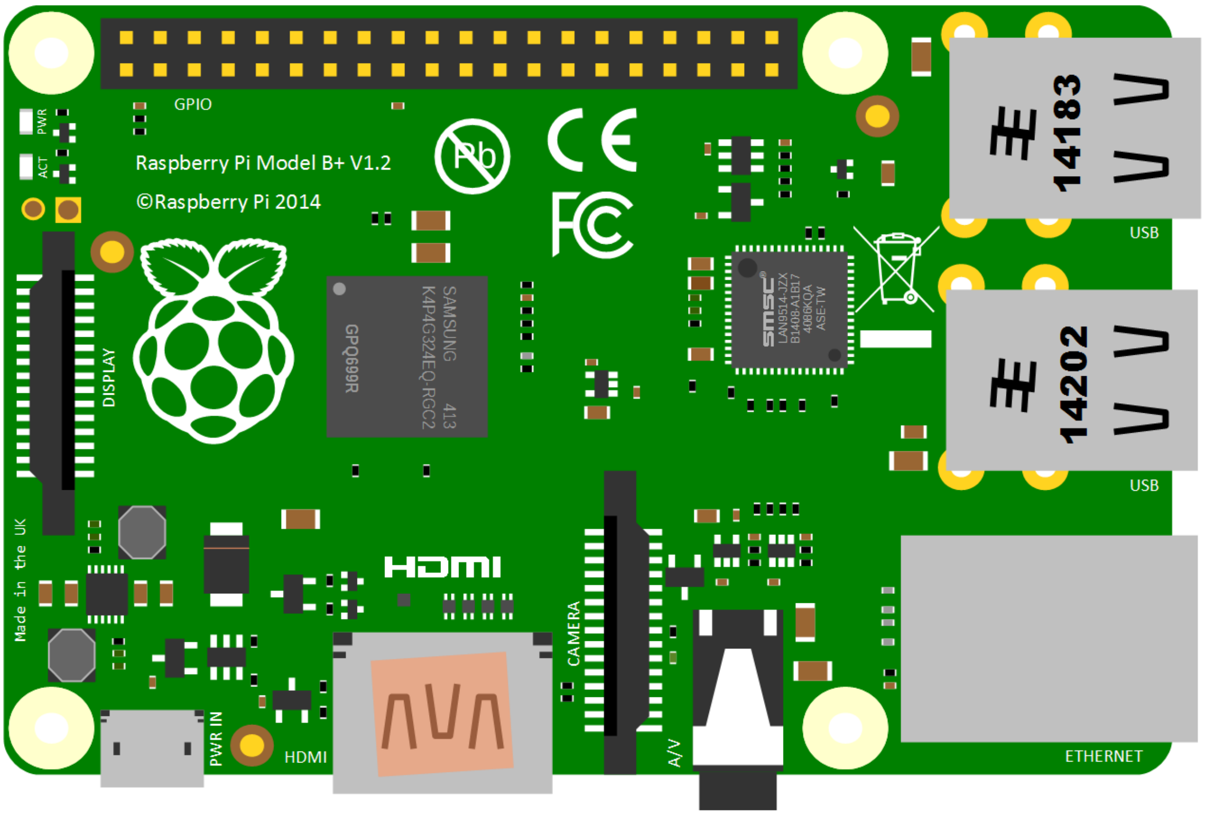
On Pi day (the 14th of March (Get it? 3-14?)) in 2018 the Raspberry Pi 3+ was announced. It included dual band WiFi, upgraded Bluetooth, Gigabit Ethernet and support for a future PoE card. The Ethernet speed was actually 300Mpbs since it still needs to operate on a USB2 bus. By this stage there had been over 9 million Raspberry Pi 3's sold and 19 million Pi's in total.

It would be easy to consider the measurement of the success of the Raspberry Pi in the number of computer boards sold. Yet, this would most likely not be the opinion of those visionaries who began the journey to develop the boards. Their stated aim was to re-invigorate the desire of young people to experiment with computers and to have fun doing it. We can thus measure their success by the many projects, blogs and updated school curriculum's that their efforts have produced.

Raspberry Pi Versions

In the words of the totally awesome Raspberry Pi[13] foundation;

> *The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, playing games and learning how to program in languages like Scratch and Python.*



The Raspberry Pi B+ Board

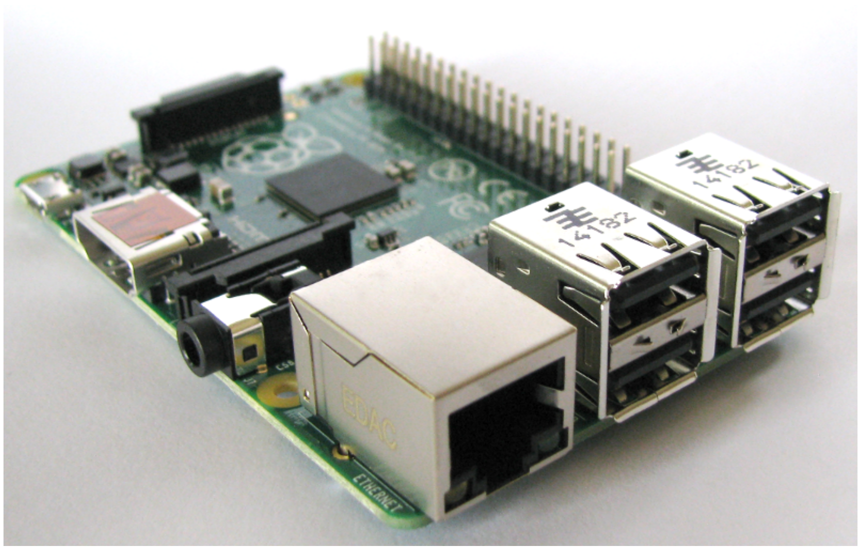
There are (at time of writing) eight different models on the market. The A, B, A+, B+, 'model B 2', 'model B 3', 'model B 3+' (which I'm just going to call the B2, B3 and B3+ respectively), the Zero and Zero W. A lot of projects will typically use either the the B2, B3 or the B3+ for no reason other than they offer a good range of USB ports (4), 1024 MB of RAM, an HMDI video connection and an Ethernet connection. For all intents and purposes either the B2, B3 or B3+ can be used interchangeably for the projects depending on connectivity requirements as the B3 and B3+ has WiFi and Bluetooth built in. For size limited situations or where lower power is an advantage, the Zero or Zero W is useful, although there is a need to cope with reduced connectivity options

[13]http://www.raspberrypi.org/help/what-is-a-raspberry-pi/

(a single micro USB connection) although the Zero W has WiFi and Bluetooth built in. Always aim to use the latest version of the Raspbian operating system (or at least one released on or after the 14th of March 2018). For best results browse the 'Downloads[14]' page of raspberrypi.org.

[14]https://www.raspberrypi.org/downloads/

Raspberry Pi B+, B2, B3 and B3+

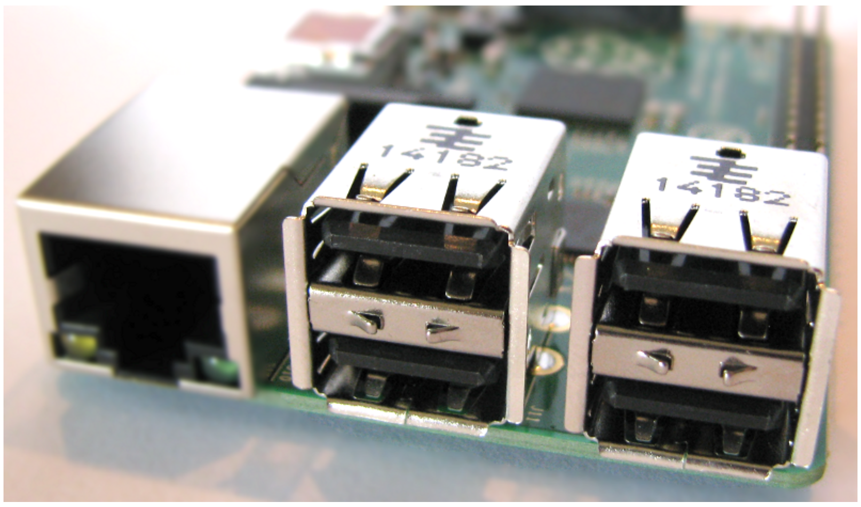


Raspberry Pi B models

The model B+, B2, B3 and B3+ all share the same form factor and have been a consistent standard for the layout of connectors since the release of the B+ in July 2014. They measure 85 x 56 x 17mm, weighs 45g and are powered by Broadcom chipsets of varying speeds, numbers of cores and architectures.

USB Ports

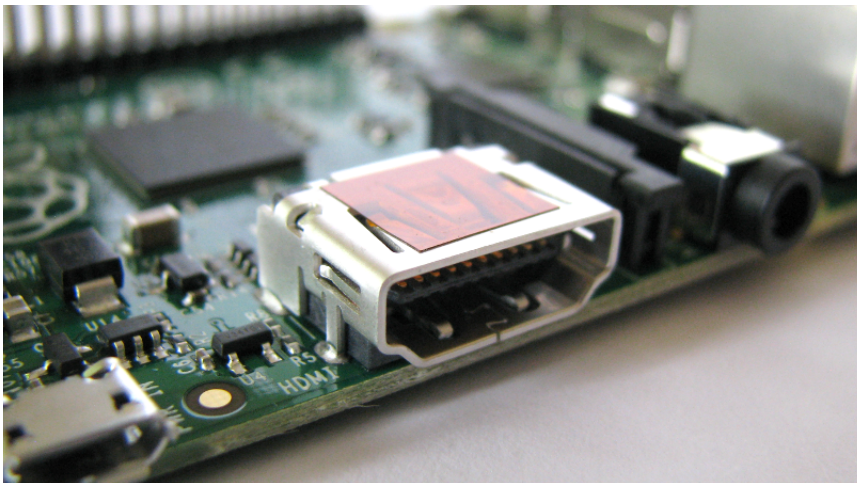
They include 4 x USB Ports (with a maximum output of 1.2A)



Raspberry Pi B+ USB Ports

Video Out

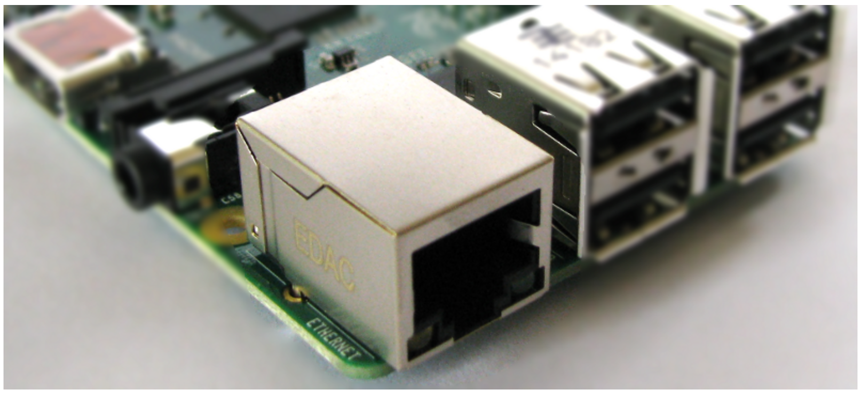
Integrated Videocore 4 graphics GPU capable of playing full 1080p HD video via a HDMI video output connector. HDMI standards rev 1.3 & 1.4 are supported with 14 HDMI resolutions from 640×350 to 1920×1200 plus various PAL and NTSC standards.



Raspberry Pi B Models HDMI Video Output

Ethernet Network Connection

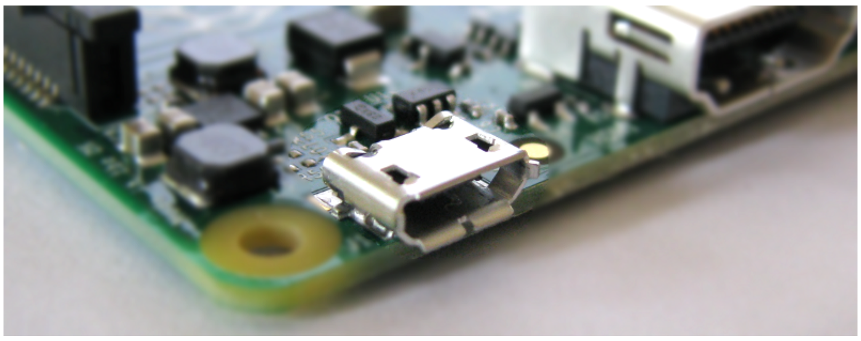
There is an integrated Ethernet Port for network access. On the B2 and B3 the connection speed is fast ethernet (10/100 bps). The B3+ introduced a 300bps connection speed.



Raspberry Pi Model B Ethernet Connector

USB Power Input Jack

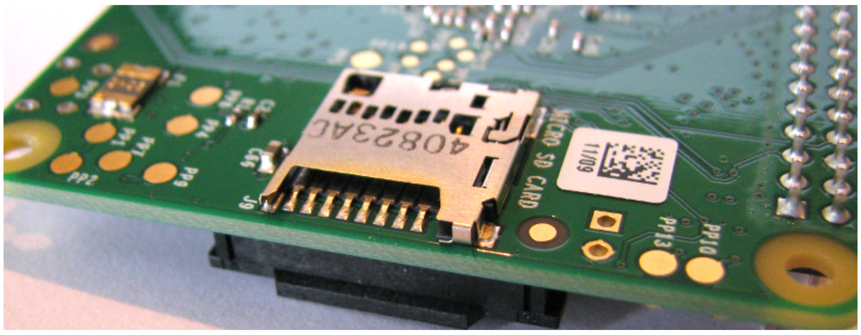
The boards include a 5V 2A Micro USB Power Input Jack.



Raspberry Pi Model B+ USB Power Input

MicroSD Flash Memory Card Slot

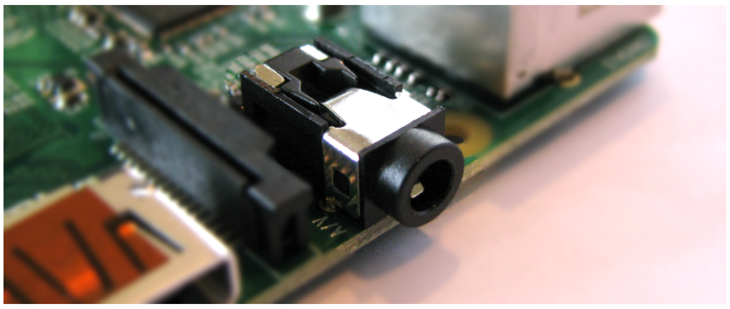
There is a microSD card socket on the ‘underside ‘of the board. On the Model B2 this is a ‘push-push’ socket. On the B3 and later this is a simple friction fit.



Raspberry Pi B+ MicroSD Card Socket

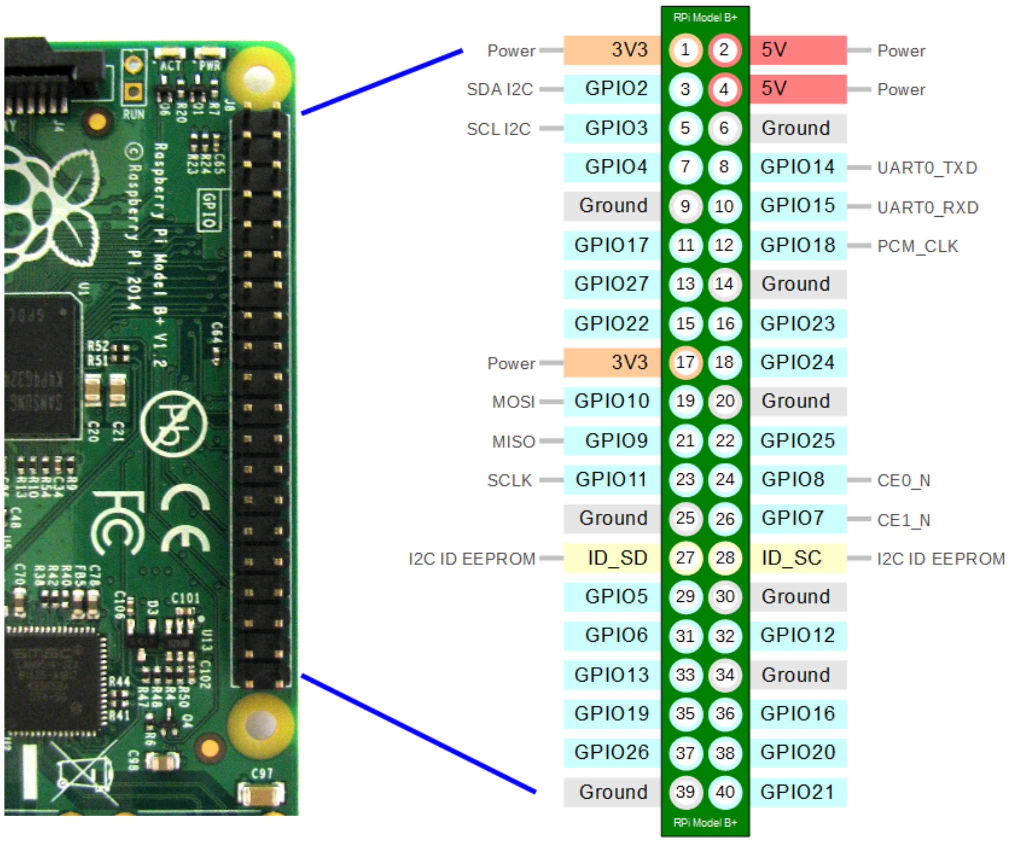
Stereo and Composite Video Output

The B+, B2, B3 and B3+ includes a 4-pole (TRRS[15]) type connector that can provide stereo sound if you plug in a standard headphone jack and composite video output with stereo audio if you use a TRRS adapter.


Raspberry Pi B+ A/V Connector

40 Pin Header

The Raspberry Pi B+, B2, B3 and B3+ include a 40-pin, 2.54mm header expansion slot (Which allows for peripheral connection and expansion boards).



Raspberry Pi B+ GPIO Connector

[15]http://www.cablechick.com.au/blog/understanding-trrs-and-audio-jacks/

Raspberry Pi Peripherals

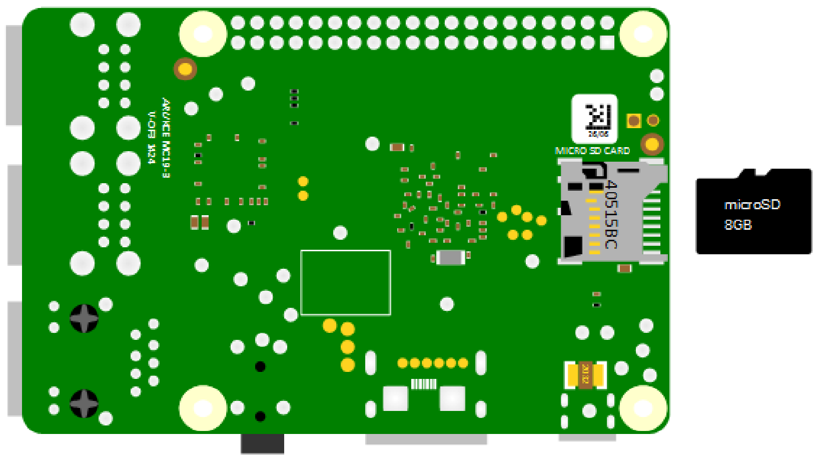
To make a start using the Raspberry Pi we will need to have some additional hardware to allow us to configure it.

SD Card

Traditionally the Raspberry Pi needs to store the Operating System and working files on a MicroSD card (actually a MicroSD card all models except the older A or B models which use a full size SD card). There is the ability to boot from a mass storage device or the network, but it is slightly 'non-trivial', so we won't cover it.

MicroSD Card

The MicroSD card receptacle is on the rear of the board and on the Model B2 it is a 'push-push' type which means that you push the card in to insert it and then to remove it, give it a small push and it will spring out.

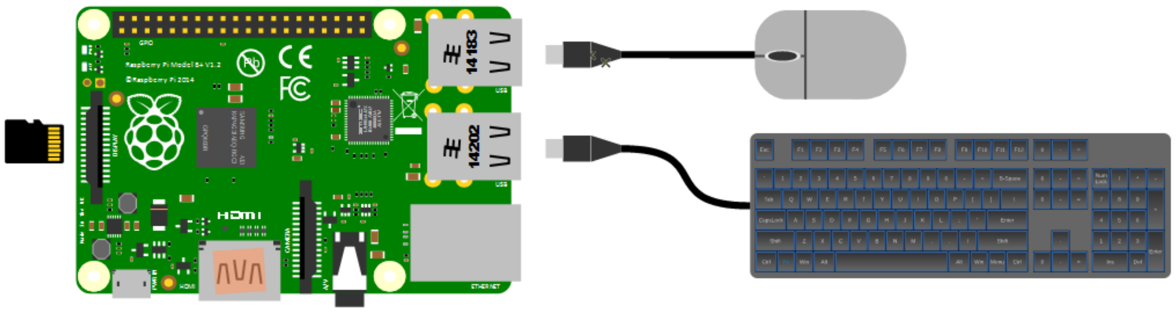


MicroSD Card Positioning

This is the equivalent of a hard drive for a regular computer, but we're going for a minimal effect. We will want to use a minimum of an 8GB card (smaller is possible, but 8 is recommended). Also try to select a higher speed card if possible (class 10 or similar) as this will speed things up a bit.

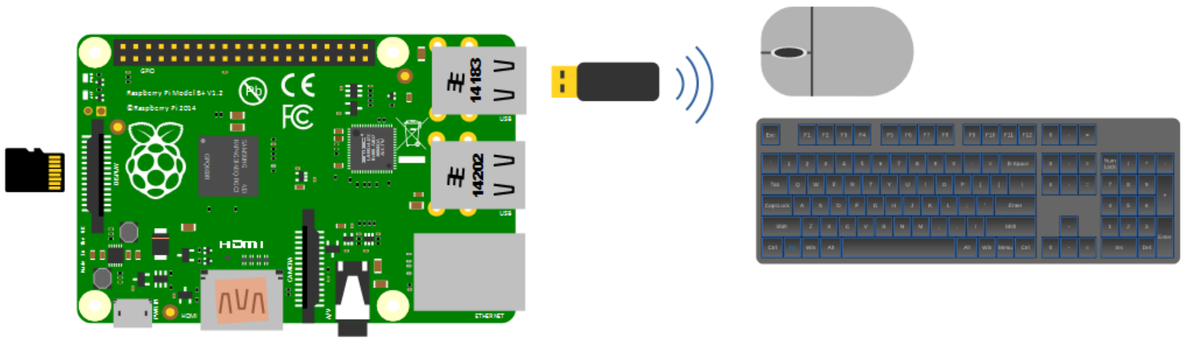
Keyboard / Mouse

While we will be making the effort to access our system via a remote computer, we will need a keyboard and a mouse for the initial set-up. Because the B+, B2, B3 and B3+ models of the Pi have 4 x USB ports, there is plenty of space for us to connect wired USB devices.



Wired Keyboard and Mouse

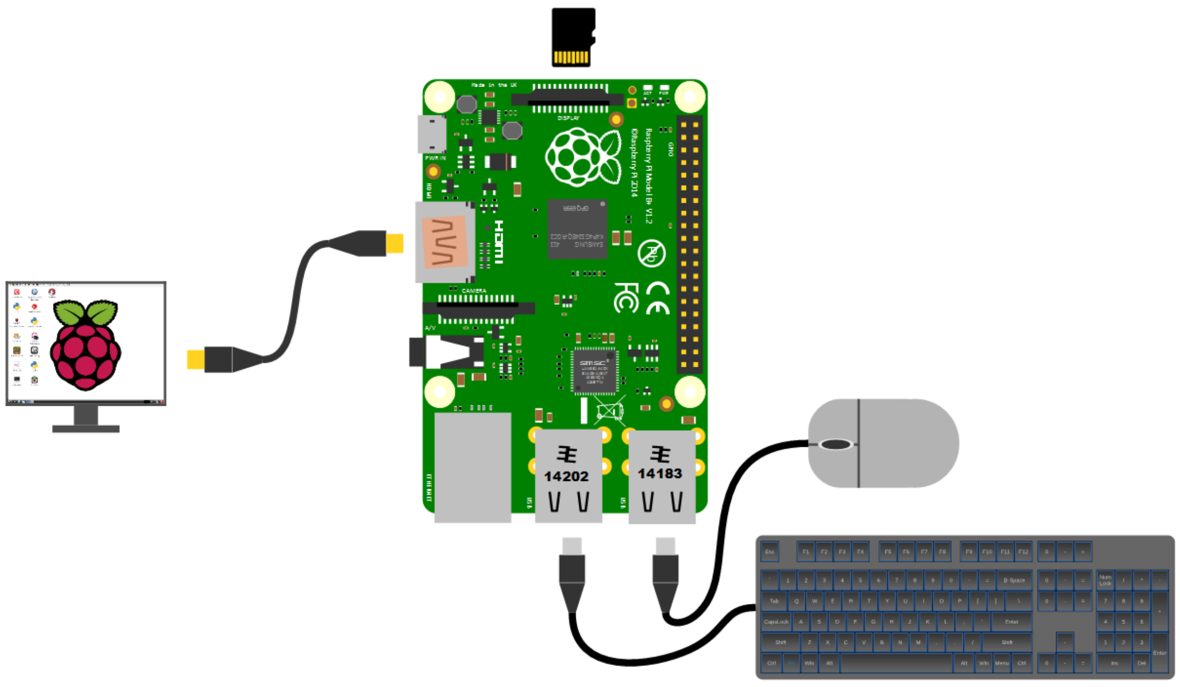
An external wireless combination would most likely be recognised without any problem and would only take up a single USB port, but if we build towards a remote capacity for using the Pi (using it headless, without a keyboard / mouse / display), the nicety of a wireless connection is not strictly required.



Wireless Keyboard and Mouse

Video

The Raspberry Pi comes with an HDMI port ready to go which means that any monitor or TV with an HDMI connection should be able to connect easily.



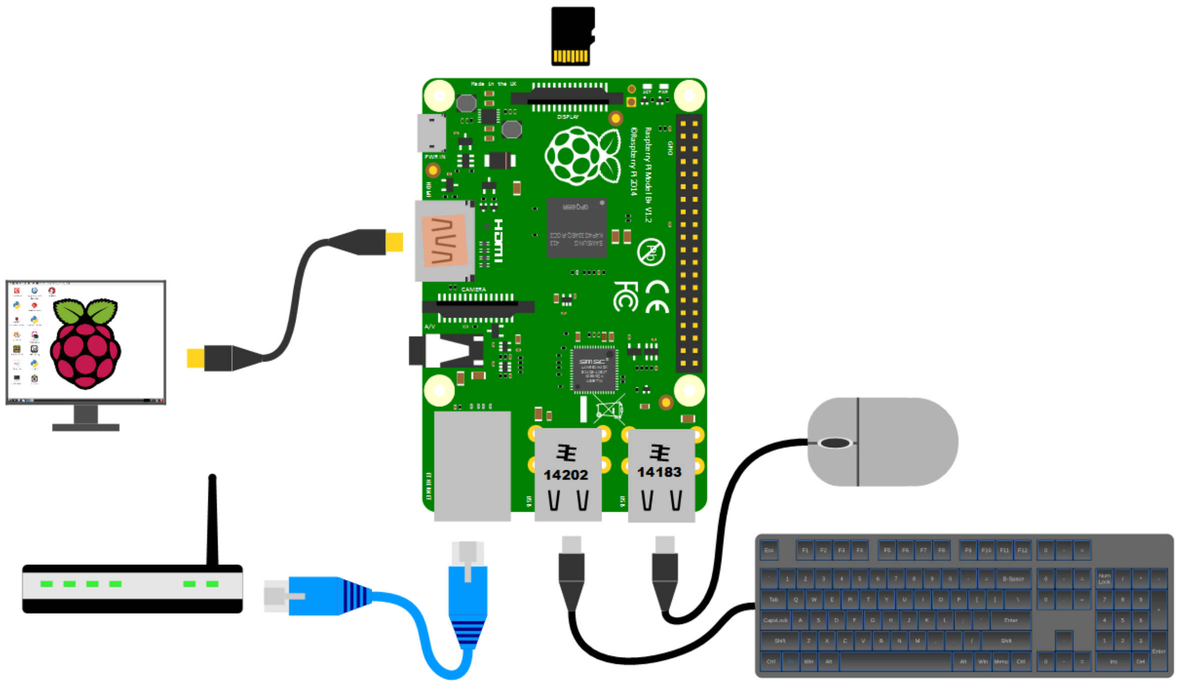
HDMI Connected Monitor

Because this is kind of a hobby thing you might want to consider utilising an older computer monitor with a DVI or 15 pin ‘D’ connector. If you want to go this way you will need an adapter to convert the connection.

VGA to HDMI Adapter

Network

The B+, B2, B3 and B3+ models of the Raspberry Pi have a standard RJ45 network connector on the board ready to go. In a domestic installation this is most likely easiest to connect into a home ADSL modem or router.

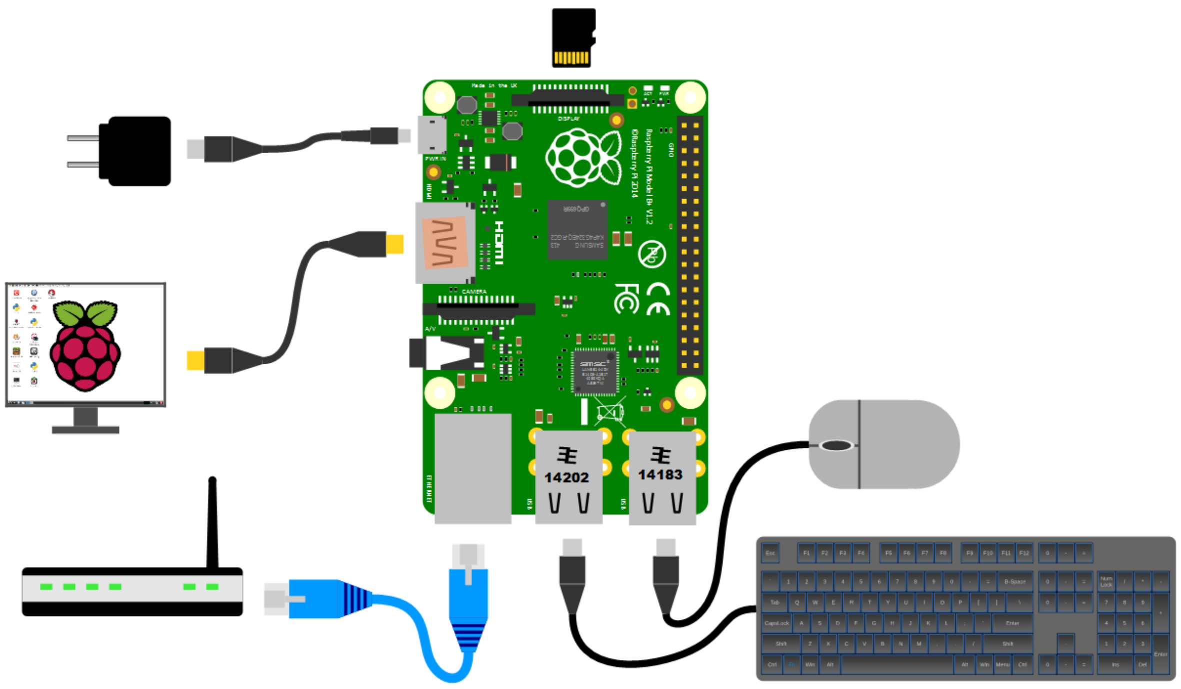


HDMI Connected Monitor

This 'hard-wired' connection is great for getting started, but we will work through using a wireless solution later in the book.

Power supply

The Pi can be powered up in a few ways. The simplest is to use the micro USB port to connect from a standard USB charging cable. You probably have a few around the house already for phones or tablets.



Power Supply Connection

However, it's worth thinking about the application that we use our Pi for. Depending on how much we ask of the unit, we might want to pay attention to the amount of current that our power supply can deliver. The A+, B+ and Zero models will function adequately with a 700mA supply, but the B2, B3 and B3+ models will draw more current and if we want to use multiple wireless devices or supplying sensors that demand increased power, we will need to consider a supply that is capable of an output up to 2.5A.

Cases

We should get ourselves a simple case to keep the Pi reasonably secure. There are a wide range of options to select from. These range from cheap but effective to more costly than the Pi itself (not hard) and looking fancy.

You could use a simple plastic case[16] that can be brought for a few dollars;

Simple ABS plastic case

For a very practical design and a warm glow from knowing that you're supporting a worthy cause, you could go no further than the official Raspberry Pi case[17] that includes removable side-plates and loads of different types of access. All for the paltry sum of about $9.

Official Raspberry Pi case

[16]http://www.dx.com/p/abs-case-box-for-raspberry-pi-b-black-346332
[17]https://www.raspberrypi.org/blog/raspberry-pi-official-case/

Operating Systems

An operating system is software that manages computer hardware and software resources for computer applications. For example Microsoft Windows could be the operating system that will allow the browser application Firefox to run on our desktop computer.

Variations on the Linux operating system are the most popular on our Raspberry Pi. Often they are designed to work in different ways depending on the function of the computer.

Linux[18] is a computer operating system that is can be distributed as free and open-source software[19]. The defining component of Linux is the Linux kernel, an operating system kernel first released on 5 October 1991 by Linus Torvalds.

Linux was originally developed as a free operating system for Intel x86-based personal computers. It has since been made available to a huge range of computer hardware platforms and is one of the most popular operating systems on servers, mainframe computers and supercomputers. Linux also runs on embedded systems, which are devices whose operating system is typically built into the firmware and is highly tailored to the system; this includes mobile phones, tablet computers, network routers, facility automation controls, televisions and video game consoles. Android, the most widely used operating system for tablets and smart-phones, is built on top of the Linux kernel. In our case we will be using a version of Linux that is assembled to run on the ARM CPU architecture used in the Raspberry Pi.

The development of Linux is one of the most prominent examples of free and open-source software collaboration. Typically, Linux is packaged in a form known as a Linux ‘distribution’, for both desktop and server use. Popular mainstream Linux distributions include Debian, Ubuntu and the commercial Red Hat Enterprise Linux. Linux distributions include the Linux kernel, supporting utilities and libraries and usually a large amount of application software to carry out the distribution’s intended use.

A distribution intended to run as a server may omit all graphical desktop environments from the standard install, and instead include other software to set up and operate a solution stack such as LAMP (Linux, Apache, MySQL and PHP). Because Linux is freely re-distributable, anyone may create a distribution for any intended use.

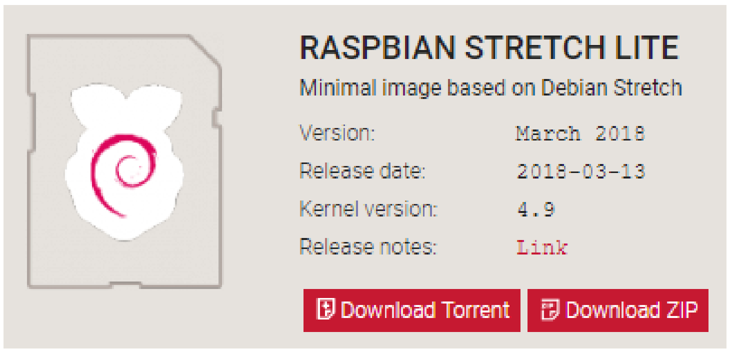
Welcome to Raspbian

The Raspbian Linux distribution is based on Debian Linux. At the time of writing there have been three different editions published. ‘Wheezy’, ‘Jessie’ and ‘Stretch’. Debian is a widely used Linux distribution that allows Raspbian users to leverage a huge quantity of community based experience in using and configuring software. The Wheezy edition is the earlier of the three and was the stock edition from the inception of the Raspberry Pi till the end of 2015. From that point Jessie was the default distribution until mid 2017 when Stretch took over.

[18]http://en.wikipedia.org/wiki/Linux
[19]http://en.wikipedia.org/wiki/Free_and_open-source_software

Downloading

The best place to source the latest version of the Raspbian Operating System is to go to the raspberrypi.org page; http://www.raspberrypi.org/downloads/. We will download the 'Lite' version (which doesn't use a desktop GUI). If you've never used a command line environment, then good news! You're about to enter the World of '*real*' computer users :-).


RASPBIAN STRETCH LITE

Minimal image based on Debian Stretch

Version:	March 2018
Release date:	2018-03-13
Kernel version:	4.9
Release notes:	Link

Raspbian Download

You can download via bit torrent or directly as a zip file, but whatever the method you should eventually be left with an 'img' file for Raspbian.

To ensure that the projects we work on can be used with either the B+, B2 or B3 models we need to make sure that the version of Raspbian we download is from 2015-01-13 or later. Earlier downloads will not support the more modern CPU of the B2 or B3. To support the newer CPU of the B3+ (and all the previous CPUs) we will need a version of Raspbian from 2018-03-13 or later.

Image File

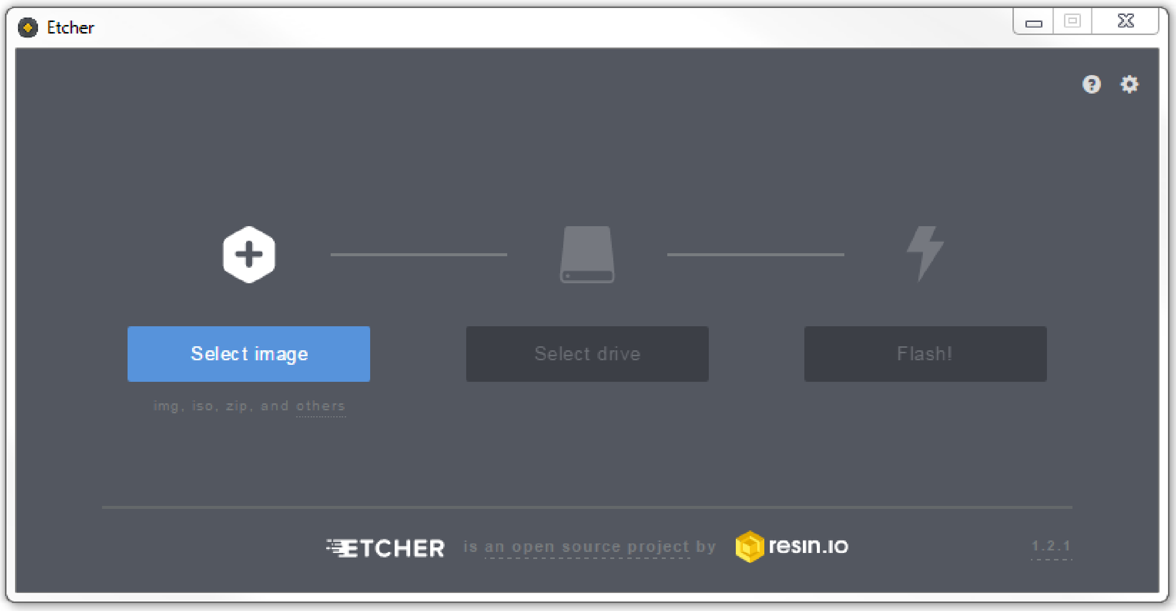
We should always try to download our image files from the authoritative source!

Writing the Operating System image to the SD Card

Once we have an image file we need to get it onto our SD card.

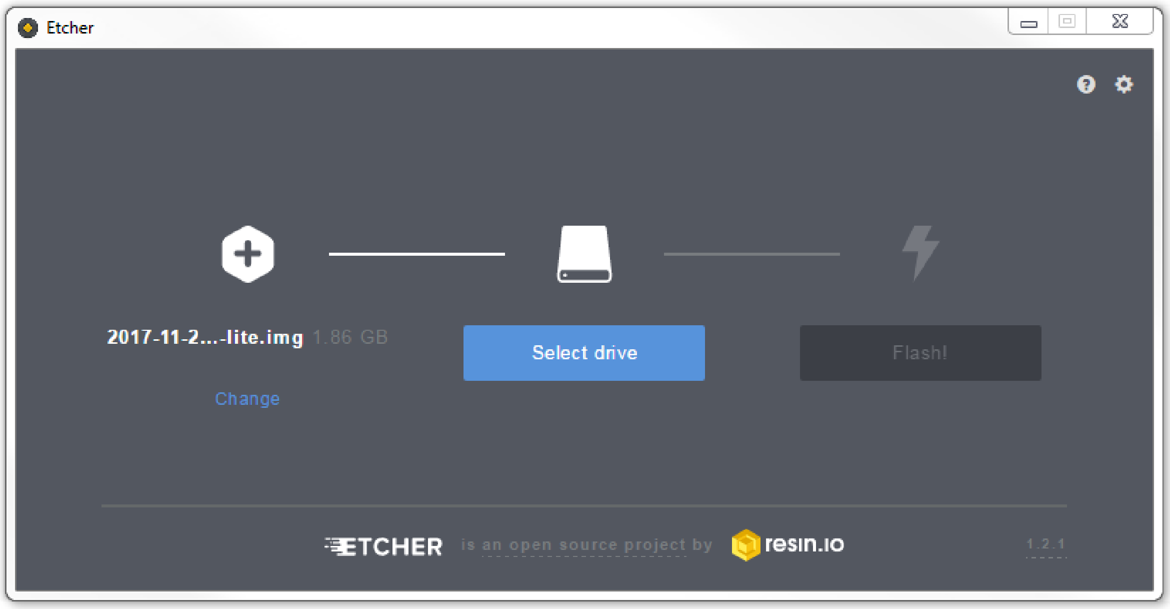
We will work through an example using Windows 7 but the process should be very similar for other operating systems as we will be using the excellent open source software Etcher[20] which is available for Windows, Linux and macOS.

Download and install Etcher and start it up.


Etcher
Select image
img, iso, zip, and others
Select drive
Flash!
ETCHER is an open source project by resin.io
1.2.1

Etcher Start

Select the img file that you want to install.

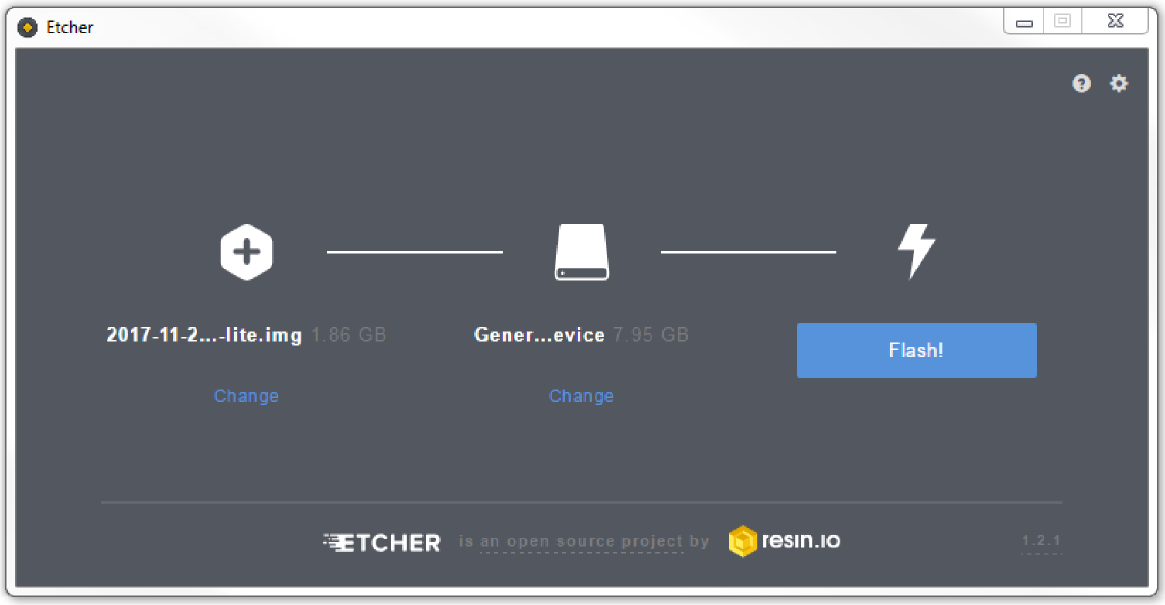

Etcher
2017-11-2...-lite.img 1.86 GB
Change
Select drive
Flash!
ETCHER is an open source project by resin.io
1.2.1

Etcher SD Card Selection

You will need an SD card reader capable of accepting your MicroSD card (you may require an

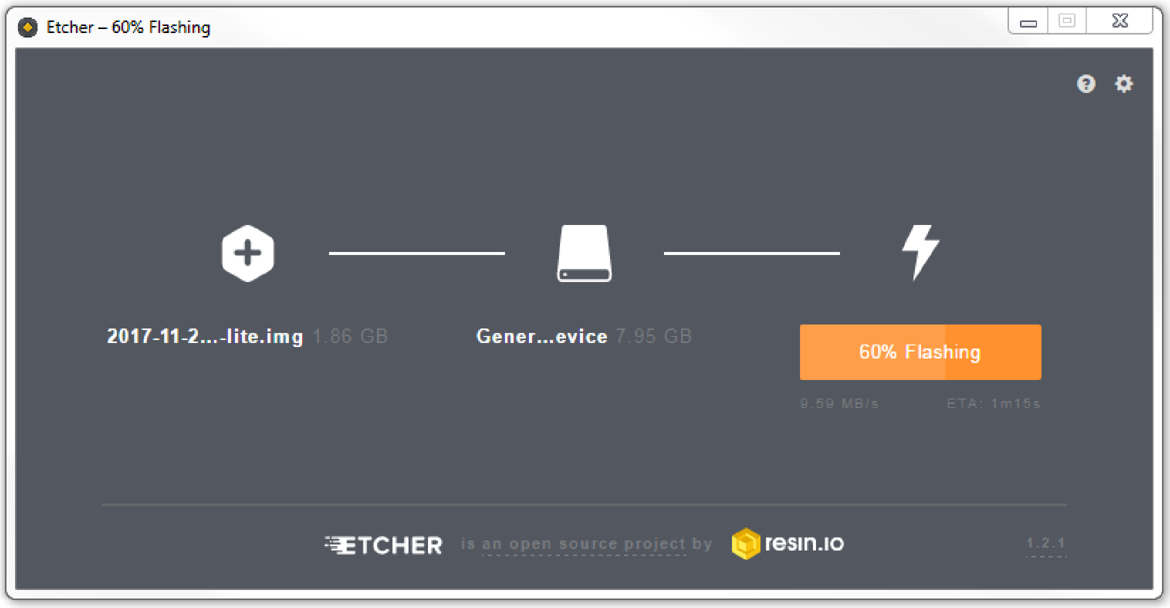
[20]https://etcher.io/

adapter or have a reader built into your desktop or laptop). Place the card in the reader and you should see Etcher automatically select it for writing (Etcher is very good at presenting options for installing that are **only** SD cards).


Etcher
2017-11-2...-lite.img 1.86 GB
Change
Gener...evice 7.95 GB
Change
Flash!
ETCHER is an open source project by resin.io
1.2.1

Flash the drive

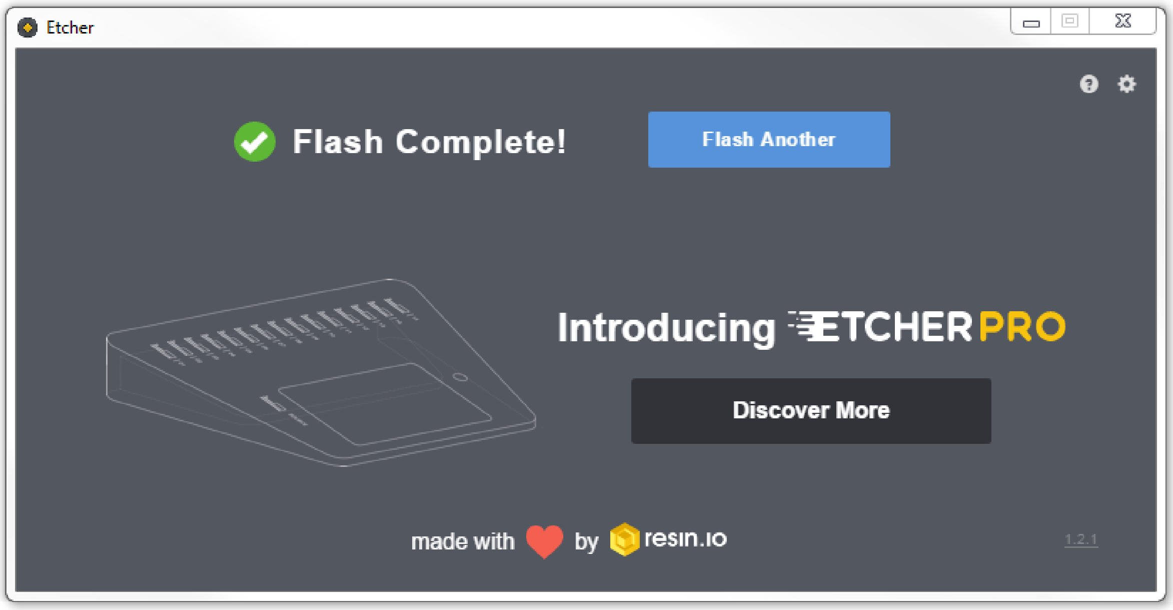
Then click on 'Flash!' to burn the card.



Etcher in progress

Etcher will write the image to the SD card. The time taken can vary a little, but it should only take about 3-4 minutes with a class 10 SD card.

Once written, Etcher will validate the write process (this can be disabled if desired).



Flash Complete!

When the process is finished Etcher will automatically unmount the SD card.

Enabling Secure Shell Access

One of the awesome things when learning to use a Raspberry Pi comes when you begin to access it remotely from another computer. This is a bit of an 'Ah Ha!' moment for some people as they begin to appreciate just how networks and the Internet is built. We are going to enable and use remote access via what is called 'SSH'. We'll start using it later in the book, but for now we can take the opportunity to enable it for later use. We do this by creating a file called 'ssh' on our freshly written SD card. Then, when the Pi then boots up it sees the file and automatically knows to enable SSH.

`SSH` *used* to be enabled by default, but doing so presents a potential security concern, so it has been disabled by default as of the end of 2016. In our case it's a feature that we want to use.

Eject the card from the computer and then re-insert it. When the computer recognises the card, open it and right-click in the folder to create a new file. This can be a simple txt file so long as the file prefix is 'ssh'. It doesn't need to have anything in it, there just needs to be a file there.

Now we can unmount the SD card and eject it again.

Powering On

Insert the card into the slot on the Raspberry Pi and turn on the power.

You will see a range of information scrolling up the screen before eventually being presented with a login prompt.

The Command Line interface

Because we have installed the 'Lite' version of Raspbian, when we first boot up, the process should automatically re-size the root file system to make full use of the space available on your SD card. If this isn't the case, the facility to do it can be accessed from the Raspberry Pi configuration tool (raspi-config) that we will look at in a moment.

Once the reboot is complete (if it occurs) you will be presented with the console prompt to log on;

```
Raspbian GNU/Linux 7 raspberrypi tty1

raspberrypi login:
```

The default username and password is:

Username: pi

Password: raspberry

Enter the username and password.

Congratulations, you have a working Raspberry Pi and are ready to start getting into the thick of things!

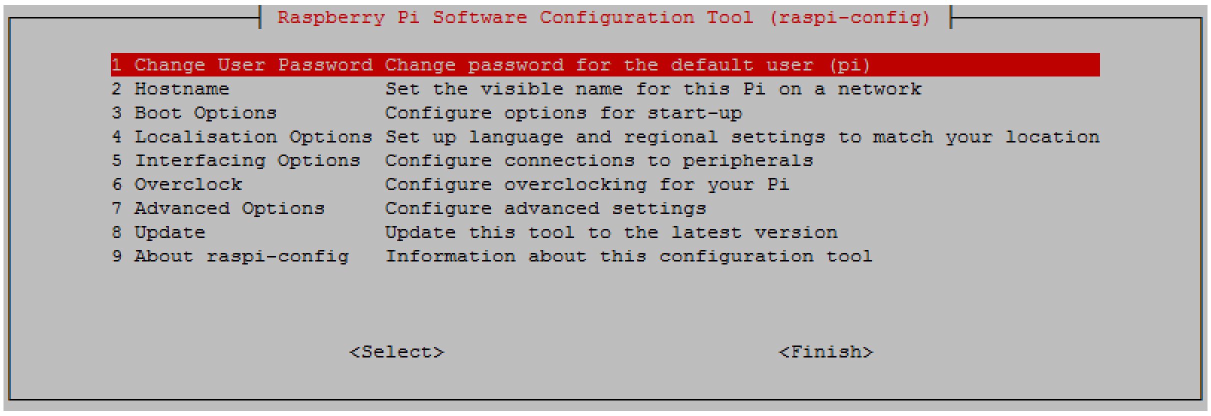
Firstly we'll do a bit of house keeping.

Raspberry Pi Software Configuration Tool

We will use the Raspberry Pi Software Configuration Tool to change the locale and keyboard configuration to suit us. This can be done by running the following command;

```
sudo raspi-config
```

The `sudo` portion of the command makes sure that you will have the permission required to run the `apt-get` process.

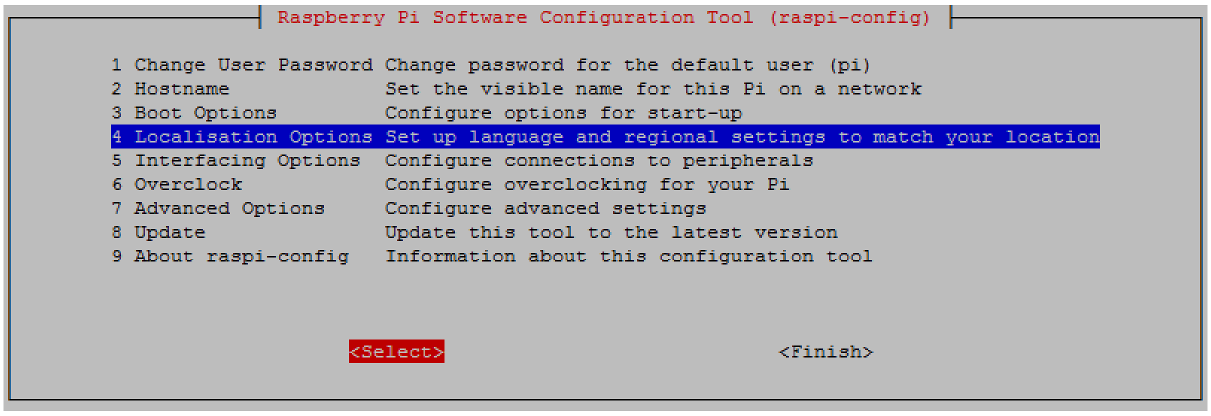


```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the default user (pi)
2 Hostname             Set the visible name for this Pi on a network
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 peripherals
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>
```

Raspberry Pi Software Configuration Tool

Use the up and down arrow keys to move the highlighted section to the selection you want to make then press tab to highlight the `<Select>` option (or `<Finish>` if you've finished).

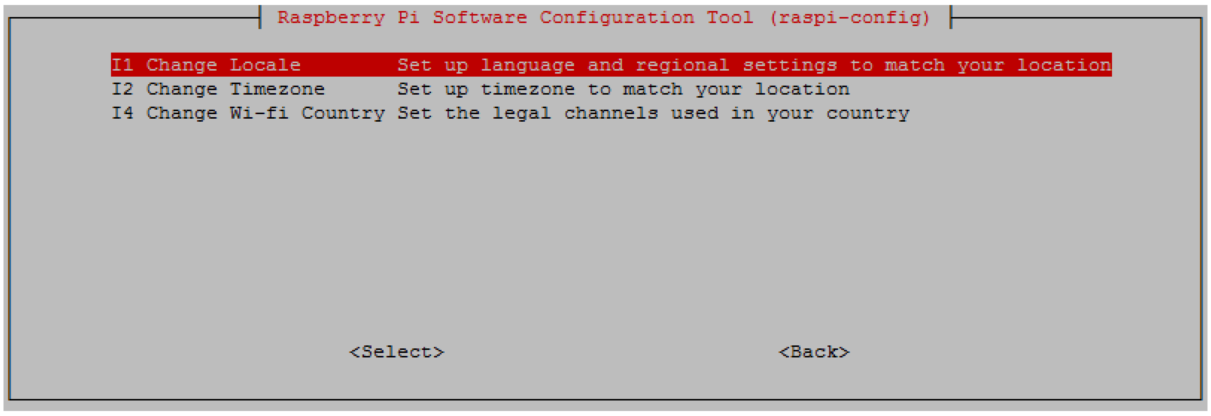
Lets change the settings for our operating system to reflect our location for the purposes of having the correct time, language and WiFi regulations. These can all be located via selection '4 Localisation Options' on the main menu.



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the default user (pi)
2 Hostname             Set the visible name for this Pi on a network
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 peripherals
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>
```

Select Localisation Options

Select this and work through any changes that are required for your installation based on geography.



```
Raspberry Pi Software Configuration Tool (raspi-config)

I1 Change Locale       Set up language and regional settings to match your location
I2 Change Timezone     Set up timezone to match your location
I4 Change Wi-fi Country Set the legal channels used in your country

                  <Select>                          <Back>
```

Localisation Options

Once you exit out of the `raspi-config` menu system, if you have made a few changes, there is a probability that you will be asked if you want to re-boot the Pi. That's a pretty good idea.

Once the reboot is complete you will be presented with the console prompt to log on again;

Software Updates

After configuring our Pi we'll want to make sure that we have the latest software for our system. This is a useful thing to do as it allows any additional improvements to the software we will be using to be enhanced or security of the operating system to be improved. This is probably a good time to mention that we will need to have an Internet connection available.

Type in the following line which will find the latest lists of available software;

```
sudo apt-get update
```

You should see a list of text scroll up while the Pi is downloading the latest information.

Then we want to upgrade our software to latest versions from those lists using;

```
sudo apt-get upgrade
```

The Pi should tell you the lists of packages that it has identified as suitable for an upgrade along with the amount of data that will be downloaded and the space that will be used on the system. It will then ask you to confirm that you want to go ahead. Tell it 'Y' and we will see another list of details as it heads off downloading software and installing it.

Power Up the Pi

To configure the Raspberry Pi for our purpose we will extend our Pi a little. This makes configuring and using the device easier and to be perfectly honest, making life hard for ourselves is so *exhausting!* Let's not do that.

Static IP Address

As we mentioned earlier, enabling remote access is a really useful thing. This will allow us to configure and operate our raspberry Pi from a separate computer. To do so we will want to assign our Raspberry Pi a static IP address.

An Internet Protocol address (IP address) is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication.

There is a strong likelihood that our Raspberry Pi already has an IP address and it should appear a few lines above the 'login' prompt when you first boot up;

```
My IP address is 10.1.1.25

Raspbian GNU/Linux 7 raspberrypi tty1

raspberrypi login:
```

The `My IP address...` part should appear just above or around 15 lines above the login line, depending on the version of Raspbian we're using. In this example the IP address 10.1.1.25 belongs to the Raspberry Pi.

This address will *probably* be a 'dynamic' IP address and could change each time the Pi is booted. For the purposes of using the Raspberry Pi with a degree of certainty when logging in to it remotely it's easier to set a fixed IP address.

This description of setting up a static IP address makes the assumption that we have a device running on our network that is assigning IP addresses as required. This sounds complicated, but in fact it is a very common service to be running on even a small home network and most likely on an ADSL modem/router or similar. This function is run as a service called DHCP[21] (**D**ynamic **H**ost **C**onfiguration **P**rotocol). You will need to have access to this device for the purposes of knowing what the allowable ranges are for a static IP address.

[21]http://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol

The Netmask

A common feature for home modems and routers that run DHCP devices is to allow the user to set up the range of allowable network addresses that can exist on the network. At a higher level we should be able to set a 'netmask' which will do the job for us. A netmask looks similar to an IP address, but it allows you to specify the range of addresses for 'hosts' (in our case computers) that can be connected to the network.

A very common netmask is 255.255.255.0 which means that the network in question can have any one of the combinations where the final number in the IP address varies. In other words with a netmask of 255.255.255.0, the IP addresses available for devices on the network '10.1.1.x' range from 10.1.1.0 to 10.1.1.255 or in other words any one of 256 unique addresses.

CIDR Notation

An alternative to specifying a netmask in the format of '255.255.255.0' is to use a system called **C**lassless **I**nter-**D**omain **R**outing, or CIDR. The idea is to add a specification in the IP address itself that indicates the number of significant bits that make up the netmask.

For example, we could designate the IP address 10.1.1.17 as associated with the netmask 255.255.255.0 by using the CIDR notation of 10.1.1.17/24. This means that the first 24 bits of the IP address given are considered significant for the network routing.

Using CIDR notation allows us to do some very clever things to organise our network, but at the same time it can have the effect of confusing people by introducing a pretty complex topic when all they want to do is get their network going :-). So for the sake of this explanation we can assume that if we wanted to specify an IP address and a netmask, it could be accomplished by either specifying each separately (IP address = 10.1.1.17 and netmask = 255.255.255.0) or in CIDR format (10.1.1.1/24)

Distinguish Dynamic from Static

The other service that our DHCP server will allow is the setting of a range of addresses that can be assigned dynamically. In other words we will be able to declare that the range from 10.1.1.20 to 10.1.1.255 can be dynamically assigned which leaves 10.1.1.0 to 10.1.1.19 which can be set as static addresses.

You might also be able to reserve an IP address on your modem / router. To do this you will need to know what the MAC (or hardware address) of the Raspberry Pi is. To find the hardware address on the Raspberry Pi type;

```
ifconfig -a
```

(For more information on the `ifconfig` command check out the Linux commands section)

This will produce an output which will look a little like the following;

```
eth0 Link encap:Ethernet HWaddr 00:08:C7:1B:8C:02
     inet addr:10.1.1.26 Bcast:10.1.1.255 Mask:255.255.255.0
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:53 errors:0 dropped:0 overruns:0 frame:0
     TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:4911 (4.7 KiB)  TX bytes:4792 (4.6 KiB)
```

The figures `00:08:C7:1B:8C:02` are the Hardware or MAC address.

Because there are a huge range of different DHCP servers being run on different home networks, I will have to leave you with those descriptions and the advice to consult your devices manual to help you find an IP address that can be assigned as a static address. Make sure that the assigned number has not already been taken by another device. In a perfect World we would hold a list of any devices which have static addresses so that our Pi's address does not clash with any other device.

Be aware that if you don't have a section of your IP address range set aside for static addresses you run the risk of having the DHCP service unwittingly assign a device that *wants* a dynamic address with the same value that you have already assigned for your Raspberry Pi. Such a conflict is not a good thing.

For the sake of the upcoming projects we will assume that the address 10.1.1.120 is available.

Default Gateway

Before we start configuring we will need to find out what the default gateway is for our network. A default gateway is an IP address that a device (typically a router) will use when it is asked to go to an address that it doesn't immediately recognise. This would most commonly occur when a computer on a home network wants to contact a computer on the Internet. The default gateway is therefore typically the address of the modem / router on your home network.

We can check to find out what our default gateway is from Windows by going to the command prompt (Start > Accessories > Command Prompt) and typing;

```
ipconfig
```

This should present a range of information including a section that looks a little like the following;

```
Ethernet adapter Local Area Connection:

  IPv4 Address. . . . . . . . . . . : 10.1.1.15
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . : 10.1.1.1
```

The default router gateway is therefore '10.1.1.1'.

Lets edit the `dhcpcd.conf` file

On the Raspberry Pi at the command line we are going to start up a text editor and edit the file that holds the configuration details for the network connections.

The file is `/etc/dhcpcd.conf`. That is to say it's the `dhcpcd.conf` file which is in the `etc` directory which is in the root (`/`) directory.

To edit this file we are going to type in the following command;

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

Remember, the `sudo` portion of the command makes sure that you will have the permission required to edit the `dhcpcd.conf` file, `nano` is the name of the text editor and `/etc/dhcpcd.conf` is telling the computer which file to edit.

The nano[22] file editor will start and show the contents of the `dhcpcd.conf` file which should look a little like the following;

```
 A sample configuration for dhcpcd.
# See dhcpcd.conf(5) for details.

# Allow users of this group to interact with dhcpcd via the control socket.
#controlgroup wheel

# Inform the DHCP server of our hostname for DDNS.
hostname

# Use the hardware address of the interface for the Client ID.
clientid
# or
# Use the same DUID + IAID as set in DHCPv6 for DHCPv4 ClientID per RFC4361.
#duid
```

[22]http://www.nano-editor.org/

```
# Persist interface configuration when dhcpcd exits.
persistent

# Rapid commit support.
# Safe to enable by default because it requires the equivalent option set
# on the server to actually work.
option rapid_commit

# A list of options to request from the DHCP server.
option domain_name_servers, domain_name, domain_search, host_name
option classless_static_routes
# Most distributions have NTP support.
option ntp_servers
# Respect the network MTU. This is applied to DHCP routes.
option interface_mtu

# A ServerID is required by RFC2131.
require dhcp_server_identifier

# Generate Stable Private IPv6 Addresses instead of hardware based ones
slaac private

# Example static IP configuration:
#interface eth0
#static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1

# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0
```

The file actually contains some commented out sections that provide guidance on entering the correct configuration.

We are going to add the information that tells the network interface to use eth0 at our static address that we decided on earlier (`10.1.1.120`) along with information on the netmask to use (in CIDR format) and the default gateway of our router. To do this we will add the following lines to the end of the information in the `dhcpcd.conf` file;

```
# Custom static IP address for eth0.
interface eth0
static ip_address=10.1.1.120/24
static routers=10.1.1.1
static domain_name_servers=10.1.1.1
```

Here we can see the IP address and netmask (`static ip_address=10.1.1.120/24`), the gateway address for our router (`static routers=10.1.1.1`) and the address where the computer can also find DNS information (`static domain_name_servers=10.1.1.1`).

In a simplistic explanation, the **D**omain **N**ame **S**ystem (DNS) makes sure that the Internet can find resources easily based on a naming convention.

Once you have finished press ctrl-x to tell nano you're finished and it will prompt you to confirm saving the file. Check your changes over and then press 'y' to save the file (if it's correct). It will then prompt you for the file-name to save the file as. Press return to accept the default of the current name and you're done!

To allow the changes to become operative we can type in;

```
sudo reboot
```

This will reboot the Raspberry Pi and we should see the (by now familiar) scroll of text and when it finishes rebooting you should see;

```
My IP address is 10.1.1.120

Raspbian GNU/Linux 7 raspberrypi tty1

raspberrypi login:
```

Which tells us that the changes have been successful (bearing in mind that the IP address above should be the one *you* have chosen, not necessarily the one we have been using as an example).

Remote access

To allow us to work on our Raspberry Pi from our normal desktop we can give ourselves the ability to connect to the Pi from another computer. The will mean that we don't need to have the keyboard / mouse or video connected to the Raspberry Pi and we can physically place it somewhere else and still work on it without problem. This process is called 'remotely accessing' our computer .

To do this we need to install an application on our windows desktop which will act as a 'client' in the process and have software on our Raspberry Pi to act as the 'server'. There are a couple of different ways that we can accomplish this task, but because we will be working at the command line (where all we do is type in our commands (like when we first log into the Pi)) we will use what's called SSH access in a 'shell'.

Remote access via SSH

Secure Shell (SSH[23]) is a network protocol that allows secure data communication, remote command-line login, remote command execution, and other secure network services between two networked computers. It connects, via a secure channel over an insecure network, a server and a client running SSH server and SSH client programs, respectively (there's the client-server model again).

In our case the SSH program on the server is running sshd and on the Windows machine we will use a program called 'PuTTY'.

Setting up the Server (Raspberry Pi)

SSH is already installed and operating but to check that it is there and working type the following from the command line;

```
/etc/init.d/ssh status
```

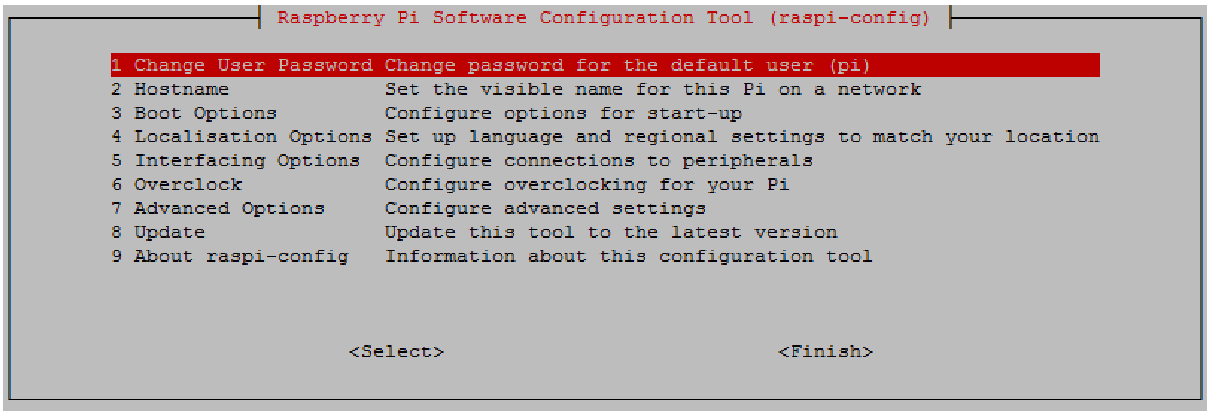
The Pi should respond with the message that the program `sshd` is active (running).

```
pi@raspberrypi:~ $ /etc/init.d/ssh status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
   Active: active (running) since Tue 2017-04-25 03:30:16 UTC; 1h 28min ago
 Main PID: 2135 (sshd)
   CGroup: /system.slice/ssh.service
           └─2135 /usr/sbin/sshd -D
```

If it isn't, run the following command;

[23] http://en.wikipedia.org/wiki/Secure_Shell

```
sudo raspi-config
```

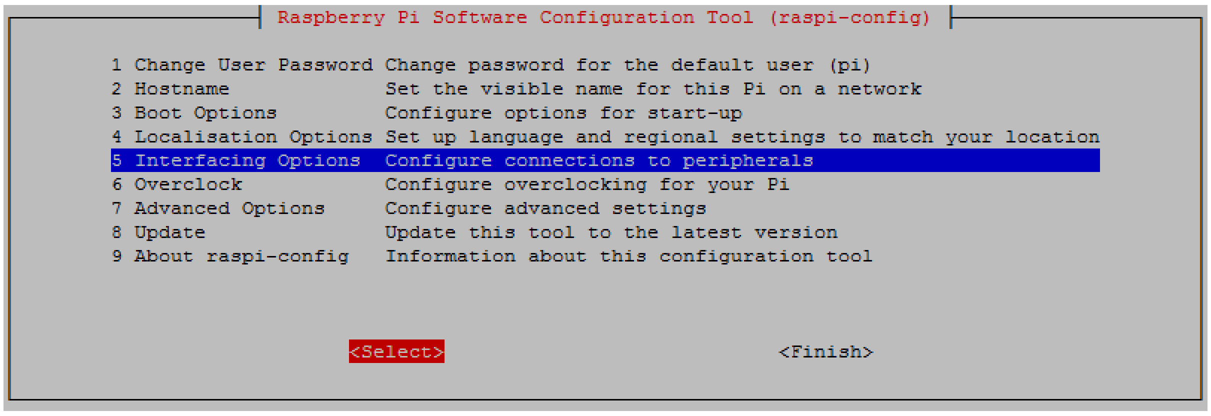


```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the default user (pi)
2 Hostname             Set the visible name for this Pi on a network
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 peripherals
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>
```

Raspberry Pi Software Configuration Tool

Use the up and down arrow keys to move the highlighted section to the selection you want to make then press tab to highlight the `<Select>` option (or `<Finish>` if you've finished).

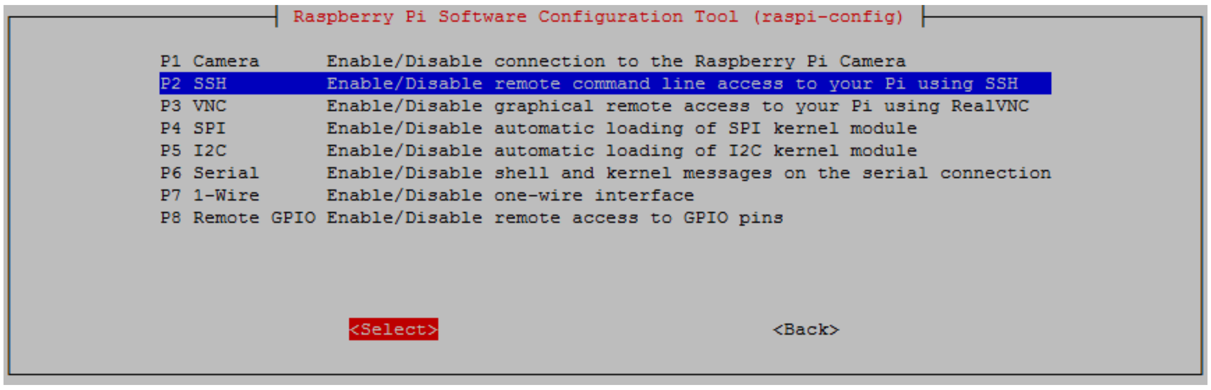
To enable SSH select '5 Interfacing Options' from the main menu.



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Change User Password Change password for the default user (pi)
2 Hostname             Set the visible name for this Pi on a network
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 peripherals
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>
```

Interfacing Options

From here we select 'P2 SSH'



```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

              <Select>                         <Back>
```

Enabling ssh

And we should be done!

Setting up the Client (Windows)

The client software we will use is called 'Putty[24]'. It is open source and available for download from here[25].

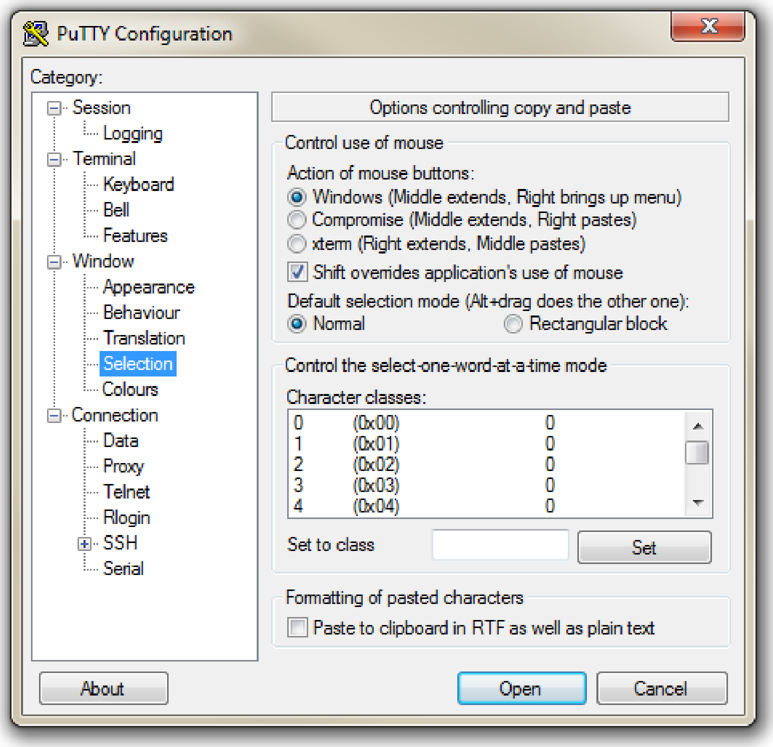
On the download page there are a range of options available for use. The best option for us is most likely under the '**For Windows on Intel x86**' heading and we should just download the 'putty.exe' program.

Save the file somewhere logical as it is a stand-alone program that will run when you double click on it (you can make life easier by placing a short-cut on the desktop).

Once we have the file saved, run the program by double clicking on it and it will start without problem.

The first thing we will set-up for our connection is the way that the program recognises how the mouse works. In the 'Window' Category on the left of the PuTTY Configuration box, click on the 'Selection' option. On this page we want to change the 'Action of mouse' option from the default of 'Compromise (Middle extends, Right paste)' to 'Windows (Middle extends, Right brings up menu)'. This keeps the standard Windows mouse actions the same when you use PuTTY.

[24]http://www.putty.org/
[25]http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html



Options controlling copy and paste

Control use of mouse

Action of mouse buttons:
(● Windows (Middle extends, Right brings up menu))
Compromise (Middle extends, Right pastes)
xterm (Right extends, Middle pastes)
[✓ Shift overrides application's use of mouse]
Default selection mode (Alt+drag does the other one):
[✓ Normal] Rectangular block

Control the select-one-word-at-a-time mode

Character classes:

```
0    (0x00)    0
1    (0x01)    0
2    (0x02)    0
3    (0x03)    0
4    (0x04)    0
```

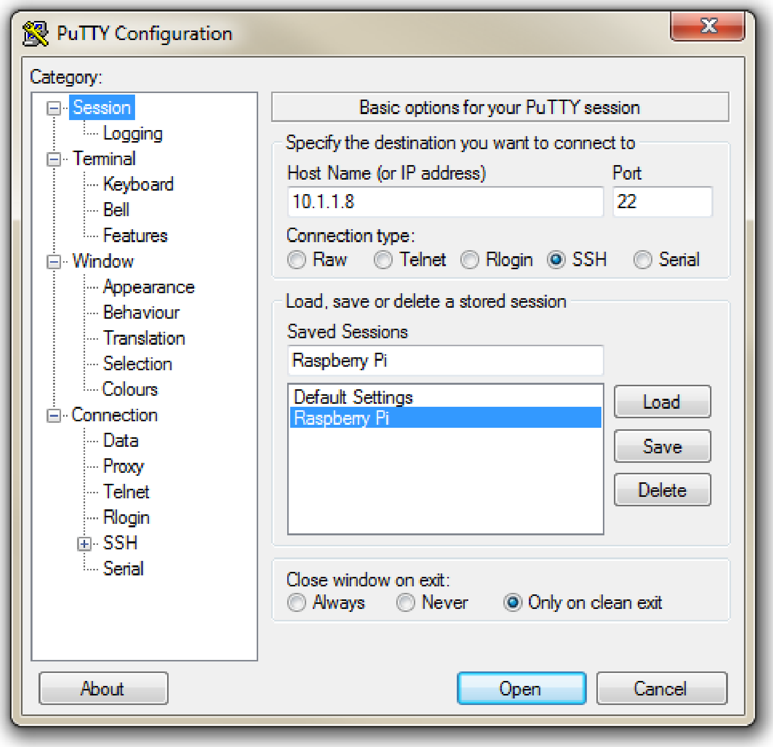
Set to class

Formatting of pasted characters

Paste to clipboard in RTF as well as plain text

PuTTY Selection Set-up

Now select the 'Session' Category on the left hand menu. Here we want to enter our static IP address that we set up earlier (10.1.1.160 in the example that we have been following, but use *your* one) and because we would like to access this connection on a frequent basis we can enter a name for it as a saved session (In the screen-shot below it is imaginatively called 'Raspberry Pi'). Then click on 'Save'.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 10.1.1.8

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: Raspberry Pi

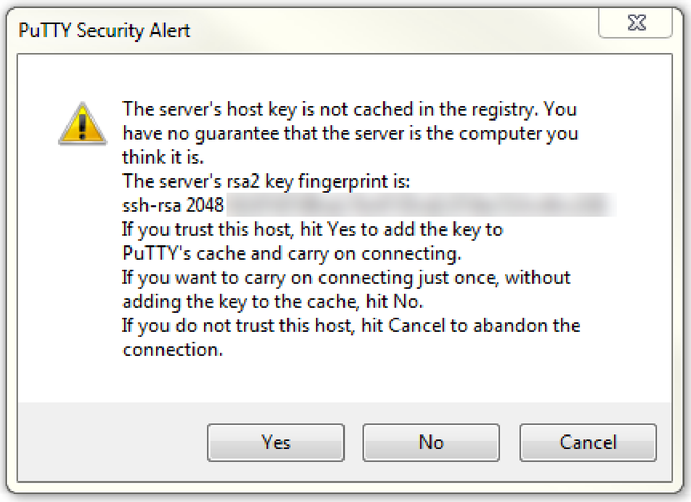
Default Settings
Raspberry Pi

Close window on exit: Always Never [✓ Only on clean exit]

PuTTY Session Set-up

Now we can select our raspberry Pi Session (per the screen-shot above) and click on the 'Open' button.

The first thing you will be greeted with is a window asking if you trust the host that you're trying to connect to.

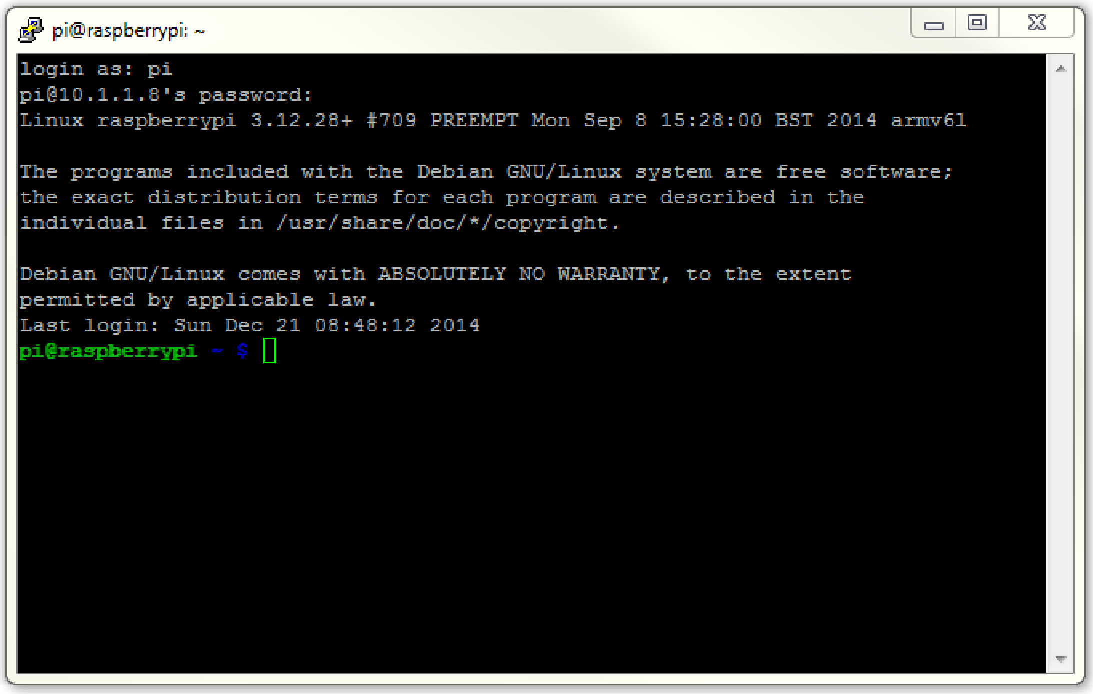


The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 [redacted]
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

PuTTY Session Connection

In this case it is a pretty safe bet to click on the 'Yes' button to confirm that we know and trust the connection.

Once this is done, a new terminal window will be shown with a prompt to `login as:` . Here we can enter our user name ('pi') and then our password (if it's still the default, the password is 'raspberry').



```
login as: pi
pi@10.1.1.8's password:
Linux raspberrypi 3.12.28+ #709 PREEMPT Mon Sep 8 15:28:00 BST 2014 armv6l

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Dec 21 08:48:12 2014
pi@raspberrypi ~ $
```

PuTTY Session Connected

There you have it. A command line connection via SSH. Well done.

If this is the first time that you've done something like this it can be a very liberating feeling. To complete the feeling of freedom let's set up a wireless network connection.

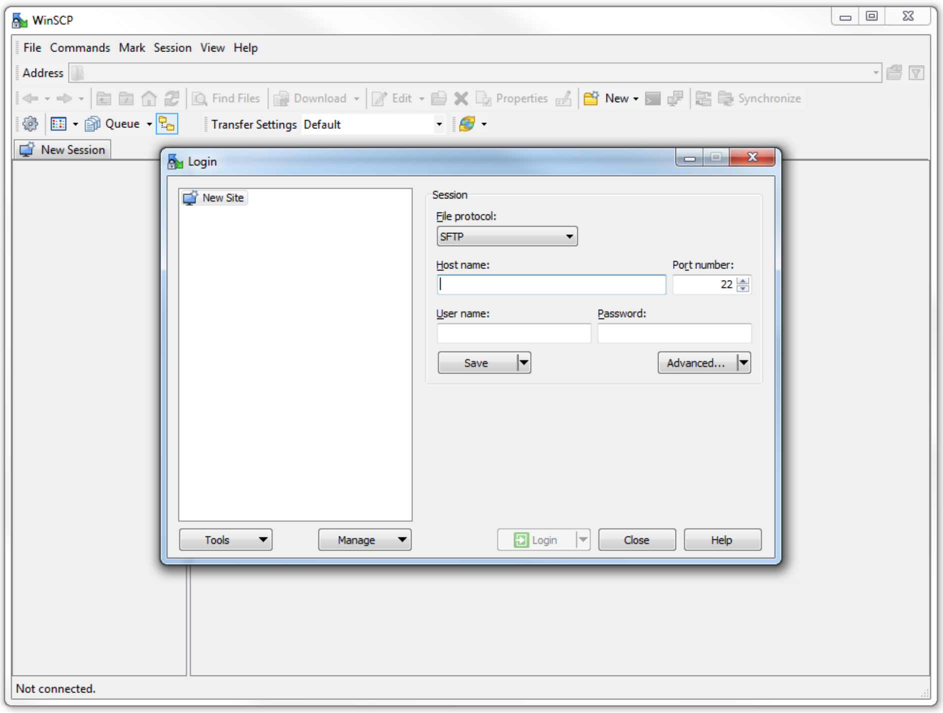
WinSCP

To make the process of transferring files from Windows easier I would recommend looking to the program WinSCP[26].

This provides a very intuitive way to copy files between your desktop and the Pi.

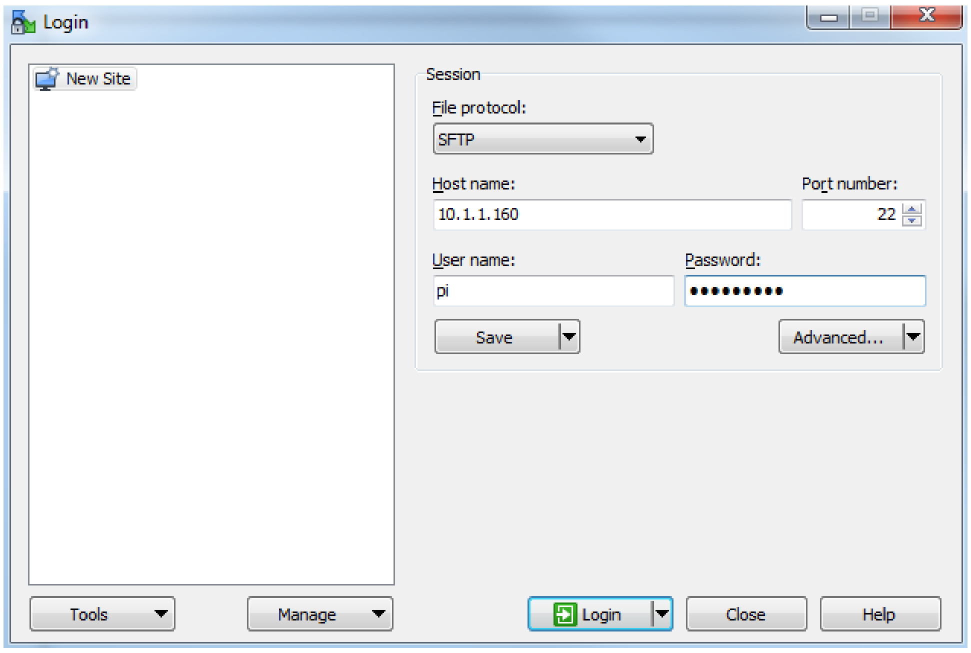
Download and install the program. Once installed, click on the desktop icon.

[26]https://winscp.net/eng/download.php



WinSCP New Login Page

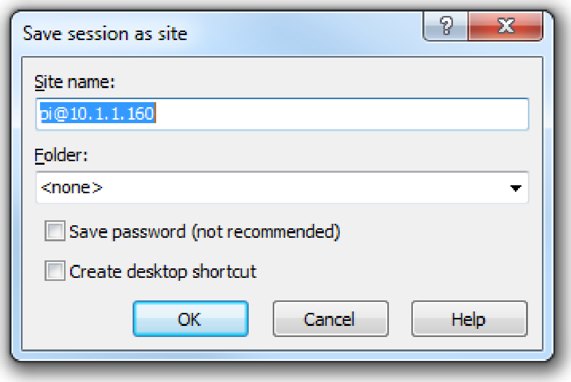
The program opens with default login page. Enter the 'Host name' field with the IP address of the Pi. Also put in the username and password of the Pi.



File protocol: SFTP
Host name: 10.1.1.160
Port number: 22
User name: pi
Password: •••••••••

WinSCP Host Name, User and Password

Click on 'Save' to save the login details for ease of future access.


Save session as site

Site name: pi@10.1.1.160

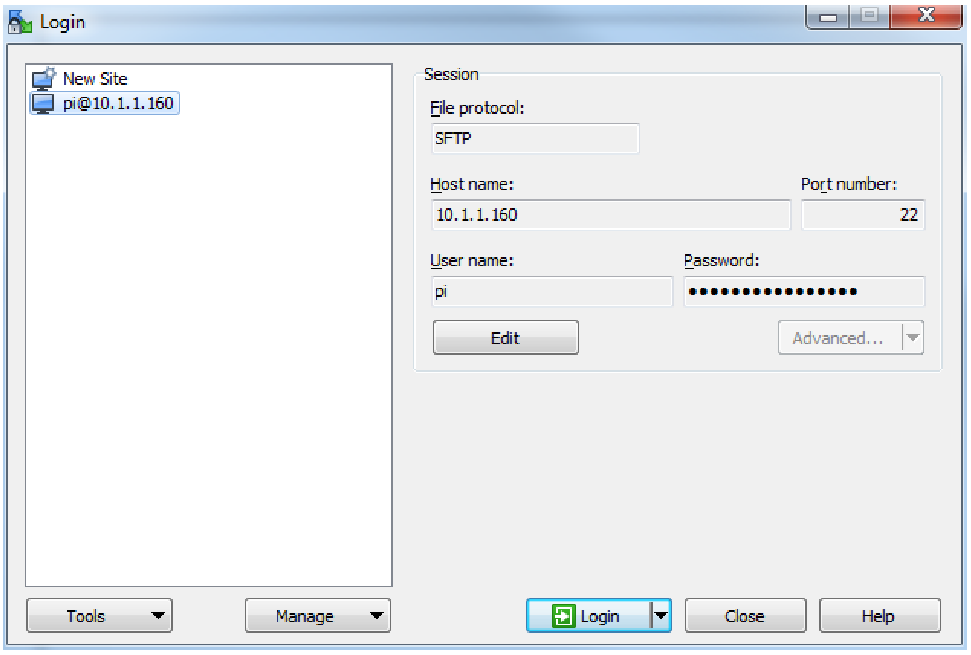
Folder: <none>

☐ Save password (not recommended)

☐ Create desktop shortcut

WinSCP Save the Session

Enter the ‘Site name’ as a name of the Pi or leave it as the default, with the user and IP address. Check the ‘Save password’ for a convenient but insecure way to avoid typing in the username and password in the future. Then press OK


Login

New Site
pi@10.1.1.160

Session

File protocol: SFTP

Host name: 10.1.1.160

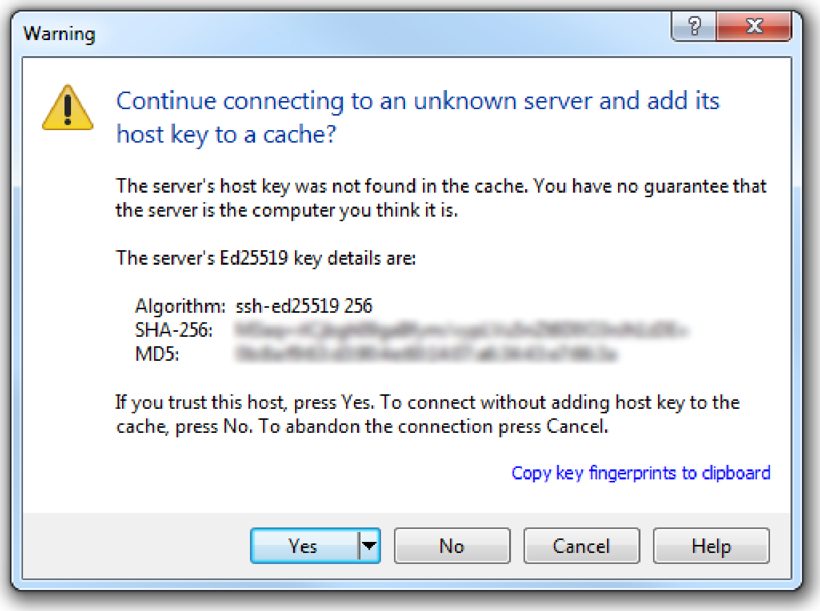
Port number: 22

User name: pi

Password: ••••••••••••••••

WinSCP Login

The saved login details now appear on the left hand pane. Click on ‘Login’ to log in to the Pi.


Warning

Continue connecting to an unknown server and add its host key to a cache?

The server's host key was not found in the cache. You have no guarantee that the server is the computer you think it is.

The server's Ed25519 key details are:

```
Algorithm:  ssh-ed25519 256
SHA-256:    [redacted]
MD5:        [redacted]
```

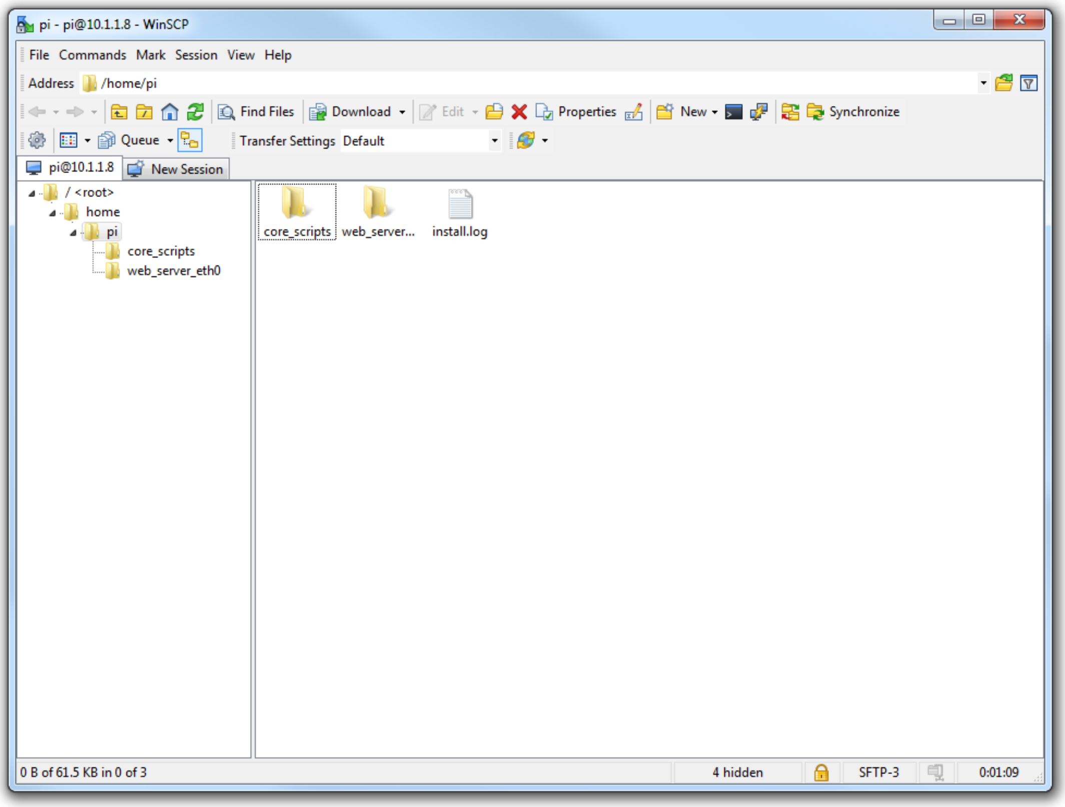
If you trust this host, press Yes. To connect without adding host key to the cache, press No. To abandon the connection press Cancel.

Copy key fingerprints to clipboard

WinSCP Warning

We will receive a warning about connecting to an unknown server for the first time. Assuming that we are comfortable doing this (i.e. that we know that we are connecting the Pi correctly) we can click on ‘Yes’.

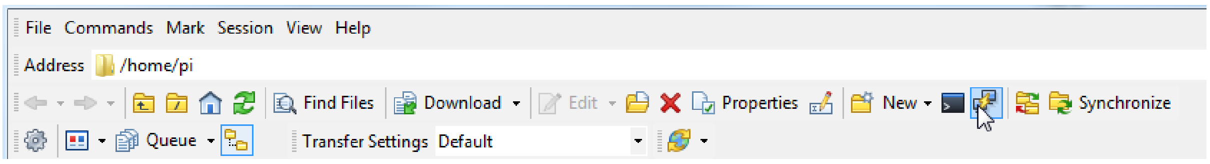
There is a possibility that it might fail on its first attempt, but tell it to reconnect if it does and we should be in!



WinSCP File Tree

Here we can see a familiar tree structure for file management and we have the ability to copy files via dragging and dropping them into place.

Assuming that we already have PuTTY installed we should be able to click on the 'Open Session in PuTTY' icon and we will get access to the command line.



WinSCP File Tree

Setting up a WiFi Network Connection

Our set-up of the Raspberry Pi will allow us to carry out all the (computer interface) interactions via a remote connection. However, the Raspberry Pi is currently making that remote connection via a fixed network cable. It could be argued that the lower number of connections that we need to run to our machine the better. The most obvious solution to this conundrum is to enable a wireless connection.

It should be noted that enabling a wireless network will not be a requirement for everyone, and as such, I would only recommend it if you need to. If you're using a model B3, B3+ or Zero W you have WiFi built in, otherwise you will need to purchase a USB WiFi dongle and correctly configure it.

Built in WiFi Enabling

We need to edit the file `wpa_supplicant.conf` at `/etc/wpa_supplicant/wpa_supplicant.conf`. This looks like the following;

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

The `country=NZ` line will probably indicate a different country depending on what you have set up as your localisation configuration.

Use the `nano` command as follows;

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

We need to add the ssid (the wireless network name) and the password for the WiFi network here so that the file looks as follows (using *your* ssid and password of course);

```
country=NZ
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="homenetwork"
    psk="h0mepassw0rd"
    key_mgmt=WPA-PSK
}
```

If you're not sure about the name (ssid) of your network, a simple test would be to use a phone or tablet to see what WiFi connection it is using (assuming that you are using your own WiFi connection).

Make the changes operative

To allow the changes to become operative we can type in;

```
sudo reboot
```

Once we have rebooted, we can check the status of our network interfaces by typing in;

```
ifconfig
```

This will display the configuration for our wired Ethernet port, our 'Local Loopback' (which is a fancy way of saying a network connection for the machine that you're using, that doesn't require an actual network (ignore it in the mean time)) and the wlan0 connection which should look a little like this;

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.99  netmask 255.255.255.0  broadcast 10.1.1.255
        inet6 fe80::8b9f:3e4f:dcf0:12a9  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:e3:b7:f2  txqueuelen 1000  (Ethernet)
        RX packets 51  bytes 9384 (9.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 35  bytes 6078 (5.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

This would indicate that our wireless connection has been assigned the dynamic IP address 10.1.1.99.

We should be able to test our connection by connecting to the Pi via SSH and 'PuTTY' on the Windows desktop using the address 10.1.1.99.

In theory you are now the proud owner of a computer that can be operated entirely separate from all connections except power!

Make the built in WiFi IP address static

In the same way that we would edit the `/etc/dhcpcd.conf` file to set up a static IP address for our physical connection (eth0) we will now edit it with the command…

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

This time we will add the details for the `wlan0` connection to the end of the file. Those details (assuming we will use the 10.1.1.17 IP address) should look like the following;

```
# Custom static IP address for wlan0.
interface wlan0
static ip_address=10.1.1.17/24
static routers=10.1.1.1
static domain_name_servers=10.1.1.1
```

What we could also do (if you haven't already) is remove the section for the eth0 connection so that it reverts to a dynamic address (assuming that the WiFi IP address is the one we want fixed.

Our wireless lan (`wlan0`) is now designated to be a static IP address (with the details that we had previously assigned to our wired connection) and we have added the 'ssid' (the network name) of the network that we are going to connect to and the password for the network.

Make the changes operative

To allow the changes to become operative we can type in;

```
sudo reboot
```

We're done!

WiFi Via USB Dongle

Using an external USB WiFi dongle can be something of an exercise if not done right. In my own experience, I found that choosing the right wireless adapter was the key to making the job simple enough to be able to recommend it to new users. Not all WiFi adapters are well supported and if you are unfamiliar with the process of installing drivers or compiling code, then I would recommend that you opt for an adapter that is supported and will work 'out of the box'. There is an excellent page on elinux.org[27] which lists different adapters and their requirements. I eventually opted for the Edimax EW-7811Un which literally 'just worked' and I would recommend it to others for it's ease of use and relatively low cost (approximately $15 US).

[27]http://elinux.org/RPi_USB_Wi-Fi_Adapters

Edimax WiFi USB Adapter

The same advice is given here as for the built in WiFi set-up. Bearing in mind that we are going to be adjusting our network connection, it is highly recommended that the following configuration changes take place with the keyboard / mouse and monitor connected to the Raspberry Pi (I.e. not via a remote desktop connection).

To install the wireless adapter we should start with the Pi powered off and install it into a convenient USB connection. When we turn the power on we will see the normal range of messages scroll by, but if we're observant we will note that there are a few additional lines concerning a USB device. These lines will most likely scroll past, but once the device has finished powering up and we have logged in we can type in…

```
dmesg
```

… which will show us a range of messages about drivers that are loaded to support discovered hardware.

Somewhere in that list (hopefully towards the end) will be a series of messages that describe the USB connectors and what is connected to them. In particular we could see a group that looks a little like the following;

```
[3.382731] usb 1-1.2: new high-speed USB device number 4 using dwc_otg
[3.494250] usb 1-1.2: New USB device found, idVendor=7392, idProduct=7811
[3.507749] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[3.520230] usb 1-1.2: Product: 802.11n WLAN Adapter
[3.542690] usb 1-1.2: Manufacturer: Realtek
[3.560641] usb 1-1.2: SerialNumber: 00345767831a5e
```

That is our USB adapter which is plugged into USB slot 2 (which is the '2' in `usb 1-1.2:`). The manufacturer is listed as 'Realtek' as this is the manufacturer of the chip-set in the adapter that Edimax uses.

Editing files

Be aware that while the following section describes the set-up of a `wlan1` WiFi connection as if it is the only one, it is completely possible to have already configured eth0 and built in wlan0 and to now add a third interface. The configuration below assumes that there has been no editing of the `wpa_supplicant.conf` file, but if you've already set up a built in wlan0 you don't need to do it again.

We need to edit two files. The first is the file `wpa_supplicant.conf` at `/etc/wpa_supplicant/wpa_supplicant.conf`. This looks like the following;

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

The `country=NZ` line will probably indicate a different country depending on what you have set up as your localisation configuration.

Use the `nano` command as follows;

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

We need to add the ssid (the wireless network name) and the password for the WiFi network here so that the file looks as follows (using *your* ssid and password of course);

```
country=NZ
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="homenetwork"
    psk="h0mepassw0rd"
    key_mgmt=WPA-PSK
}
```

If you're not sure about the name (ssid) of your network, a simple test would be to use a phone or tablet to see what WiFi connection it is using (assuming that you are using your own WiFi connection).

Make the changes operative

To allow the changes to become operative we can type in;

```
sudo reboot
```

Once we have rebooted, we can check the status of our network interfaces by typing in;

```
ifconfig
```

This will display the configuration for our wired Ethernet port, our 'Local Loopback' (which is a fancy way of saying a network connection for the machine that you're using, that doesn't require an actual network (ignore it in the mean time)) and the wlan1 connection which should look a little like this;

```
wlan1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.1.97  netmask 255.255.255.0  broadcast 10.1.1.255
        inet6 fe80::c4e4:a6e5:9788:d2c2  prefixlen 64  scopeid 0x20<link>
        ether 00:ec:0b:4c:6b:99  txqueuelen 1000  (Ethernet)
        RX packets 106  bytes 18616 (18.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 34  bytes 5681 (5.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

This would indicate that our wireless connection has been assigned the dynamic IP address 10.1.1.97.

We should be able to test our connection by connecting to the Pi via SSH and 'PuTTY' on the Windows desktop using the address 10.1.1.97.

Make USB WiFi IP address static

In the same way that we would edit the `/etc/dhcpcd.conf` file to set up a static IP address for our physical connection (eth0) we will now edit it with the command…

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

This time we will add the details for the `wlan1` connection to the end of the file. Those details (assuming we will use the 10.1.1.160 IP address) should look like the following;

```
# Custom static IP address for wlan1.
interface wlan1
static ip_address=10.1.1.160/24
static routers=10.1.1.1
static domain_name_servers=10.1.1.1
```

Make the changes operative

To allow the changes to become operative we can type in;

```
sudo reboot
```

We're done!

Reconnecting to the wireless network automatically

This portion is completely optional and I only include it for the sake of those who might be installing in areas where wireless access is problematic.

I have found with experience that in spite of my best intentions, sometimes when setting up a Raspberry Pi to maintain a WiFi connection, if it disconnects for whatever reason it may not reconnect automatically.

To solve this problem we're going to write a short script that automatically reconnects our Pi to a WiFi network. The script will check to see if the Pi is connected to our local network and, if it's off-line, will restart the wireless network interface. We'll use a cron job to schedule the execution of this script at a regular interval.

Let's write a script

First, we'll need to check if the Pi is connected to the network. This is where we'll try to `ping` an IP address on our local network (perhaps our gateway address?). If the `ping` command succeeds in getting a response from the IP address, we have network connectivity. If the command fails, we'll turn off our wireless interface (wlan1) and then turn it back on (yes, the timeless solution of turning it off and on).

The script looks a little like this;

```
#!/bin/bash

# The IP address of our gateway on our local router
GATEWAY=10.1.1.1

# Send two pings, with the output going to /dev/null
ping -c2 ${GATEWAY} > /dev/null

# Check to see if the returned value from ping ($?)
# is not 0 and then act to restart wlan1 if necessary
if [ $? == 0 ]
then
    # Restart wlan1 (the wireless interface)
    ifconfig wlan1 down
    ifconfig wlan1 up
fi
```

Use nano to create the script, name it something like wifistart.sh, and save it in `/usr/local/bin`.

```
sudo nano /usr/local/bin/wifistart.sh
```

We also need to make sure it's executable by running `chmod` (using `sudo`) as follows;

```
sudo chmod +x /usr/local/bin/wifistart.sh
```

Lets run our script on a regular schedule

To make our WiFi checking script run automatically, we'll schedule a cron job using crontab;

```
crontab -e
```

… and add this line to the bottom:

```
*/5 * * * * /usr/bin/sudo -H /usr/local/bin/wifistart.sh >> /dev/null 2>&1
```

This runs the script every 5 minutes with sudo permissions, writing its output to /dev/null so it doesn't spam syslog.

Let's test it

To test that the script works as expected, we will want to take down the wlan1 interface and wait for the script to bring it back up. Before taking down wlan1, we might want to adjust the interval in `crontab` to 1 minute. And fair warning, when we disconnect wlan1, we will lose that network interface, so we will need to either have a local keyboard / monitor connected, have another network interface set up or be **really** confident that we've got everything set up right first time.

To take down wlan1 to confirm the script works, run:

```
sudo ifconfig wlan1 down
```

After waiting for 5 (or 1) minutes, we could try ssh-ing back into the Raspberry Pi or if we're keen we could have a `ping` command running on another server checking the interface to show when it stops and when it (hopefully) starts again. Assuming everything works, our Pi should reconnect seamlessly.

Setting up the Raspberry Pi Software

While the Raspberry Pi is a capable computer, we still need to install software on it to allow us to gather, store and present our data.

The software we will be using is based on the Linux Operating System. Because this is potentially unfamiliar territory (for those who haven't used Linux or had some practical computing experience), we will take our time and explain things as we go.

Web Server, PHP and Database

Because we want to be able to present the data we will be collecting, we need to set up a web server that will return measurements to other computers that will be browsing within the network (remembering that this is not intended to be connected to the Internet, just inside your home network). This type of connection is called a RESTful service.

> REpresentational State Transfer (REST), or RESTful web services provide an Application Program Interface (API) between computer systems using HTTP requests. That means that we can essentially 'browse' to a web page that is hosted on our Pi and it will respond with data about the values we are measuring.

At the same time as setting up a web server on the Pi we will install PHP. PHP is a scripting language that is widely used in developing web pages. And because we will want to store our data somewhere we will add in the SQLite[28] database. SQLite is a self-contained database engine that reads and writes directly to ordinary disk files. A complete SQL database is contained in a single file which can be transferred between platforms for backup or restoration purposes. SQLite is touted as the most widely used database engine in the world.

Install NGINX and PHP

The web server that we will use is called NGINX[29] (pronounced "Engine X"). NGINX is an open-source web server that is often recommended for its performance and low resource consumption. This obviously makes it ideal for hardware such as the Raspberry Pi. In spite of being targeted as something of a 'light' application, it is extremely powerful, capable and it is widely used in large scale applications.

We can start the install process using the following;

[28]https://www.sqlite.org/index.html
[29]https://www.nginx.com/

```
sudo apt-get install nginx php-fpm
```

We're familiar with apt-get already, but this time we're including more than one package in the installation process. Specifically we're including `nginx` and `php-fpm`.

'nginx' is obviously the name of the NGINX web server and `php-fpm` is for PHP.

The Raspberry Pi will advise you of the range of additional packages that will be installed at the same time (to support those we're installing (these additional packages are 'dependencies')). Agree to continue and the installation will proceed. This should take a few minutes or more (depending on the speed of your Internet connection).

Configuration

Firstly we should edit the default file that will get displayed as a web page if none is specified. What we want to do is to include the option to redirect to an `index.php` file.:

```
sudo nano /etc/nginx/sites-available/default
```

Replace the line:

```
index index.html index.htm index.nginx-debian.html;
```

with

```
index index.html index.htm index.php;
```

Now we want to edit the portion of the file that will handle all requests for resources that end in `.php`.

Replace the section of the file that looks like this;

```
        #location ~ \.php$ {
        #       include snippets/fastcgi-php.conf;

        #       # With php-fpm (or other unix sockets):
        #       fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        #}
```

With this;

```
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;

                # With php-fpm (or other unix sockets):
                fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        }
```

⚠ Be careful to include the last curly brace at the bottom of that block!

NGINX has its default web page location at `/var/www/html` on Raspbian. We are going to change the permissions / ownership of that folder by running following two commands:

```
sudo chown -R www-data:pi /var/www/html/
sudo chmod -R 770 /var/www/html/
```

This is necessary to let the ‘pi’ user edit the files in that location easily.

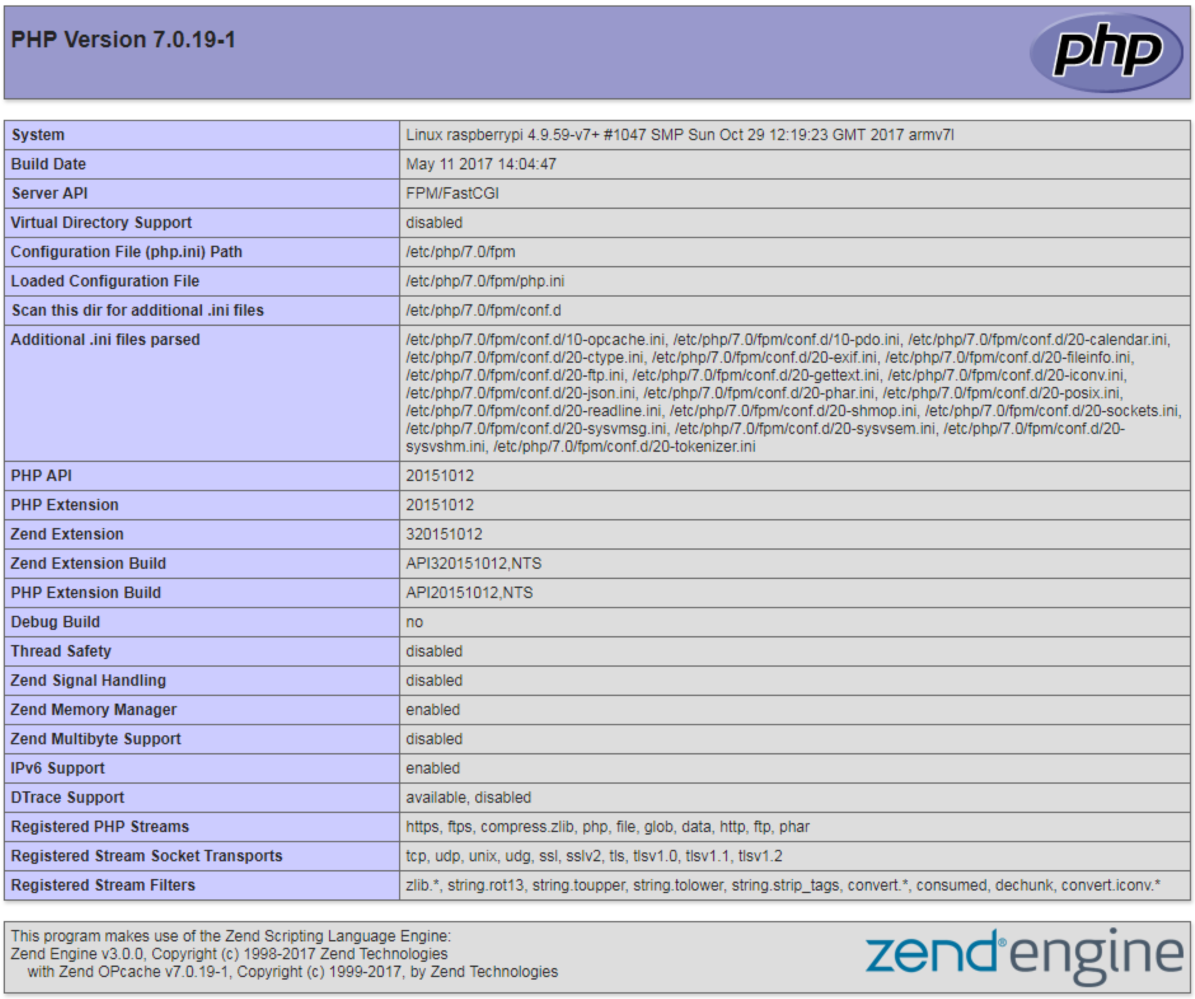
Now let’s create a suitable index.php test file with the following command:

```
echo "<?php phpinfo(); ?>" > /var/www/html/index.php
```

We can restart the NGINX service so that the changes we have made can take effect:

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

Now if we go to to the IP address of our Pi in a browser (in the example we are using it’s 10.1.1.120) and type it in to the URL bar to test. Something like the following should be displayed;


PHP Version 7.0.19-1

System	Linux raspberrypi 4.9.59-v7+ #1047 SMP Sun Oct 29 12:19:23 GMT 2017 armv7l
Build Date	May 11 2017 14:04:47
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/fpm
Loaded Configuration File	/etc/php/7.0/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/fpm/conf.d
Additional .ini files parsed	/etc/php/7.0/fpm/conf.d/10-opcache.ini, /etc/php/7.0/fpm/conf.d/10-pdo.ini, /etc/php/7.0/fpm/conf.d/20-calendar.ini, /etc/php/7.0/fpm/conf.d/20-ctype.ini, /etc/php/7.0/fpm/conf.d/20-exif.ini, /etc/php/7.0/fpm/conf.d/20-fileinfo.ini, /etc/php/7.0/fpm/conf.d/20-ftp.ini, /etc/php/7.0/fpm/conf.d/20-gettext.ini, /etc/php/7.0/fpm/conf.d/20-iconv.ini, /etc/php/7.0/fpm/conf.d/20-json.ini, /etc/php/7.0/fpm/conf.d/20-phar.ini, /etc/php/7.0/fpm/conf.d/20-posix.ini, /etc/php/7.0/fpm/conf.d/20-readline.ini, /etc/php/7.0/fpm/conf.d/20-shmop.ini, /etc/php/7.0/fpm/conf.d/20-sockets.ini, /etc/php/7.0/fpm/conf.d/20-sysvmsg.ini, /etc/php/7.0/fpm/conf.d/20-sysvsem.ini, /etc/php/7.0/fpm/conf.d/20-sysvshm.ini, /etc/php/7.0/fpm/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv2, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.0.0, Copyright (c) 1998-2017 Zend Technologies
    with Zend OPcache v7.0.19-1, Copyright (c) 1999-2017, by Zend Technologies

Testing the web server

Marvellous.

Database

As mentioned earlier, we will use a SQLite database to store the information that we collect.

SQLite is incredibly easy to install.

```
sudo apt-get install sqlite3 php7.0-sqlite3
```

And that's it!

Because SQLite does not rely on a client-server relationship, applications that interact with the SQLite database read and write directly to the database file (or files). It therefore relies on the security of the permissions on the operating system to provide separation. This means that accessing the database from a separate computer is problematic, but it simplifies interaction for the user that is operating the database locally.

Create a database and a table

When we read data from our sensor, we will record it in a database. SQLite is a database program, but we still need to set up a database file that SQLite will read. In fact when we come to record and explore our data we will be dealing with a 'table' of data that will exist inside a database.

We will create a database called 'measurements' and in that database we will create a table called 'light' That table will record regular values from our light sensor (LDR) and the time that they were taken.

Creating our database and initiating interaction with it is done as follows;

```
sqlite3 measurements
```

Once the program is started we are presented with a prompt for the database;

```
SQLite version 3.16.2 2017-01-06 16:32:41
Enter ".help" for usage hints.
sqlite>
```

From this prompt we can begin to provide commands and when we are finished we can exit from SQLite by typing in `.quit`. At any stage if we forget what command we should be running we can type in `.help` and the program will give us a list of commands.

We will create a table called 'light' which will contain a **d**ate **t**ime **g**roup field called 'dtg'.

Since SQLite doesn't have a dedicated storage class set aside for dates and/or times we will store the times as text. We will do this in the format `YYYY-MM-DD HH:MM:SS`.

For the light value we will name the field 'ldr' and we will use the 'INTEGER' type.

Enter each of the following lines at the `sqlite>` prompt. The semicolon represents the end of the command.

```
CREATE TABLE IF NOT EXISTS light (
'dtg' TEXT,
'ldr' INTEGER NOT NULL);
```

We can then confirm that the table exists using the command '.table';

```
sqlite> .table
light
```

There we have it! A simple database and a table ready to go.

Connecting Analog Sensors to the Raspberry Pi

The Raspberry Pi is a marvel of connectivity. It's 40 pin header and associated peripheral ports provide a spectacular range of options to interface with the world outside the Raspberry Pi. However, one feature that the Pi doesn't have built in is the facility to accept an analog input.

Analog vs. analogue

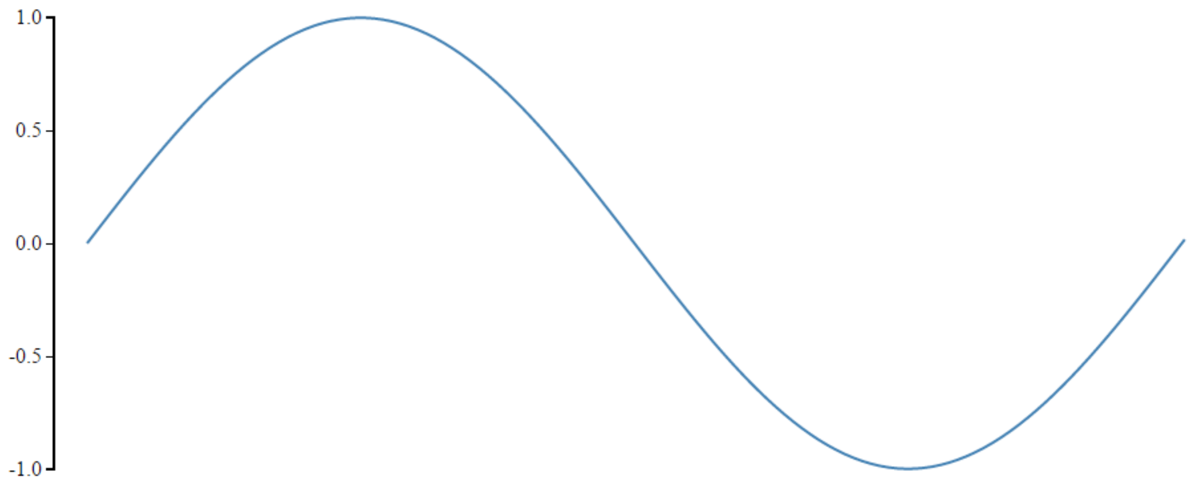
With the word traditionally spelled analogue, American English tends to drop the silent -ue, making analog. The spellings are largely interchangeable, though analog is usually used in relation to electronics, while analogue is often used in the sense something that bears analogy to something else. Frankly I'm torn. I'm going with 'analog' in the text for this book, but my instincts tell me 'analogue'. I'm sure the Internet has an opinion.

Analog and Digital

Signals (or even information in general) can be broken down into two different types; Analog and digital.

Analog

An analog signal is one that has an infinitely variable range of values that can change over time.



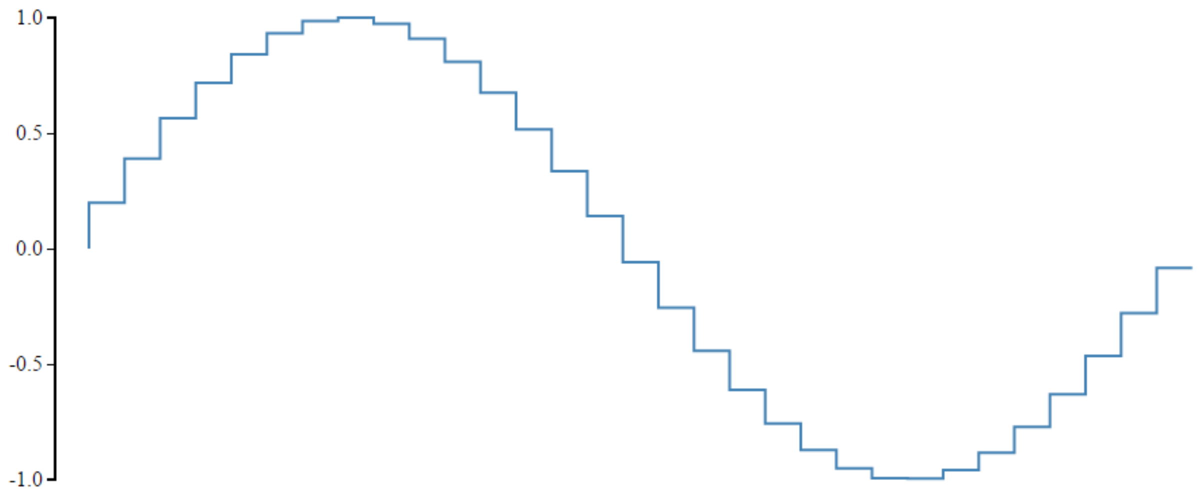
Analog

If we consider the question of how much light is shining outside we could imagine that the level of brightnesses varies between the blackness of a moonless night and a overcast sky to a cloudless day with the sun high in the sky.

These are rough approximations of dark and light, but between the two extremes is a range of brightness levels which are always changing. If we wanted to measure how bright it was at any particular time we could set ourselves a numeric range of 0 representing the middle of the night and 100 representing the middle of the day and the number that represented the brightness at any particular time would be somewhere between those two numbers. Typically in electronics an analog signal is a voltage that will be anywhere on that variable range between two limits. So straight up we can see that our analog sensor is going to be a device that provides an output value that varies between two extremes.

Digital

A digital signal represents information as discrete values.



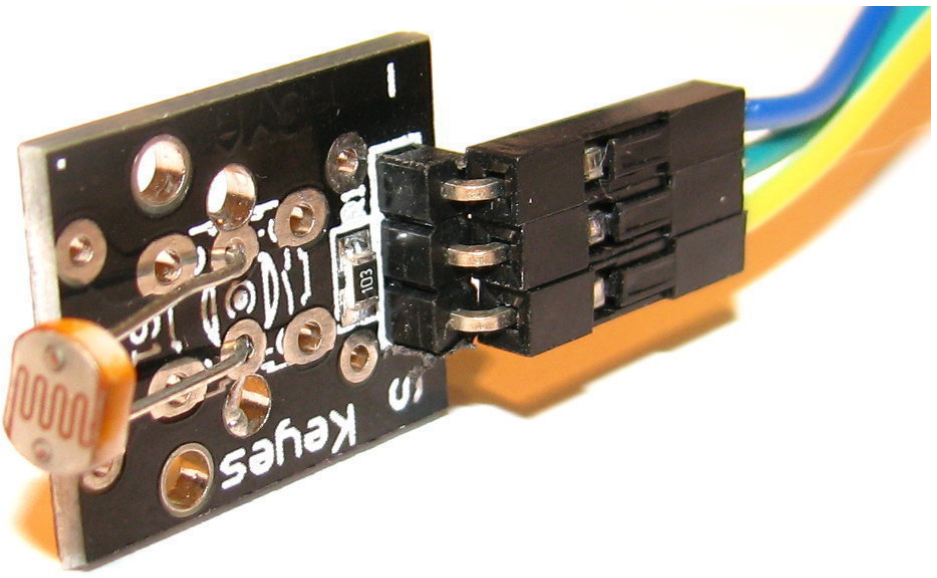
Digital

For example at it's most fundamental the light level outside could be described as dark or light. Represented numerically this could be dark = 0 and bright = 1. While this is perfectly valid, we would often prefer to have a little more granularity in our measurement and so we can increase the number of discrete steps that represent light levels to match our expectations of the type of information we're interested in. if we add another couple of levels, we could have light that was dark = 0, dim = 0.33, glowing = 0.66 and bright = 1. We can continue to improve the resolution of our numerical perception of the level of light in a process that is called Analog to Digital Conversion or ADC. Essentially creating steps that represent different levels of what would otherwise be a smooth transition.

The Boards

The Analog Sensor

While this project is more about the conversion of analog signals into digital ones, this project will use a Keyes KY-018[30] sensor based on a **L**ight **D**ependent **R**esistor (LDR) to produce a variable resistance in the presence of different light levels. An applied voltage (from the Pi) returns a variable voltage from the LDR. It is this variable voltage that is then digitised with the ADC.



Keyes KY-018 Analog Light Sensor

In essence there are a range of different sensors that could be used to produce an analog signal. I have successfully also connected the Keyes analog hall effect sensor (KY-035, which senses magnetic fields) and are will be others in that range that will work in the same way.

The Light Dependant Resistor (LDR or Photoresistor)

Our sensor will use an LDR to produce a variable resistance in the presence of different light levels.

In the dark, their resistance is very high, sometimes up to 1MΩ, but when the LDR sensor is exposed to light, the resistance drops dramatically, even down to a few ohms, depending on the light intensity. LDRs have a sensitivity that varies with the wavelength of the light applied and are non-linear devices.

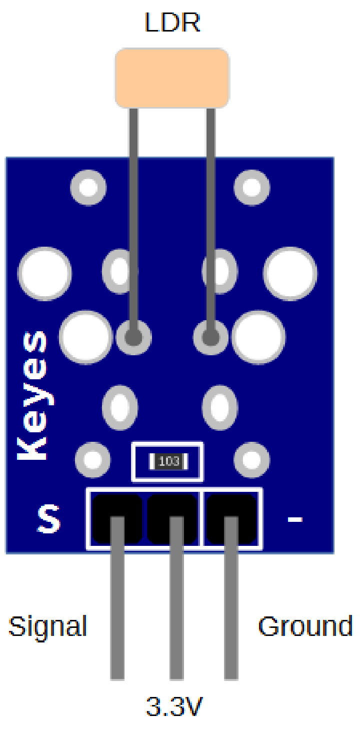
They are widely used in cameras, solar garden lights, clocks, mini night-lights, and a variety of light control devices.

Specifications from a typical LDR[31] show that as illumination increases, the resistance of an LDR decreases.

[30]https://tkkrlab.nl/wiki/Arduino_KY-018_Photo_resistor_module
[31]http://kennarar.vma.is/thor/v2011/vgr402/ldr.pdf

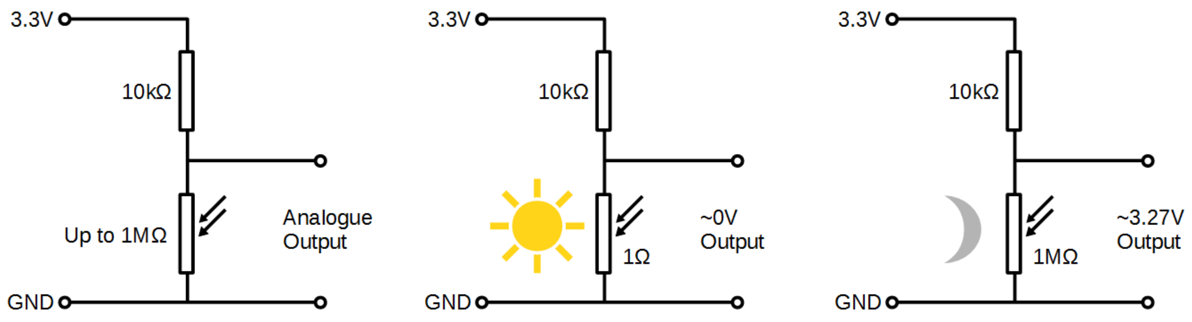
Light Level	Resistance
Moonlight	1,000,000 Ohms
60W bulb at 1m	6,000 Ohms
Fluorescent Lighting	1,000 Ohms
Bright sunlight	1 Ohm

The Keyes KY-018 sensor board comprises an LDR and a fixed resistor with header pins for connecting the ground, the reference voltage (we will use the 3.3V from the Pi) and the sensors analog voltage output.



Keyes KY-018 Photoresister Sensor Board

If we consider a simplified circuit of our sensor, the LDR in series with a fixed resistor allows the variation in resistance to develop a variation in output voltage.



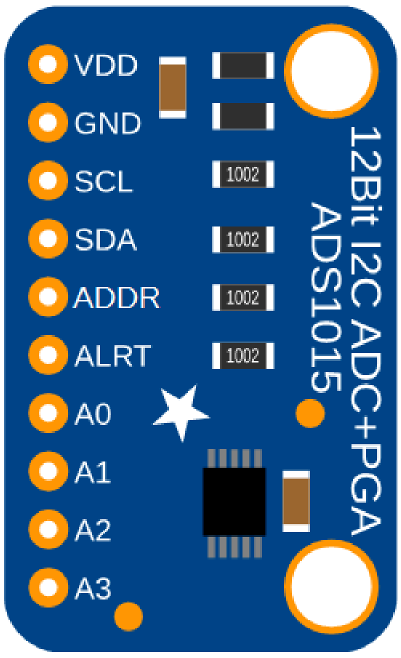
LDR Sensor Output Voltage

Analog to Digital Conversion (ADC)

Luckily there are a wide range of options available to convert analog signals into digital ones. Therefore, people wanting to receive an analog signal with a Raspberry Pi can simply include a separate ADC into their project and it will work wonderfully. That's what we're going to do here using the ADS1015 from Adafruit[32]. The ADS1015 has a 12bit resolution giving it the ability to convert an analog signal into one of 4096 discrete levels.

The ADS1015 Analog to Digital Converter

The ADS1015 is actually a component *on* our ADC board. This component is manufactured by integrated circuits manufacturer Texas Instruments. The circuit board that we're using in the project is from Adafruit. It incorporates some interconnection circuity to make the signals as stable as practical and to provide a convenient physical interface (via header pins). The ADS1015 provides 12-bit (4096 levels) precision at up to 3300 readings per second (the rate is programmable). The board can be configured to accept four sensors of the type we will be using (single-ended), or two differential channels (which use two varying signals instead of a single signal and a ground). There is also a programmable gain amplifier built in with up to x16 gain, to help amplify smaller signals to the full range. The ADC can operate on a voltage range from 2V to 5V Which is applied to the VDD pin).



ADS1015 Sensor Board

If all this sounds a bit 'electrickery', don't worry. The aim here is to provide ourselves with enough information to get us started and if we feel like pressing on and learning more we will :-).

The ADS1015 will send the digital levels to the Pi via the I2C communications protocol. The

[32]http://www.adafruit.com/products/1083

address that this connection is made on can be changed to one of four options so you can have up to 4 ADS1015's connected for up 16 sensor inputs!

Inter-Integrated Circuit or I2C (pronounced as either I-squared-C or I-2-C) connection is generically referred to as a "two-wire interface". It's commonly used to attach low-speed peripherals to computing devices.

I2C can be used to connect up to 127 nodes via a bus which has two data wires, called SCL and SDA. SCL is the **S**erial **CL**ock line which is used to synchronize all data transfers over the I2C bus. SDA is the **S**erial **DA**ta line. The I2C bus works on a 'Master - Slave' system, where in this case the master is the Raspberry Pi. Slaves can be integrated circuits such as sensors or micro controllers.

When the master wishes to communicate with a slave it sends a series of pulses down the SDA and SCL lines. The data that is sent includes a unique address that identifies the slave with which the master needs to interact. When data is being sent on the SDA line, clock pulses are sent on the SCL line to keep master and slave synchronised..

Measure

Hardware required

- A Raspberry Pi (huge range of sources)
- ADS1015 Analog to Digital Converter from Adafruit[33].
- Female to Female Dupont connector cables (Deal Extreme[34] or build your own!)
- Photoresistor module KY-018[35] from Keyes[36].

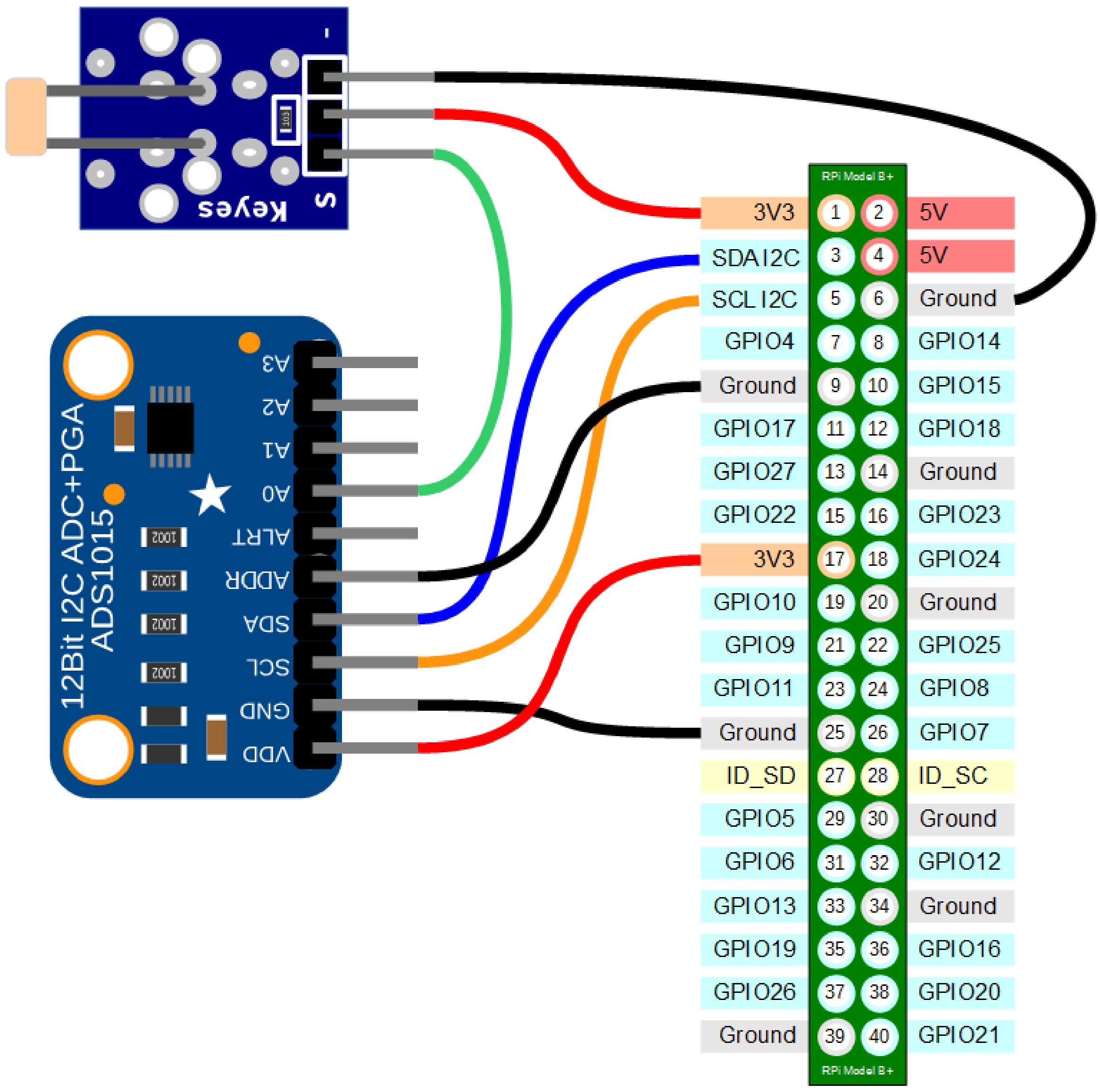
Connect

The LDR sensor board should be connected with ground pin (labelled '-') to a ground connector, the reference voltage pin (in the case of the board shown below the centre pin) to a 3.3V connector and the signal output pin (labelled 'S') to the A0 pin on the ADS1015 ADC.

The ADS1015 board should have the VDD pin connected to a 3.3V pin, the GND to a ground pin, the SCL pin to the SCL I2C connector on pin 5 and the SDA pin to the SDA I2C connector on pin 3 and lastly the ADDR pin should be connected to ground.

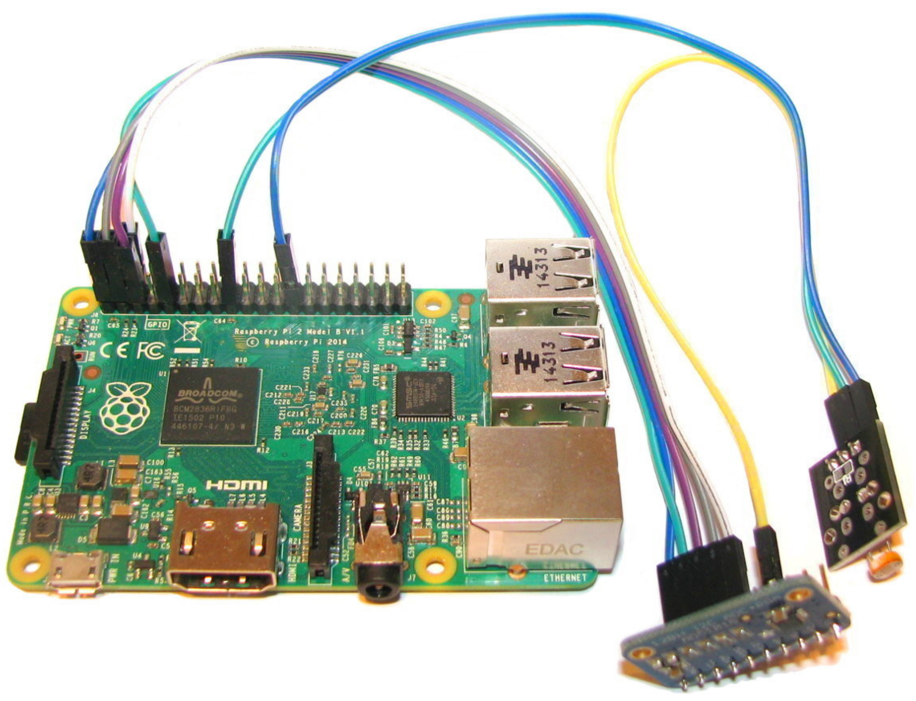
Both boards will support a connection to the 'VDD' and reference voltage connector of 5V, **but this is not advisable for the Raspberry Pi** as the resulting signal levels on the SDA connector *may* be higher than desired for the Pi's input. This connection can be safely used with an Arduino board.

[33]http://www.adafruit.com/products/1083
[34]http://www.dx.com/p/8-pins-female-to-female-dupont-cable-for-raspberry-pi-multicolored-21cm-326450
[35]https://tkkrlab.nl/wiki/Arduino_KY-018_Photo_resistor_module
[36]http://en.keyes-robot.com/index.aspx



LDR Sensor Board Connection

Connecting the sensor practically can be achieved in a number of ways. You could use a Pi Cobbler break out connector mounted on a bread board connected to the appropriate pins. But because the connection is relatively simple we could build a minimal configuration that will plug directly onto the pins using Dupont header connectors and jumper wire. The image below shows how simple this can be.



Physical Connection of ADS1015 and LDR Sensor

Test

Since the ADS1015 uses the I2C protocol to communicate, we need to load the appropriate kernel support modules onto the Raspberry Pi to allow this to happen.

Firstly make sure that our software is up to date

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

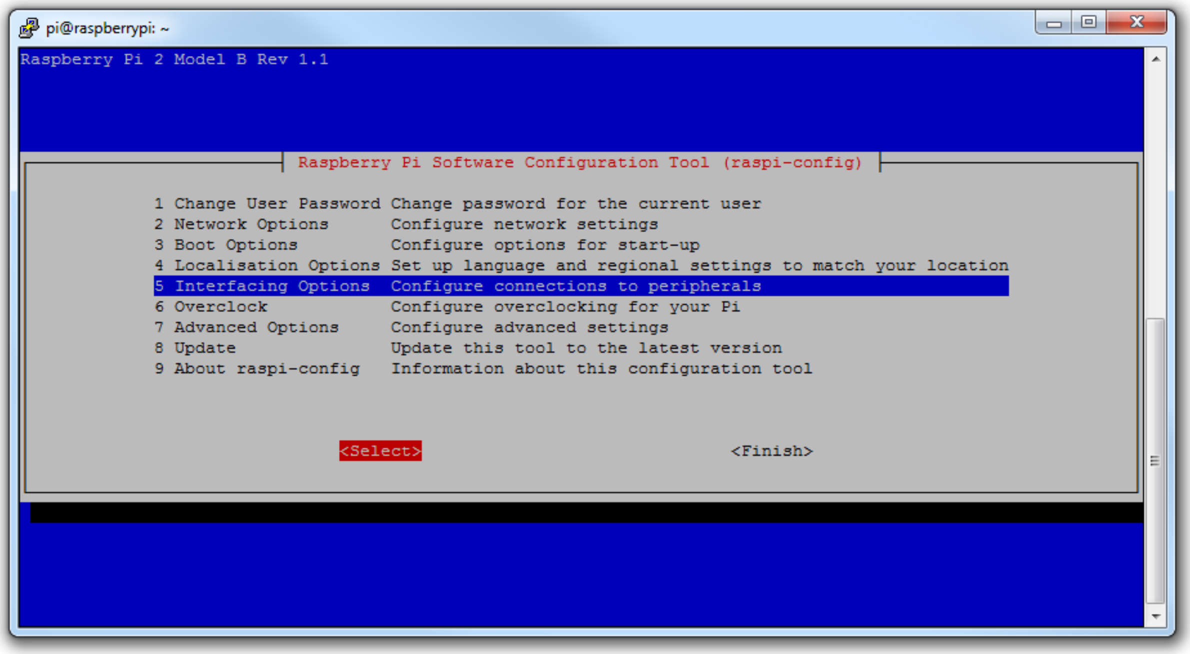
Since we are using the Raspbian distribution there is a simple method to start the process of configuring the Pi to use the I2C protocol.

We can start by running the command;

```
sudo raspi-config
```

This will start the Raspberry Pi Software Configuration Tool.

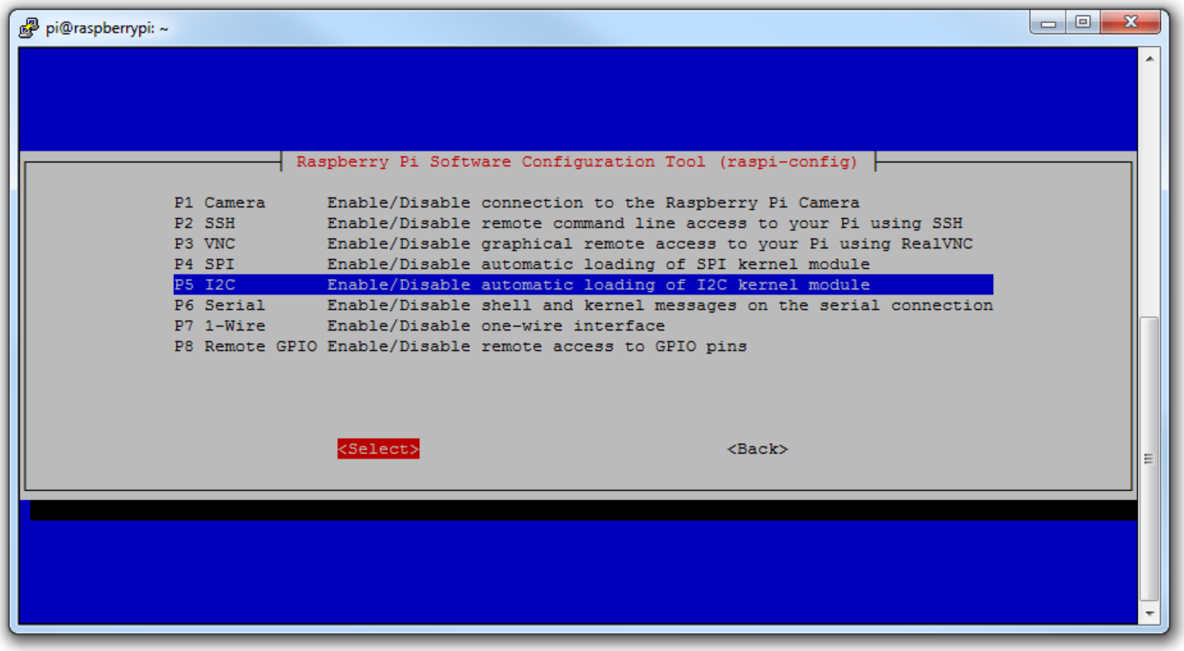
On the first page select the Interfacing Options with the arrow keys and then tab to select



```
pi@raspberrypi: ~
Raspberry Pi 2 Model B Rev 1.1

            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 peripherals
  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>
```

Interfacing Options

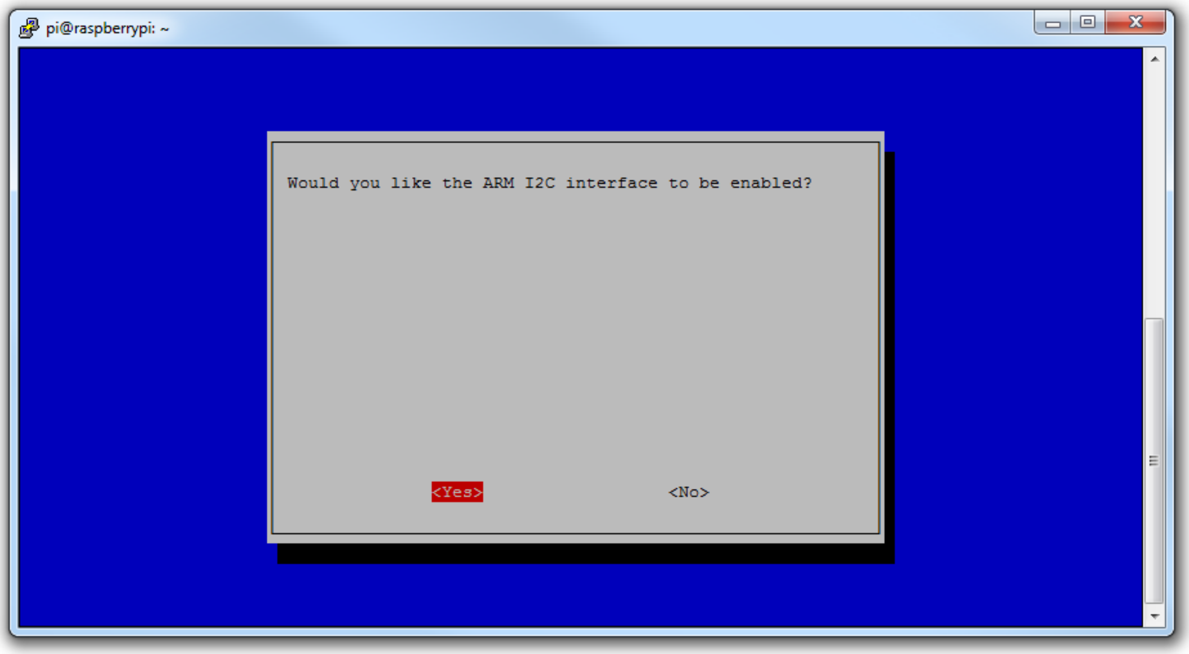
Then we select the I2C option for automatic loading of the kernel module;



```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/Disable connection to the Raspberry Pi Camera
P2 SSH         Enable/Disable remote command line access to your Pi using SSH
P3 VNC         Enable/Disable graphical remote access to your Pi using RealVNC
P4 SPI         Enable/Disable automatic loading of SPI kernel module
P5 I2C         Enable/Disable automatic loading of I2C kernel module
P6 Serial      Enable/Disable shell and kernel messages on the serial connection
P7 1-Wire      Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to GPIO pins

             <Select>                        <Back>
```

Automatic Loading

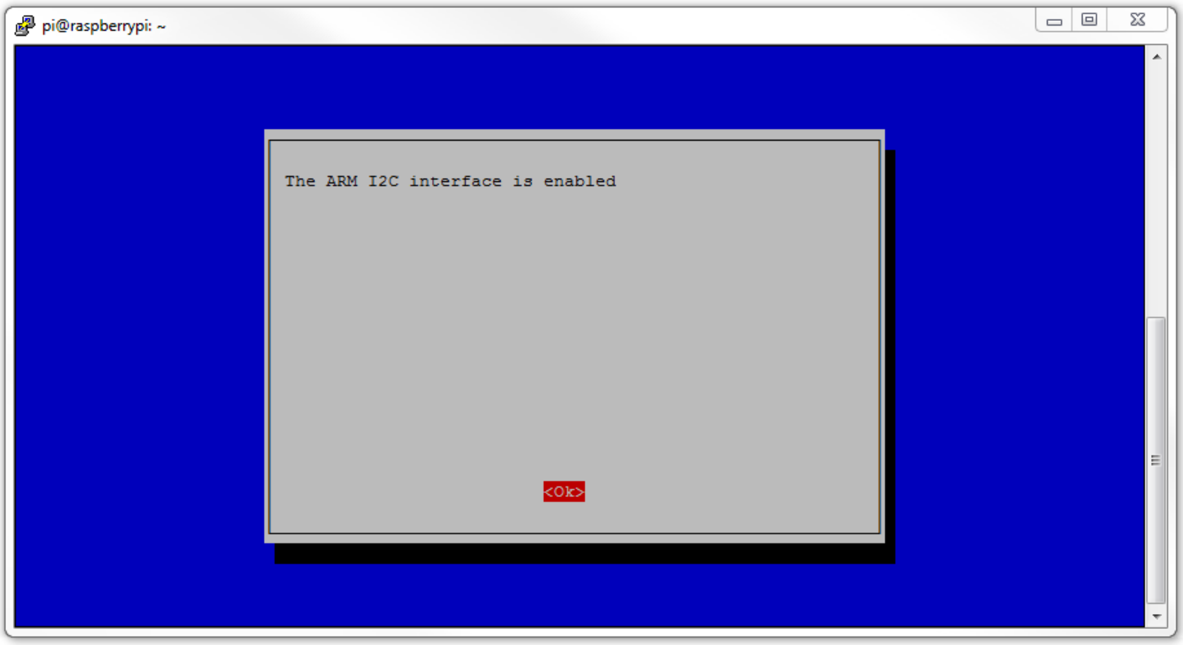
Would we like the ARM I2C interface to be enabled? Yes we would;



```
Would you like the ARM I2C interface to be enabled?

         <Yes>                <No>
```

ARM I2C Interface Enabled

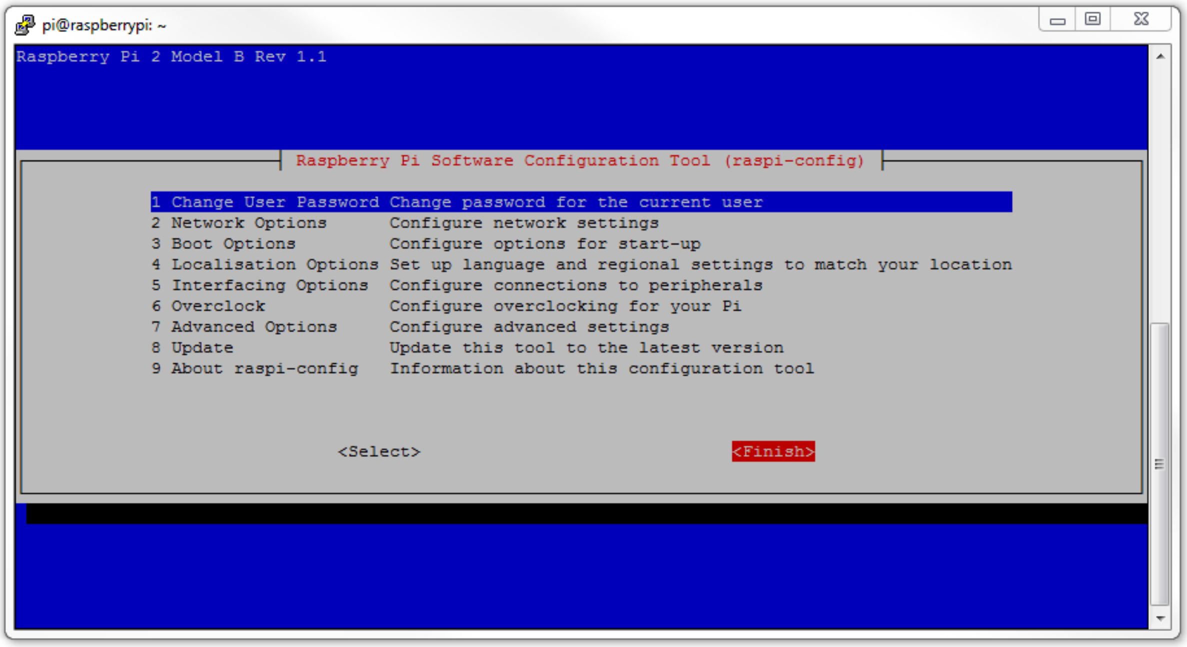
Press ‘OK’ to acknowledge that the interface is enabled.



```
The ARM I2C interface is enabled
```

Enabled!

Press tab to select 'Finish'.



```
Raspberry Pi 2 Model B Rev 1.1

           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 peripherals
 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>
```

We're Finished

There's still some work to do to get things sorted. We need to check the `/etc/modules` file using:

```
sudo nano /etc/modules
```

Where we need to ensure that the following line is at the end of the file:

```
i2c-dev
```

Under some circumstances (depending on the kernel version we are using) we would also need to update the /boot/config.txt file. We can do this using;

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

Make sure that the following line is uncommented (the '#' is removed from in front of the line) in the file;

```
dtparam=i2c_arm=on
```

The we should load tools for working with I2C devices using the following command;

```
sudo apt-get install i2c-tools
```

... and now we should reboot to load the config.txt if we changed it earlier

```
sudo reboot
```

We can now check to see if our sensor is working using;

```
sudo i2cdetect -y 1
```

If we were using an older B model of Raspberry Pi with 256MB of RAM, we would need to use `sudo i2cdetect -y 0`.

The output should look something like;

```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

This shows us that we have detected our ADS1015 on address '48'. The ADS1015 can support four different addresses as shown on page 17 of the data sheet[37]. The address is selected by what the ADDR (short for **addr**ess!) pin on the board is connected to;

ADDR PIN	ADDRESS
Ground	48
VDD	49
SDA	4A
SCL	4B

As we noted earlier, this means we can connect up to four ADS1015's on the same I2C bus.

Now we want to install Python libraries designed to read the values from the ADS1015. The library we are going to use was designed specifically to work with the Adafruit ADS1015/ADS1115 ADCs[38]. In carrying out this library development, Adafruit have invested a not inconsiderable amount of time and resources. In return please consider supporting Adafruit and open-source hardware by purchasing products from Adafruit[39]!

```
sudo apt-get install git build-essential python-dev python-smbus
```

Now we create a directory that we will use to download the Adafruit Python library (assuming that we're starting from our home directory);

```
mkdir adc
cd adc
```

Now we will download the library into the `adc` directory (the following command retrieves the library from the github site);

[37]http://www.adafruit.com/datasheets/ads1015.pdf
[38]http://www.adafruit.com/product/1083
[39]https://www.adafruit.com/

```
git clone https://github.com/adafruit/Adafruit_Python_ADS1X15
```

Now that we've downloaded it, we can set it up. Run the following commands;

```
cd Adafruit_Python_ADS1x15
sudo python setup.py install
```

This will download and extract the tools. Then we can change into the examples directory

```
cd examples
```

Now we can run the example program `simpletest.py` as follows;

```
python simpletest.py
```

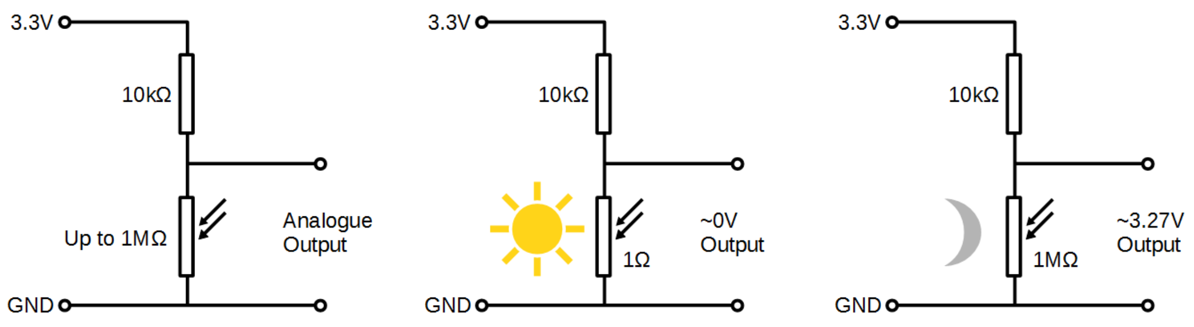
This will start a program that will present a continuous reading of the four analog channels connected to the ADC.

```
Reading ADS1x15 values, press Ctrl-C to quit...
|      0 |      1 |      2 |      3 |
-------------------------------------
|   5920 |   4704 |   4608 |   4608 |
|   5920 |   4576 |   4608 |   4656 |
|   7568 |   4640 |   4592 |   4656 |
|  14368 |   4656 |   4608 |   4672 |
|  15152 |   4656 |   4608 |   4656 |
|  14160 |   4608 |   4592 |   4672 |
|  13696 |   4608 |   4624 |   4640 |
|  14016 |   4608 |   4608 |   4704 |
|   8384 |   4608 |   4624 |   4672 |
|   4560 |   4592 |   4624 |   4656 |
```

If we move the LDR about we should see that channel `0` varies up and down as the amount of light it receives varies. Success!

We should see that when the sensor is exposed to a stronger light the value decreases and when it gets darker the value increases.

If we remember back to our earlier diagram showing the type of connection this helps put the changes into context



LDR Sensor Output Voltage

We can press Ctrl-C to stop the program.

At this point we have successfully read and displayed an analog signal from a sensor using the Raspberry Pi.

Record

To record this data we will use a Python program that connects to our sensor via the ADC, reads the returned value and writes that value into our database. At the same time a time stamp will be added automatically.

Our Python program will use `cron` to execute the program at a regular intervals (We used this earlier to automatically reconnect to the network if required.).

Record the readings

The following Python code simply reads channel 0 of our ADC and writes the value for time and light into our database.

The full code can be found in the code samples bundled with this book (light-record.py).

```
#!/usr/bin/python
#encoding:utf-8

import time                 #Import time library
import sqlite3              #Import SQLite library
import Adafruit_ADS1x15     #Import ADS1x15 library

print "Light sensor measurement in progress"

# Create an ADS1015 ADC (12-bit) instance.
adc = Adafruit_ADS1x15.ADS1015()

# Choose a gain of 1 for reading voltages from 0 to 4.09V.
GAIN = 1

# Read ADC channel 0 using the gain value.
ldr = adc.read_adc(0, gain=GAIN)

# Get the time in the right format
dtg = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())

# Print the time and LDR values
print "Local current time :", dtg
print "LDR Value via ADC :", ldr

# Write the values to the database
try:
    # Opens a file called measurements
    db = sqlite3.connect('/home/pi/measurements')
    # Get a cursor object
    cursor = db.cursor()
    # Insers the values into the table
    cursor.execute('''INSERT INTO light(dtg, ldr)
                 VALUES(?,?)''', (dtg,ldr))
    # Commit the change
    db.commit()
# Catch any exception
except Exception as e:
    # Roll back any change if something goes horribly wrong
    db.rollback()
    raise e
finally:
    # Close the db connection
    db.close()
```

This script can be saved in our home directory (`/home/pi`) and can be run by typing;

```
python light-record.py
```

The output should look something like this;

```
Light sensor measurement in progress
Local current time : 2018-04-15 09:31:23
LDR Value via ADC : 176
```

Run this script a few times and then we can check the results in our database by starting up SQLite as follows;

```
sqlite3 measurements
```

From the SQLite prompt we can query the database to return all the records using `SELECT * FROM light;` as follows;

```
sqlite> SELECT * FROM light;
2018-04-15 08:39:23|221
2018-04-15 08:41:34|56
2018-04-15 08:53:18|158
sqlite>
```

There are three records, including the times they were taken and the light levels recorded.

Recording data on a regular basis with `cron`

As mentioned earlier, while our code is a thing of beauty, it only records a single entry for the light every time it is run.

What we need to implement is a schedule so that at a regular time, the program is run. This is achieved using `cron` via the `crontab`.

To set up our schedule we need to edit the `crontab` file. This is is done using the following command;

```
crontab -e
```

Once run it will open the crontab in the nano editor. We want to add in an entry at the end of the file that looks like the following;

```
* * * * * /usr/bin/python /home/pi/light-record.py
```

This instructs the computer that every minute, every hour of every day of every month we run the command `/usr/bin/python /home/pi/light-record.py` (which, if we were at the command line in the pi home directory, we would run as `python light-record.py`, but since we can't guarantee where we will be when running the script, we are supplying the full path to the `python` command and the `light-record.py` script.

Save the file and when the next hour rolls over our program will run on its designated schedule and we will have light values written to our database every minute. After a while we can check it out by running a `SELECT * FROM light;` on the measurement database.

```
sqlite> SELECT * FROM distance;
2018-04-15 08:39:23|221
2018-04-15 08:41:34|56
2018-04-15 08:53:18|158
2018-04-15 10:00:06|159
2018-04-15 10:01:06|160
2018-04-15 11:02:06|159
sqlite>
```

Managing database size

While it's a great idea to save our local data into a database, we stand the risk of gradually letting that database fill up until it exceeds the capacity of our storage.

In the case of the measurements that we are carrying out, the readings are happening pretty regularly, so it's worth thinking about. Capturing some simple measurements every minute means in the scheme of things that's about 10,000 recordings per week.

What we're looking for is a script that will run on a repeating schedule and remove old records. Sound familiar? That's a very similar process to what we are doing when we record our data. A python script that is executed regularly by `cron`.

Here's how we can do it.

The following python script (which we can name `db-manage.py`) opens our database, deletes any records older than a year, cleans up and exits.

```
#!/usr/bin/python
#encoding:utf-8

#Import SQLite library
import sqlite3

# Opens a database file called measurements
conn = sqlite3.connect('/home/pi/measurements', isolation_level=None)
db = conn.cursor()

# Delete any records that are older than 1 year
db.execute('DELETE FROM light WHERE dtg<DATETIME("now","localtime", "-1 years")')
# VACUUM the database to remove any unnecessary data
db.execute('VACUUM')

# Commit the changes to the database and close the connection
conn.commit()
conn.close
```

The file is available as `db-manage.py` and can be found in the code sample extras that can be downloaded with this book.

It's a pretty simple script and we can schedule its operation by editing the `crontab` file like so;

```
crontab -e
```

We want to add in an entry at the end of the file that looks like the following;

```
1 0 */1 * * /usr/bin/python /home/pi/db-manage.py
```

This instructs the computer that at 1 minute past the hour at midnight (hence the `0`) on the 1st day of every month we run the command `/usr/bin/python /home/pi/db-manage.py` (which, if we were at the command line in the pi home directory, we would run as `python db-manage.py`, but since we can't guarantee where we will be when running the script, we are supplying the full path to the `python` command and the `db-manage.py` script.

Save the file and every month our program will run on its designated schedule and will make sure to delete any records older than a year.

Explore

Simple data point API

The main mechanism for exploring and using our data is going to be via a simple data block returned from a http request.

What does all that actually mean?

That's a good question. Ultimately we're measuring something and we want to be able to communicate that measurement to an external service. That service could be another database somewhere or to a web page or to a system that will alert based on the value of the light levels being within certain boundaries.

The very simplest way that we can do this is to present the data as the measured values when we ask for them in a web request. This could be thought of as a simplified form of an API (and I plan to make something more complicated in the future).

Technically, 'API' stands for Application Programming Interface. Think of it as a software program that allows a consistent transfer of information between two computers.

The data will be presented as JSON as that is one of the most ubiquitous data forms around.

Comma separated values and tab separated values are fairly well understood forms of data. They are expressed as rows and columns of information that are separated using a known character. While these forms of data are simple to understand, it is not easy to incorporate a hierarchy structure to the data, and when you try, it isn't natural and makes managing the data difficult.

JavaScript Object Notation (JSON) presents a different mechanism for storing data. A light weight description could read "*JSON is a text-based open standard designed to present human-readable data. It is derived from the JavaScript scripting language, but it is language and platform independent.*"

Enough esoterics, what does the magic code look like that will do this?

```
<?php

$db = new PDO('sqlite://home/pi/measurements');

$result = $db->query('SELECT * FROM light ORDER BY dtg DESC LIMIT 1');

$datapie = array();

$result->setFetchMode(PDO::FETCH_ASSOC);

while ($row = $result->fetch()) {
    extract($row);
    echo json_encode($row);
}

?>
```

We can save this file as `light.php` and have it in the `/var/www/html` directory on our Pi (`light.php` can be found in the code sample extras that can be downloaded with this book).

How we can put in the IP address of our Pi to our browser along with our distance php file (`http://10.1.1.120/light.php`) and we should get something like the following appear in the browser;

```
{"dtg":"2018-03-24 11:01:06","ldr":"159"}
```

What good will getting this data be? Well……. I'm a bit of a believer that the information that gets captured by the Pi shouldn't ultimately reside on the device in the long term. In the perfect world I would see it being requested by an external service that was checking a range of data points that would exist around the home (pressure, temperature inside / outside, CO2 levels, is the car parked in the garage, that sort of thing) so this is more of an enabling device than a 'let's display stuff' deal. But I hear what you're saying. "That's lame. How can I impress people with that?". Fair point. To deal with that problem let's make a simple graph.

Extracting a Range of Data

Righto… If we're going to make a graph of our light levels we'll need a variation of our API that will gather and present a range of data that our graph can then display.

This will form a piece of code that our graph will use as a JSON formatted data source.

It will look as follows;

```
<?php

$data= array();

// connect to the database
$db = new PDO("sqlite://home/pi/measurements");
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// prepare SQL command and execute
$query = "SELECT * FROM light WHERE
          dtg>DATETIME('now','localtime', '-24 hours')";
$result = $db->prepare( $query );
$result->execute();

// compile the returned data
$values = $result->fetchAll(PDO::FETCH_ASSOC);
array_push($data, $values);

// print the data
echo json_encode($values);

// close the database connection
$db = NULL;

?>
```

This block of PHP code will connect to our database and instead of returning a single piece of data it will return ('echo') a range of values from the past 24 hours. We'll call the file `light-range.php` and it will be in the `/var/www/html` directory. A copy of the file can be found in the code sample extras that can be downloaded with this book.

The following file is our graph which will use our `light-range.php` file and display it. It uses the d3.js visualisation library and for a full description of the workings of the code please feel free to consult a copy of 'D3 Tips and Tricks v4.x'. It's free and can be downloaded from here[40].

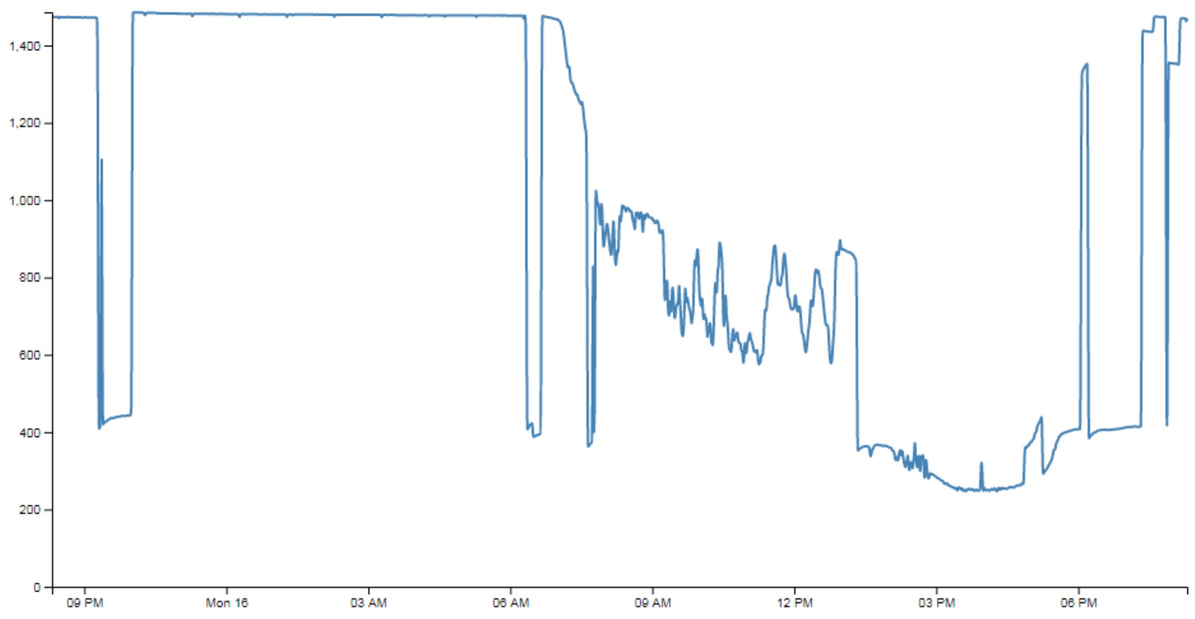
[40]https://leanpub.com/d3-t-and-t-v4

```
<!DOCTYPE html>
<meta charset="utf-8">
<style> /* set the CSS */

.line {
  fill: none;
  stroke: steelblue;
  stroke-width: 2px;
}

</style>
<body>

<!-- load the d3.js library -->
<script src="https://d3js.org/d3.v4.min.js"></script>
<script>

// set the dimensions and margins of the graph
var margin = {top: 20, right: 20, bottom: 30, left: 50},
    width = 960 - margin.left - margin.right,
    height = 500 - margin.top - margin.bottom;

// parse the date / time
var parseTime = d3.timeParse("%Y-%m-%d %H:%M:%S");

// set the ranges
var x = d3.scaleTime().range([0, width]);
var y = d3.scaleLinear().range([height, 0]);

// define the line
var valueline = d3.line()
    .curve(d3.curveBasis)
    .x(function(d) { return x(d.date); })
    .y(function(d) { return y(d.close); });

// append the svg obgect to the body of the page
// appends a 'group' element to 'svg'
// moves the 'group' element to the top left margin
var svg = d3.select("body").append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
  .append("g")
    .attr("transform",
          "translate(" + margin.left + "," + margin.top + ")");

// Get the data
```

```
d3.json("light-range.php", function(error, data) {
  if (error) throw error;

  // format the data
  data.forEach(function(d) {
      d.date = parseTime(d.dtg);
      d.close = +d.ldr;
  });

  // Scale the range of the data
  x.domain(d3.extent(data, function(d) { return d.date; }));
  y.domain([0, d3.max(data, function(d) { return d.close; })]);

  // Add the valueline path.
  svg.append("path")
      .data([data])
      .attr("class", "line")
      .attr("d", valueline);

  // Add the X Axis
  svg.append("g")
      .attr("transform", "translate(0," + height + ")")
      .call(d3.axisBottom(x));

  // Add the Y Axis
  svg.append("g")
      .call(d3.axisLeft(y));

});

</script>
</body>
```

We will want to place a copy of this file which we will call `light-graph.html` in the `/var/www/html` directory. A copy of it can be found in the code sample extras that can be downloaded with this book.

We can see the end result by putting the web address into our browser. It should look something like `http://10.1.1.120/light-graph.html`. The end result should look a bit like the following;



Light Levels Graph

Wrap Up

There we have it.

We've assembled our Raspberry Pi with an analog light sensor and an analog to digital converter. We installed an operating system and configured it for use. We've set up networking and installed a database and a web server. We've written code to record data into our database and an API to pull data out of it. We've even installed a graph to display it in a visual form. Nice work.

There is a strong possibility that the information I have laid out here could be littered with evil practices and gross inaccuracies.

But look on the bright side. Irrespective of the nastiness of the way that any of it was accomplished or the inelegance of the code, if the picture drawn on the screen is relatively pretty, you can walk away with a smile. :-)

Those with a smattering of knowledge of any of the topics I have butchered above (or below) are fully justified in feeling a large degree of righteous indignation. To those I say, please feel free to amend where practical and possible, but please bear in mind this was written from the point of view of someone with only a little experience in the topic and therefore try to keep any instructions at a level where a new entrant can step in.

Bibliography

The following texts were incredibly useful in drafting up this project.

RPi and I2C Analog-Digital Converter (OpenLabTools[41])

[41]http://openlabtools.eng.cam.ac.uk/Resources/Datalog/RPi_ADS1115/

ADS1015 12-Bit ADC - 4 Channel with Programmable Gain Amplifier(Adafruit[42])

ADS1015 datasheet (Adafruit[43])

Adafruit 4-Channel ADC Breakouts (Adafruit Learning System[44])

Analog Sensors On The Raspberry Pi Using An MCP3008 (Matt, raspberrypi-spy.co.uk[45])

Analog Input Board for the Espiresso Pressure Sensor (int03.co.uk[46])

Arduino KY-018 Photo resistor module (TkkrLab[47])

Shenzhen KEYES DIY Robot co., Ltd[48]

Light dependant resister datasheet (Sunroom Technologies[49])

---

[42]http://www.adafruit.com/products/1083
[43]http://www.adafruit.com/datasheets/ads1015.pdf
[44]https://learn.adafruit.com/adafruit-4-channel-adc-breakouts?view=all
[45]http://www.raspberrypi-spy.co.uk/2013/10/analogue-sensors-on-the-raspberry-pi-using-an-mcp3008/
[46]http://int03.co.uk/http://int03.co.uk/blog/2014/12/17/analogue-input-board-for-the-pressure-sensor-espiresso/
[47]https://tkkrlab.nl/wiki/Arduino_KY-018_Photo_resistor_module
[48]http://en.keyes-robot.com/index.aspx
[49]http://kennarar.vma.is/thor/v2011/vgr402/ldr.pdf

Linux Concepts

What is Linux?

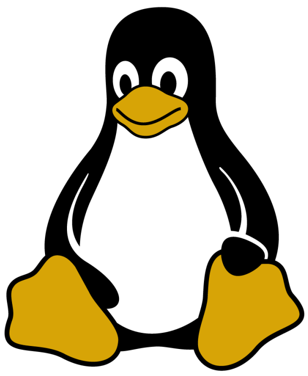
In it's simplest form, the answer to the question "What is Linux?" is that it's a computer operating system. As such it is the software that forms a base that allows applications that run on that operating system to run.

In the strictest way of speaking, the term 'Linux' refers to the Linux kernel. That is to say the central core of the operating system, but the term is often used to describe the set of programs, tools, and services that are bundled together with the Linux kernel to provide a fully functional operating system.

An operating system is software that manages computer hardware and software resources for computer applications. For example Microsoft Windows could be the operating system that will allow the browser application Firefox to run on our desktop computer.

Linux[50] is a computer operating system that is can be distributed as free and open-source software[51]. The defining component of Linux is the Linux kernel, an operating system kernel first released on 5 October 1991 by Linus Torvalds.

Linux was originally developed as a free operating system for Intel x86-based personal computers. It has since been made available to a huge range of computer hardware platforms and is a leading operating system on servers, mainframe computers and supercomputers. Linux also runs on embedded systems, which are devices whose operating system is typically built into the firmware and is highly tailored to the system; this includes mobile phones, tablet computers, network routers, facility automation controls, televisions and video game consoles. Android, the most widely used operating system for tablets and smart-phones, is built on top of the Linux kernel.



The Linux mascot 'Tux'

The development of Linux is one of the most prominent examples of free and open-source software collaboration. Typically, Linux is packaged in a form known as a Linux distribution, for

[50] http://en.wikipedia.org/wiki/Linux
[51] http://en.wikipedia.org/wiki/Free_and_open-source_software

both desktop and server use. Popular mainstream Linux distributions include Debian, Ubuntu and the commercial Red Hat Enterprise Linux. Linux distributions include the Linux kernel, supporting utilities and libraries and usually a large amount of application software to carry out the distribution's intended use.

A distribution intended to run as a server may omit all graphical desktop environments from the standard install, and instead include other software to set up and operate a solution stack such as LAMP (Linux, Apache, MySQL and PHP). Because Linux is freely re-distributable, anyone may create a distribution for any intended use.

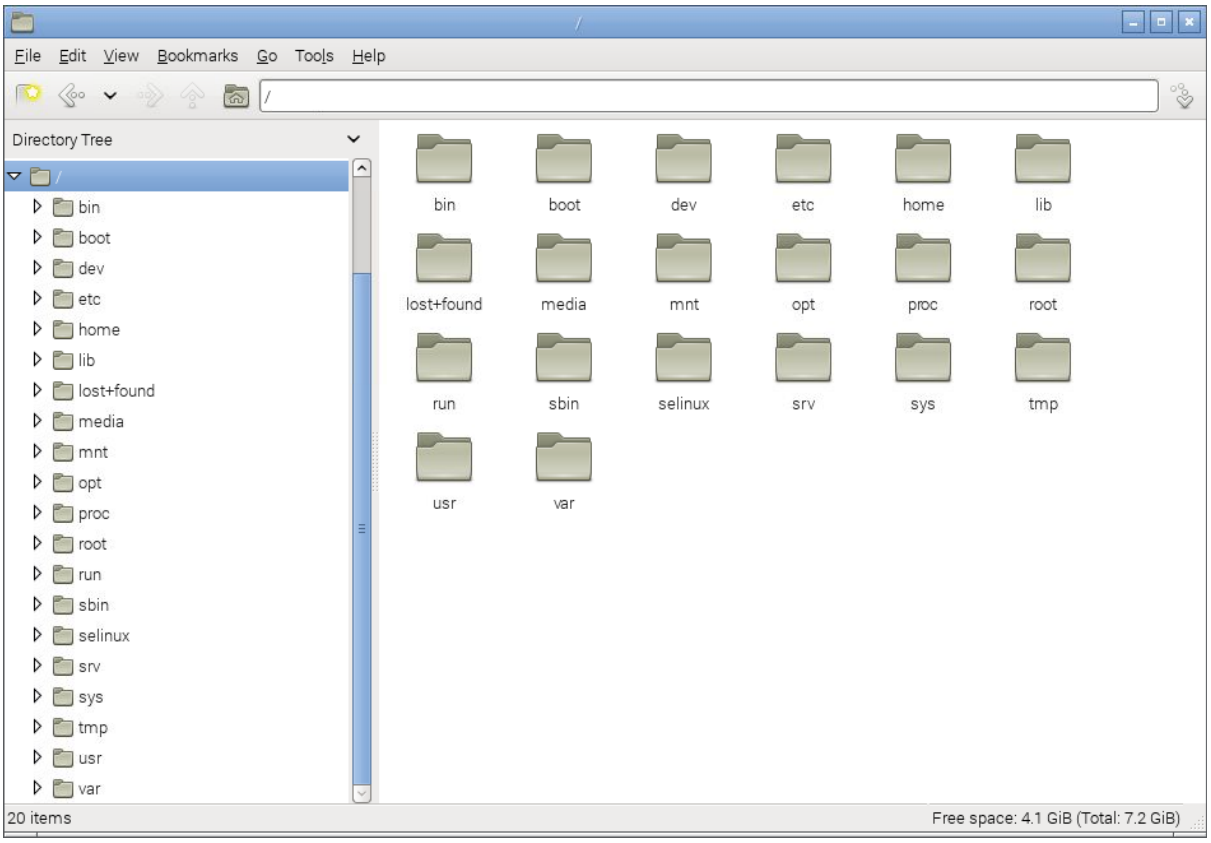
Linux is not an operating system that people will typically use on their desktop computers at home and as such, regular computer users can find the barrier to entry for using Linux high. This is made easier through the use of Graphical User Interfaces that are included with many Linux distributions, but these graphical overlays are something of a shim to the underlying workings of the computer. There is a greater degree of control and flexibility to be gained by working with Linux at what is called the 'Command Line' (or CLI), and the booming field of educational computer elements such as the Raspberry Pi[52] have provided access to a new world of learning opportunities at this more fundamental level.

[52]http://raspberrypi.org/

Linux Directory Structure

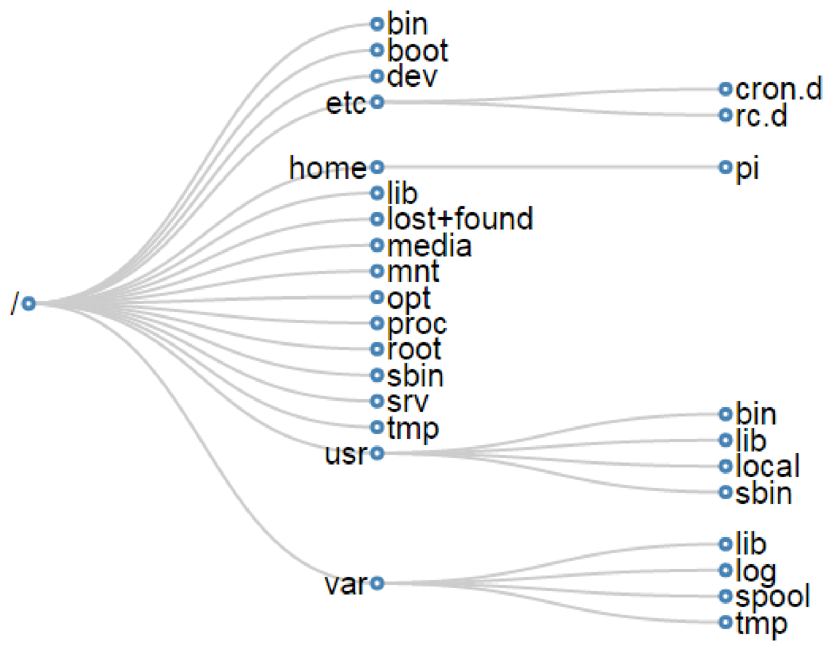
To a new user of Linux, the file structure may feel like something at best arcane and in some cases arbitrary. Of course this isn't entirely the case and in spite of some distribution specific differences, there is a fairly well laid out hierarchy of directories and files with a good reason for being where they are.

We are frequently comfortable with the concept of navigating this structure using a graphical interface similar to that shown below, but to operate effectively at the command line we need to have a working knowledge of what goes where.



Linux Directories

The directories we are going to describe form a hierarchy similar to the following;



Directory Hierarchy

For a concise description of the directory functions check out the cheat sheet. Alternatively their function and descriptions are as follows;

`/`

The `/` or 'root' directory contains all other files and directories. It is important to note that this is **not** the root users home directory (although it used to be many years ago). The root user's home directory is `/root`. Only the root user has write privileges for this directory.

`/bin`

The `/bin` directory contains common essential binary executables / commands for use by all users. For example: the commands cd, cp, ls and ping. These are commands that may be used by both the system administrator and by users, but which are required when no other filesystems are mounted.

`/boot`

The `/boot` directory contains the files needed to successfully start the computer during the boot process. As such the `/boot` directory contains information that is accessed *before* the Linux kernel begins running the programs and process that allow the operating system to function.

`/dev`

The `/dev` directory holds device files that represent physical devices attached to the computer such as hard drives, sound devices and communication ports as well as 'logical' devices such as a random number generator and `/dev/null` which will essentially discard any information sent to it. This directory holds a range of files that strongly reinforces the Linux precept that *Everything is a file*.

`/etc`

The `/etc` directory contains configuration files that control the operation of programs. It also contains scripts used to startup and shutdown individual programs.

`/etc/cron.d`

The `/etc/cron.d`, `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly`, `/etc/cron.monthly` directories contain scripts which are executed on a regular schedule by the crontab process.

`/etc/rc?.d`

The `/rc0.d`, `/rc1.d`, `/rc2.d`, `/rc3.d`, `/rc4.d`, `/rc5.d`, `/rc6.d`, `/rcS.d` directories contain the files required to control system services and configure the mode of operation (runlevel) for the computer.

`/home`

Because Linux is an operating system that is a 'multi-user' environment, each user requires a space to store information specific to them. This is done via the `/home` directory. For example, the user 'pi' would have `/home/pi` as their home directory.

`/lib`

The `/lib` directory contains shared library files that supports the executable files located under `/bin` and `/sbin`. It also holds the kernel modules (drivers) responsible for giving Linux a great deal of versatility to add or remove functionality as needs dictate.

`/lost+found`

The `/lost+found` directory will contain potentially recoverable data that might be produced if the file system undergoes an improper shut-down due to a crash or power failure. The data recovered is unlikely to be complete or undamaged, but in some circumstances it may hold useful information or pointers to the reason for the improper shut-down.

`/media`

The `/media` directory is used as a directory to temporarily mount removable devices (for example, `/media/cdrom` or `/media/cdrecorder`). This is a relatively new development for Linux and comes as a result of a degree of historical confusion over where was best to mount these types of devices (`/cdrom`, `/mnt` or `/mnt/cdrom` for example).

`/mnt`

The `/mnt` directory is used as a generic **mou**n**t** point for filesystems or devices. Recent use of the directory is directing it towards it being used as a temporary mount point for system administrators, but there is a degree of historical variation that has resulted in different distributions doing things different ways (for example, Debian allocates `/floppy` and `/cdrom` as mount points while Redhat places them in `/mnt/floppy` and `/mnt/cdrom` respectively).

`/opt`

The `/opt` directory is used for the installation of third party or additional **opt**ional software that is not part of the default installation. Any applications installed in this area should be installed in such a way that it conforms to a reasonable structure and should not install files outside the `/opt` directory.

`/proc`

The `/proc` directory holds files that contain information about running **proc**esses and system resources. It can be described as a pseudo filesystem in the sense that it contains runtime system information, but not 'real' files in the normal sense of the word. For example the `/proc/cpuinfo` file which contains information about the computers cpus is listed as `0` bytes in length and yet if it is listed it will produce a description of the cpus in use.

`/root`

The `/root` directory is the home directory of the System Administrator, or the 'root' user. This could be viewed as slightly confusing as all other users home directories are in the `/home` directory and there is already a directory referred to as the 'root' directory (`/`). However, rest assured that there is good reason for doing this (sometimes the `/home` directory could be mounted on a separate file system that has to be accessed as a remote share).

`/sbin`

The `/sbin` directory is similar to the `/bin` directory in the sense that it holds binary executables / commands, but the ones in `/sbin` are essential to the working of the operating system and are identified as being those that the system administrator would use in maintaining the system. Examples of these commands are fdisk, shutdown, ifconfig and modprobe.

`/srv`

The `/srv` directory is set aside to provide a location for storing data for specific services. The rationale behind using this directory is that processes or services which require a single location and directory hierarchy for data and scripts can have a consistent placement across systems.

`/tmp`

The `/tmp` directory is set aside as a location where programs or users that require a temporary location for storing files or data can do so on the understanding that when a system is rebooted or shut down, this location is cleared and the contents deleted.

`/usr`

The `/usr` directory serves as a directory where user programs and data are stored and shared. This potential wide range of files and information can make the `/usr` directory fairly large and complex, so it contains several subdirectories that mirror those in the root (`/`) directory to make organisation more consistent.

`/usr/bin`

The `/usr/bin` directory contains binary executable files for users. The distinction between `/bin` and `/usr/bin` is that `/bin` contains the essential commands required to operate the system even if no other file system is mounted and `/usr/bin` contains the programs that users will require to do normal tasks. For example; `awk`, `curl`, `php`, `python`. If you can't find a user binary under `/bin`, look under `/usr/bin`.

`/usr/lib`

The `/usr/lib` directory is the equivalent of the `/lib` directory in that it contains shared library files that supports the executable files for users located under `/usr/bin` and `/usr/sbin`.

`/usr/local`

The `/usr/local` directory contains users programs that are installed locally from source code. It is placed here specifically to avoid being inadvertently overwritten if the system software is upgraded.

`/usr/sbin`

The `/usr/sbin` directory contains non-essential binary executables which are used by the system administrator. For example `cron` and `useradd`. If you can't locate a system binary in `/usr/sbin`, try `/sbin`.

`/var`

The `/var` directory contains variable data files. These are files that are expected to grow under normal circumstances For example, log files or spool directories for printer queues.

`/var/lib`

The `/var/lib` directory holds dynamic state information that programs typically modify while they run. This can be used to preserve the state of an application between reboots or even to share state information between different instances of the same application.

`/var/log`

The `/var/log` directory holds log files from a range of programs and services. Files in `/var/log` can often grow quite large and care should be taken to ensure that the size of the directory is managed appropriately. This can be done with the `logrotate` program.

`/var/spool`

The `/var/spool` directory contains what are called ‘spool’ files that contain data stored for later processing. For example, printers which will queue print jobs in a spool file for eventual printing and then deletion when the resource (the printer) becomes available.

`/var/tmp`

The `/var/tmp` directory is a temporary store for data that needs to be held between reboots (unlike `/tmp`).

Everything is a file in Linux

A phrase that will often come up in Linux conversation is that;

> *Everything is a file*

For someone new to Linux this sounds like some sort of 'in joke' that is designed to scare off the unwary and it can sometimes act as a barrier to a deeper understanding of the philosophy behind the approach taken in developing Linux.

The explanation behind the statement is that Linux is designed to be a system built of a group of interacting parts and the way that those parts can work together is to communicate using a common method. That method is to use a file as a common building block and the data in a file as the communications mechanism.

The trick to understanding what '*Everything is a file*' means, is to broaden our understanding of what a file can be.

Traditional Files

The traditional concept of a file is an object with a specific name in a specific location with a particular content. For example, we might have a file named `foo.txt` which is in the directory `/home/pi/` and it could contain a couple of lines of text similar to the following;

```
This is the first line
This is the second line
```

Directories

As unusual as it sounds a directory is also a file. The special aspect of a directory is that is is a file which contains a list of information about which files (and / or subdirectories) it contains. So when we want to list the contents of a directory using the `ls` command what is actually happening is that the operating system is getting the appropriate information from the file that represents the directory.

System Information

However, files can also be conduits of information. The `/proc/` directory contains files that represent system and process information. If we want to determine information about the type of CPU that the computer is using, the file `cpuinfo` in the `/proc/` directory can list it. By running the command 'cat /proc/cpuinfo' we can list a wealth of information about our CPU (the following is a subset of that information by the way);

```
pi@raspberrypi ~ $ cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 57.60
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt v\
fpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5

Hardware        : BCM2709
Revision        : a01041
Serial          : 000000002a4ea712
```

Now that might not mean a lot to us at this stage, but if we were writing a program that needed a particular type of CPU in order to run successfully it could check this file to ensure that it could operate successfully. There are a wide range of files in the `/proc/` directory that represent a great deal of information about how our system is operating.

Devices

When we use different devices in a Linux operating system these are also represented as a file. In the `/dev/` directory we have files that represent a range of physical devices that are part of our computer. In larger computer systems with multiple disks they could be represented as `/dev/sda1` and `/dev/sda2`, so that when we wanted to perform an action such as formatting a drive we would use the command `mkfs` on the `/dev/sda1` file.

The `/dev/` directory also holds some curious files that are used as tools for generating or managing data. For example `/dev/random` is an interface to the kernels random number device. `/dev/zero` represents a file that will constantly stream zeros (while this might sound weird, imagine a situation where you want to write over an area of disk with data to erase it). The most well known of these unusual files is probably `/dev/null`[53]. This will act as a 'null device' that will essentially discard any information sent to it.

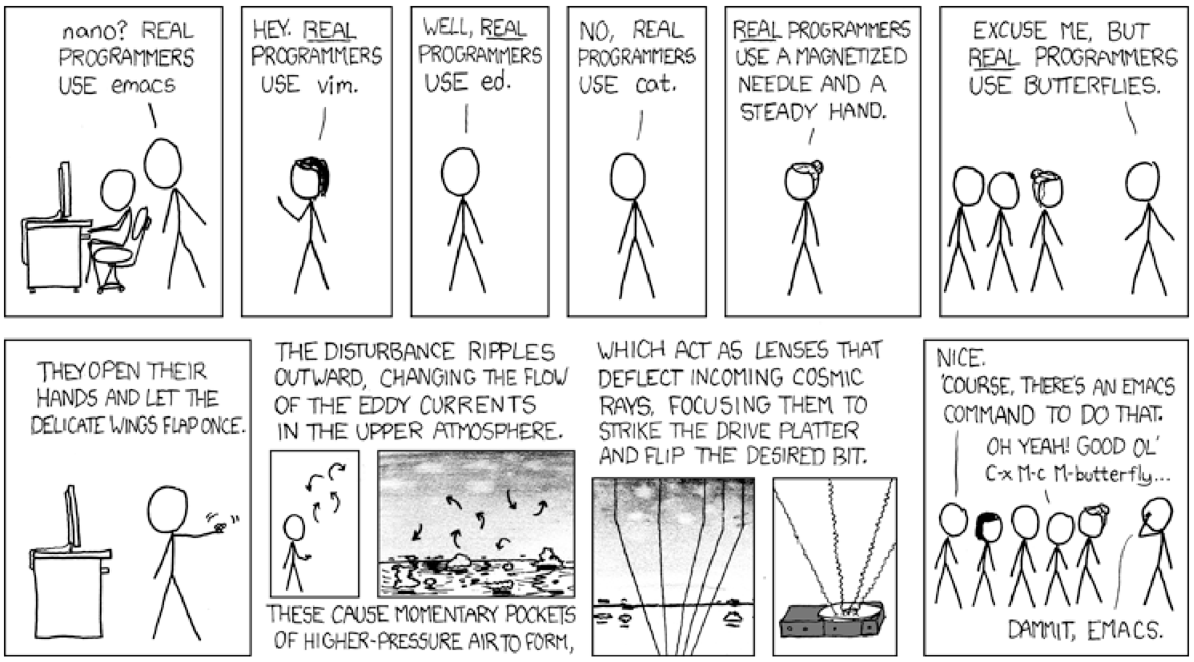
[53]https://en.wikipedia.org/wiki/Null_device

File Editing

Working in Linux is an exercise in understanding the concepts that Linux uses as its foundations such as 'Everything is a file' and the use of wildcards, pipes and the directory structure.

While working at the command line there will very quickly come the realisation that there is a need to know how to edit a file. Linux being what it is, there are many ways that files can be edited.

An outstanding illustration of this is via the excellent cartoon work of the xkcd comic strip[54] (Buy his stuff[55], it's awesome!).



nano? REAL PROGRAMMERS USE emacs
HEY. REAL PROGRAMMERS USE vim.
WELL, REAL PROGRAMMERS USE ed.
NO, REAL PROGRAMMERS USE cat.
REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.
EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.
THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.
THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.
THESE CAUSE MOMENTARY POCKETS OF HIGHER-PRESSURE AIR TO FORM,
WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.
NICE.
'COURSE, THERE'S AN EMACS COMMAND TO DO THAT.
OH YEAH! GOOD OL' C-x M-c M-butterfly...
DAMMIT, EMACS.

Real Programmers

For a taste of the possible options available Wikipedia[56] has got our back. Inevitably where there is choice there are preferences and where there are preferences there is bias. Everyone will have a preference towards a particular editor and don't let a particular bias influence you to go down a particular direction without considering your options. Speaking from personal experience I was encouraged to use 'vi' as it represented the preference of the group I was in, but because I was a late starter to the command line I struggled for the longest time to try and become familiar with it. I know I should have tried harder, but I failed. For a while I wandered in the editor wilderness trying desperately to cling to the GUI where I could use 'gedit' or 'geany' and then one day I was introduced to 'nano'.

This has become my preference and I am therefore biased towards it. Don't take my word for it. Try alternatives. I'll describe 'nano' below, but take that as a possible path and realise that

[54]http://xkcd.com/378/
[55]http://store.xkcd.com/
[56]https://en.wikipedia.org/wiki/List_of_text_editors

whatever editor works for you will be the right one. The trick is simply to find one that works for you.

The nano Editor

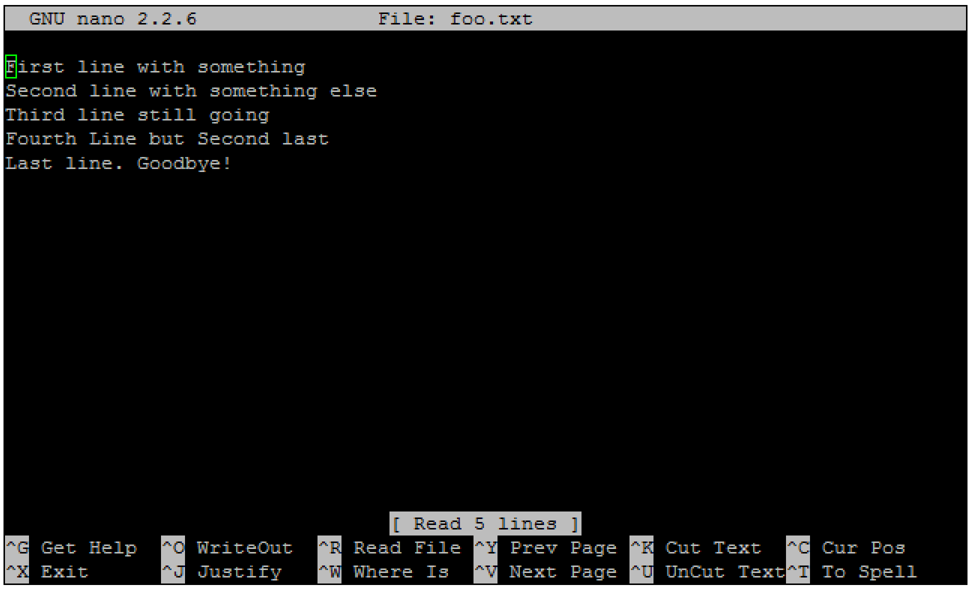
The `nano` editor can be started from the command line using just the command and the /path/name of the file.

```
nano foo.txt
```

If the file requires administrator permissions it can be executed with ‘sudo‘.

```
sudo nano foo.txt
```

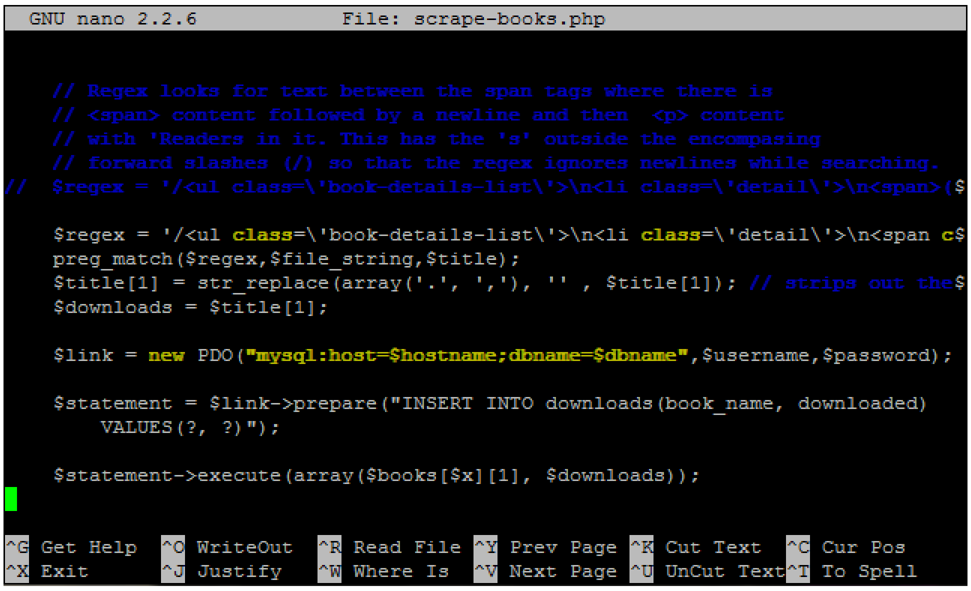
When it opens it presents us with a working space and part of the file and some common shortcuts for use at the bottom of the console;



```
  GNU nano 2.2.6                File: foo.txt

First line with something
Second line with something else
Third line still going
Fourth Line but Second last
Last line. Goodbye!

                                  [ Read 5 lines ]
^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

nano Interface

It includes some simple syntax highlighting for common file formats;



```
  GNU nano 2.2.6              File: scrape-books.php


    // Regex looks for text between the span tags where there is
    // <span> content followed by a newline and then  <p> content
    // with 'Readers in it. This has the 's' outside the encompasing
    // forward slashes (/) so that the regex ignores newlines while searching.
//  $regex = '/<ul class=\'book-details-list\'>\n<li class=\'detail\'>\n<span>($

    $regex = '/<ul class=\'book-details-list\'>\n<li class=\'detail\'>\n<span c$
    preg_match($regex,$file_string,$title);
    $title[1] = str_replace(array('.', ','), '' , $title[1]); // strips out the$
    $downloads = $title[1];

    $link = new PDO("mysql:host=$hostname;dbname=$dbname",$username,$password);

    $statement = $link->prepare("INSERT INTO downloads(book_name, downloaded)
        VALUES(?, ?)");

    $statement->execute(array($books[$x][1], $downloads));


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

nano Syntax Highlighting

This can be improved if desired (cue Google).

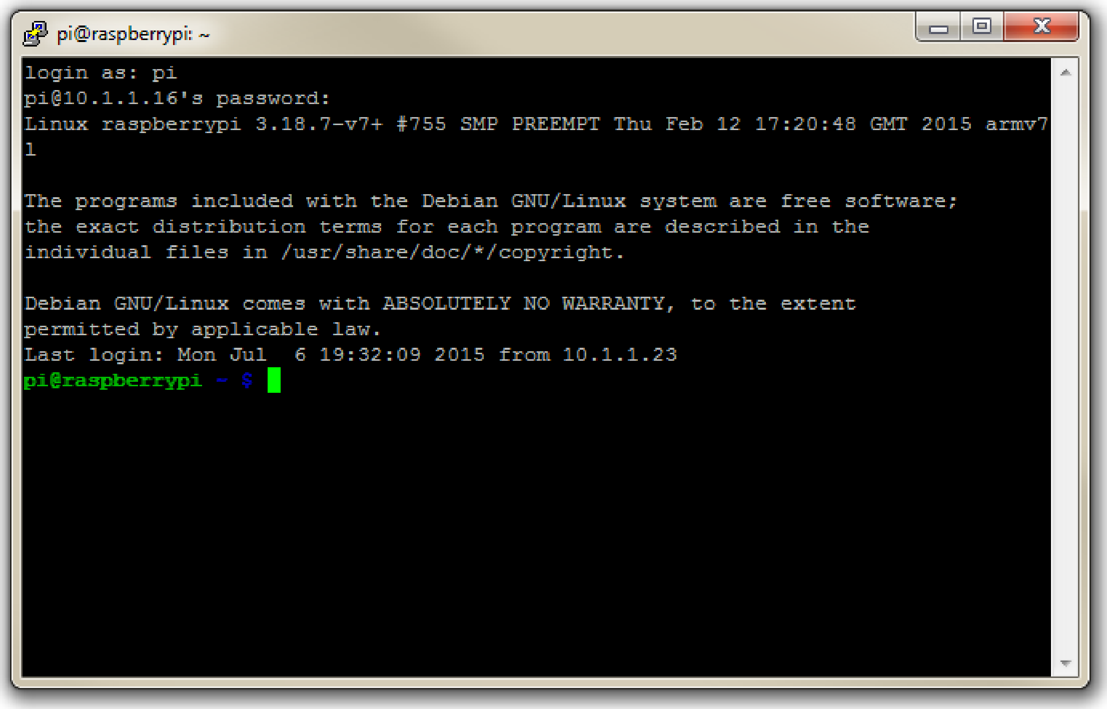
There is a swag of shortcuts to make editing easier, but the simple ones are as follows;

- CTRL-x - Exit the editor. If we are in the middle of editing a file we will be asked if we want to save our work
- CTRL-r - Read a file into our current working file. This enables us to add text from another file while working from within a new file.
- CTRL-k - Cut text.
- CTRL-u - Uncut (or Paste) text.
- CTRL-o - Save file name and continue working.
- CTRL-t - Check the spelling of our text.
- CTRL-w - Search the text.
- CTRL-a - Go to the beginning of the current working line.
- CTRL-e - Go to the end of the current working line.
- CTRL-g - Get help with nano.

Linux Commands

Executing Commands in Linux

A command is an instruction given by a user telling the computer to carry out an action. This could be to run a single program or a group of linked programs. Commands are typically initiated by typing them in at the command line (in a terminal) and then pressing the ENTER key, which passes them to the shell.



```
login as: pi
pi@10.1.1.16's password:
Linux raspberrypi 3.18.7-v7+ #755 SMP PREEMPT Thu Feb 12 17:20:48 GMT 2015 armv7
l

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jul  6 19:32:09 2015 from 10.1.1.23
pi@raspberrypi ~ $
```

The Terminal

A terminal refers to a wrapper program which runs a shell. This used to mean a physical device consisting of little more than a monitor and keyboard. As Unix/Linux systems advanced the terminal concept was abstracted into software. Now we have programs such as LXTerminal (on the Raspberry Pi) which will launch a window in a Graphical User Interface (GUI) which will run a shell into which you can enter commands. Alternatively we can dispense with the GUI all together and simply start at the command line when we boot up.

The shell is a program which actually processes commands and returns output. Every Linux operating system has at least one shell, and most have several. The default shell on most Linux systems is bash.

The Commands

Commands on Linux operating systems are either built-in or external commands. Built-in commands are part of the shell. External commands are either executables (programs written in a programming language and then compiled into an executable binary) or shell scripts.

A command consists of a command name usually followed by one or more sequences of characters that include options and/or arguments. Each of these strings is separated by white space. The general syntax for commands is;

`commandname` [options] [arguments]

The square brackets indicate that the enclosed items are optional. Commands typically have a few options and utilise arguments. However, there are some commands that do not accept arguments, and a few with no options. As an example we can run the `ls` command with no options or arguments as follows;

```
ls
```

The `ls` command will list the contents of a directory and in this case the command and the output would be expected to look something like the following;

```
pi@raspberrypi ~ $ ls
Desktop                 python_games
```

Options

An option (also referred to as a switch or a flag) is a single-letter code, or sometimes a single word or set of words, that modifies the behaviour of a command. When multiple single-letter options are used, all the letters are placed adjacent to each other (not separated by spaces) and can be in any order. The set of options must usually be preceded by a single hyphen, again with no intervening space.

So again using `ls` if we introduce the option `-l` we can show the total files in the directory and subdirectories, the names of the files in the current directory, their permissions, the number of subdirectories in directories listed, the size of the file, and the date of last modification.

The command we execute therefore looks like this;

```
ls -l
```

And so the command (with the `-l` option) and the output would look like the following;

```
pi@raspberrypi ~ $ ls -l
total 26
drwxr-xr-x 2 pi pi 4096 Feb 20 08:07 Desktop
drwxrwxr-x 2 pi pi 4096 Jan 27 08:34 python_games
```

Here we can see quite a radical change in the formatting and content of the returned information.

Arguments

An argument (also called a command line argument) is a file name or other data that is provided to a command in order for the command to use it as an input.

Using `ls` again we can specify that we wish to list the contents of the `python_games` directory (which we could see when we ran `ls`) by using the name of the directory as the argument as follows;

```
ls python_games
```

The command (with the `python_games` argument) and the output would look like the following (actually I removed quite a few files to make it a bit more readable);

```
pi@raspberrypi ~ $ ls  python_games
4row_arrow.png          gem4.png                    pentomino.py
4row_black.png          gem5.png                    pinkgirl.png
4row_board.png          gem6.png                    Plain_Block.png
4row_computerwinner.png gem7.png                    princess.png
4row_humanwinner.png    gemgem.py                   RedSelector.png
gem1.png                match5.wav                  Wall_Block_Tall.png
gem2.png                memorypuzzle_obfuscated.py  Wood_Block_Tall.png
gem3.png                memorypuzzle.py             wormy.py
```

Putting it all together

And as our final example we can combine our command (`ls`) with both an option (`-l`) and an argument (`python_games`) as follows;

```
ls -l python_games
```

Hopefully by this stage, the output shouldn't come as too much surprise, although again I have pruned some of the files for readabilities sake;

```
pi@raspberrypi ~ $ ls -l  python_games
total 1800
-rw-rw-r-- 1 pi pi   9731 Jan 27 08:34 4row_arrow.png
-rw-rw-r-- 1 pi pi   7463 Jan 27 08:34 4row_black.png
-rw-rw-r-- 1 pi pi   8666 Jan 27 08:34 4row_board.png
-rw-rw-r-- 1 pi pi  18933 Jan 27 08:34 4row_computerwinner.png
-rw-rw-r-- 1 pi pi  25412 Jan 27 08:34 4row_humanwinner.png
-rw-rw-r-- 1 pi pi   8562 Jan 27 08:34 4row_red.png
-rw-rw-r-- 1 pi pi  14661 Jan 27 08:34 tetrisc.mid
-rw-rw-r-- 1 pi pi  15759 Jan 27 08:34 tetrominoforidiots.py
-rw-rw-r-- 1 pi pi  18679 Jan 27 08:34 tetromino.py
-rw-rw-r-- 1 pi pi   9771 Jan 27 08:34 Tree_Short.png
-rw-rw-r-- 1 pi pi  11546 Jan 27 08:34 Tree_Tall.png
-rw-rw-r-- 1 pi pi  10378 Jan 27 08:34 Tree_Ugly.png
-rw-rw-r-- 1 pi pi   8443 Jan 27 08:34 Wall_Block_Tall.png
-rw-rw-r-- 1 pi pi   6011 Jan 27 08:34 Wood_Block_Tall.png
-rw-rw-r-- 1 pi pi   8118 Jan 27 08:34 wormy.py
```

`apt-get`

The `apt-get` command is a program, that is used with Debian based Linux distributions to install, remove or upgrade software packages. It's a vital tool for installing and managing software and should be used on a regular basis to ensure that software is up to date and security patching requirements are met.

There are a plethora of uses for `apt-get`, but we will consider the basics that will allow us to get by. These will include;

- Updating the database of available applications (`apt-get update`)
- Upgrading the applications on the system (`apt-get upgrade`)
- Installing an application (`apt-get install *package-name*`)
- Un-installing an application (`apt-get remove *package-name*`)

The `apt-get` command

The `apt` part of `apt-get` stands for '**a**dvanced **p**ackaging **t**ool'. The program is a process for managing software packages installed on Linux machines, or more specifically Debian[57] based Linux machines (Since those based on 'redhat[58]' typically use their `rpm` (**r**ed hat **p**ackage **m**anagement (or more lately the recursively named '**r**pm **p**ackage **m**anagement') system). As Raspbian is based on Debian, so the examples we will be using are based on `apt-get`.

APT simplifies the process of managing software on Unix-like computer systems by automating the retrieval, configuration and installation of software packages. This was historically a process best described as 'dependency hell' where the requirements for different packages could mean a manual installation of a simple software application could lead a user into a sink-hole of despair.

In common `apt-get` usage we will be prefixing the command with `sudo` to give ourselves the appropriate permissions;

`apt-get update`

```
sudo apt-get update
```

This will resynchronize our local list of packages files, updating information about new and recently changed packages. If an `apt-get upgrade` (see below) is planned, an `apt-get update` should always be performed first.

Once the command is executed, the computer will delve into the internet to source the lists of current packages and download them so that we will see a list of software sources similar to the following appear;

[57] https://www.debian.org/
[58] http://www.redhat.com/

```
pi@raspberrypi ~ $ sudo apt-get update
Hit http://raspberrypi.collabora.com wheezy Release.gpg
Get:1 http://mirrordirector.raspbian.org wheezy Release.gpg [490 B]
Get:2 http://archive.raspberrypi.org wheezy Release.gpg [473 B]
Hit http://raspberrypi.collabora.com wheezy Release
Get:3 http://mirrordirector.raspbian.org wheezy Release [14.4 kB]
Get:4 http://archive.raspberrypi.org wheezy Release [17.6 kB]
Hit http://raspberrypi.collabora.com wheezy/rpi armhf Packages
Get:5 http://mirrordirector.raspbian.org wheezy/main armhf Packages [6,904 kB]
Get:6 http://archive.raspberrypi.org wheezy/main armhf Packages [130 kB]
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en
Fetched 7,140 kB in 35s (200 kB/s)
Reading package lists... Done
```

`apt-get upgrade`

```
sudo apt-get upgrade
```

The `apt-get upgrade` command will install the newest versions of all packages currently installed on the system. If a package is currently installed and a new version is available, it will be retrieved and upgraded. Any new versions of current packages that cannot be upgraded without changing the install status of another package will be left as they are.

As mentioned above, an `apt-get update` should always be performed first so that `apt-get upgrade` knows which new versions of packages are available.

Once the command is executed, the computer will consider its installed applications against the databases list of the most up to date packages and it will prompt us with a message that will let us know how many packages are available for upgrade, how much data will need to be downloaded and what impact this will have on our local storage. At this point we get to decide whether or not we want to continue;

```
pi@raspberrypi ~ $ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  bind9-host cups-bsd cups-client cups-common libapache2-mod-php5 libbind9-80
  libisccc80 libisccfg82 liblwres80 libsdl1.2debian libsqlite3-0 libssl1.0.0
  php5-mcrypt php5-mysql raspi-config
6 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 10.7 MB of archives.
After this operation, 556 kB disk space will be freed.
Do you want to continue [Y/n]?
```

Once we say yes ('Y') the upgrade kicks off and we will see a list of the packages as they are downloaded unpacked and installed (what follows is an edited example);

```
Do you want to continue [Y/n]? y
Get:1 http://archive.raspberrypi.org/debian/wheezy/main libsdl1.2debian
armhf 1.2.15-5+rpi1 [205 kB]
Get:2 http://archive.raspberrypi.org/debian/wheezy/main raspi-config all
20150131-5 [13.3 kB]
Get:3 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libsqlite3-0
armhf 3.7.13-1+deb7u2 [414 kB]
Fetched 10.7 MB in 31s (343 kB/s)
Preconfiguring packages ...
(Reading database ... 80703 files and directories currently installed.)
Preparing to replace cups-common 1.5.3-5+deb7u5
(using .../cups-common_1.5.3-5+deb7u6_all.deb) ...
Unpacking replacement cups-common ...
Preparing to replace cups-bsd 1.5.3-5+deb7u5
(using .../cups-bsd_1.5.3-5+deb7u6_armhf.deb) ...
Unpacking replacement cups-bsd ...
Preparing to replace php5-gd 5.4.39-0+deb7u2
(using .../php5-gd_5.4.41-0+deb7u1_armhf.deb) ...
Unpacking replacement php5-gd ...
Processing triggers for man-db ...
Setting up libssl1.0.0:armhf (1.0.1e-2+rvt+deb7u17) ...
Setting up libsqlite3-0:armhf (3.7.13-1+deb7u2) ...
Setting up cups-common (1.5.3-5+deb7u6) ...
Setting up cups-client (1.5.3-5+deb7u6) ...
```

There can often be alerts as the process identifies different issues that it thinks the system might strike (different aliases, runtime levels or missing fully qualified domain names). This is not necessarily a sign of problems so much as an indication that the process had to take certain configurations into account when upgrading and these are worth noting. Whenever there is any doubt about what has occurred, Google will be your friend :-).

`apt-get install`

The `apt-get install` command installs or upgrades one (or more) packages. All additional (dependency) packages required will also be retrieved and installed.

```
sudo apt-get install *package-name*
```

If we want to install multiple packages we can simply list each package separated by a space after the command as follows;

```
sudo apt-get install package1 package2 package3
```

`apt-get remove`

```
sudo apt-get remove *package-name*
```

The `apt-get remove` command removes one (or more) packages.

`cat`

The `cat` command is a really versatile command that is typically used to carry out three different functions. It can display a file on the screen, combine different files together (con**cat**enate them) or create new files. This is another core command that is immensely useful to learn in when working with Linux from the command line. It's simple, flexible and versatile.

- `cat` [options] *filename filename* : Display, combine or create new files.

For example: To display the file `foo.txt` on the screen we would use;

```
cat foo.txt
```

To display the files `foo.txt` and `bar.txt` on the screen one after the other we would use;

```
cat foo.txt bar.txt
```

Or to combine the files `foo.txt` and `bar.txt` into a new file called `foobar.txt` using the redirection symbol `>`;

```
cat foo.txt bar.txt > foobar.txt
```

The `cat` command

The `cat` command is a vital tool to use on the Linux command line because ultimately Linux is an operating system that is file driven. Without a graphical user interface there needs to be a mechanism whereby creating and manipulating text files can be accomplished easily. The `cat` command is one of the commands that makes this possible. The name 'cat' is short for '**cat**enate' or 'con**cat**enate' (either appears to be acceptable, but 'con**cat**enate' appears to be more widely used), which is to say to connect things in a series. This is certainly one of it's common uses, but a better overview would be to say that the `cat` command is used to;

- Display text files at the command line
- Join one text file to the end of another text file, combining them
- Copy text files into a new document

Options

The only option that gets any degree of use with cat is the `-n` option that numbers the output lines.

Arguments and Examples

To display text

For example, to display a text file (foo.txt) on the screen we can use the following command;

```
cat foo.txt
```

The output would be;

```
pi@raspberrypi ~ $ cat foo.txt
This line is the contents of foo.txt
```

As we can see, the contents of the file 'foo.txt' is sent to the screen (be aware, if the file is sufficiently large, it will simple dump the contents in a long scrolling waterfall of text).

To join more than one file together

We could just as easily display two files one after the other (concatenated) as follows;

```
cat foo.txt bar.txt
```

The output would be;

```
pi@raspberrypi ~ $ cat foo.txt bar.txt
This line is the contents of foo.txt
This line is the contents of bar.txt
```

To create a new file

Instead of having the file sent to the screen, we can specify that `cat` send our file to a new (renamed) file as follows;

```
cat foo.txt > newfoo.txt
```

This could be thought of an a equivalent to a file copy action and uses the redirection symbol >.

>> and > are called append symbols. They are used to append the output to a file and not to the screen. > will delete a file if it already exists and create a new file hence for safety it is advisable to use >> wherever possible to write the output to a new file without overwriting or deleting an existing file.

Taking the process one step further we can take our original two files and combine them into a single file with;

```
cat foo.txt bar.txt > foobar.txt
```

We can then check the results of our combination using `cat` on the new file as follows;

```
cat foobar.txt
```

And the output would be;

```
pi@raspberrypi ~ $ cat foobar.txt
This line is the contents of foo.txt
This line is the contents of bar.txt
```

Then we could use `cat` to append a file to an already existing file by using the redirection operator `>>`;

```
cat newfoo.txt >> foobar.txt
```

Here we use the redirection operator `>>` to add the contents of the file `newfoo.txt` to the already existing file `foobar.txt`.

The resulting file content would be;

```
pi@raspberrypi ~ $ cat foobar.txt
This line is the contents of foo.txt
This line is the contents of bar.txt
And this is newfoo.txt
```

Finally, we can use `cat` to create a file from scratch. In this scenario if we use `cat` without a source file and redirect to a new file (here called `newfile.txt`. It will take the input from the command line to add to the file until CONTROL-d is pressed.

```
cat >> newfile.txt
Just keep typing
and entering information until...
we press ctrl-d to finish and the file gets written!
```

The resulting file content would be;

```
pi@raspberrypi ~ $ cat newfile.txt
Just keep typing
and entering information until...
we press ctrl-d to finish and the file gets written!
```

Test yourself

1. Which is the safest redirector to use?
2. Create a new file using `cat` and enter a few lines of text to give it some content
3. Copy that file using `cat` to a new file.
4. Combine the original file and the copy into a new file.
5. Display that new file on the screen.

cd

The `cd` command is used to move around in the directory structure of the file system (**c**hange **d**irectory). It is one of the fundamental commands for navigating the Linux directory structure.

`cd` [options] *directory* : Used to change the current directory.

For example, when we first log into the Raspberry Pi as the 'pi' user we will find ourselves in the `/home/pi` directory. If we wanted to change into the `/home` directory (go up a level) we could use the command;

```
cd /home
```

Take some time to get familiar with the concept of moving around the directory structure from the command line as it is an important skill to establish early in Linux.

The `cd` command

The `cd` command will be one of the first commands that someone starting with Linux will use. It is used to move around in the directory structure of the file system (hence `cd` = **c**hange **d**irectory). It only has two options and these are seldom used. The arguments consist of pointing to the directory that we want to go to and these can be absolute or relative paths.

The `cd` command can be used without options or arguments. In this case it returns us to our home directory as specified in the `/etc/passwd` file.

If we cd into any random directory (try `cd /var`) we can then run cd by itself;

```
cd
```

… and in the case of a vanilla installation of Raspbian, we will change to the `/home/pi` directory;

```
pi@raspberrypi ~ $ cd /var
pi@raspberrypi /var $ cd
pi@raspberrypi ~ $ pwd
/home/pi
```

In the example above, we changed to `/var` and then ran the `cd` command by itself and then we ran the `pwd` command which showed us that the **p**resent **w**orking **d**irectory is `/home/pi`. This is the Raspbian default home directory for the pi user.

Options

As mentioned, there are only two options available to use with the `cd` command. This is `-P` which instructs `cd` to use the physical directory structure instead of following symbolic links and the `-L` option which forces symbolic links to be followed.

For those beginning Linux, there is little likelihood of using either of these two options in the immediate future and I suggest that you use your valuable memory to remember other Linux stuff.

Arguments

As mentioned earlier, the default argument (if none is included) is to return to the users home directory as specified in the `/etc/passwd` file.

When specifying a directory we can do this by absolute or relative addressing. So if we started in the `/home/pi` directory, we could go the `/home` directory by executing;

```
cd /home
```

... or using relative addressing and we can use the `..` symbols to designate the parent directory;

```
cd ..
```

Once in the `/home` directory, we can change into the `/home/pi/Desktop` directory using relative addressing as follows;

```
cd pi/Desktop
```

We can also use the `-` argument to navigate to the previous directory we were in.

Examples

Change into the root (`/`) directory;

```
cd /
```

Test yourself

1. Having just changed from the `/home/pi` directory to the `/home` directory, what are the five variations of using the `cd` command that will take the pi user to the `/home/pi` directory
2. Starting in the `/home/pi` directory and using only relative addressing, use `cd` to change into the `/var` directory.

`chmod`

The `chmod` command allows us to set or modify a file's permissions. Because Linux is built as a multi-user system there are typically multiple different users with differing permissions for which files they can read / write or execute. `chmod` allows us to limit access to authorised users to do things like editing web files while general users can only read the files.

- `chmod` [options] *mode files* : Change access permissions of one or more files & directories

For example, the following command (which would most likely be prefixed with `sudo`) sets the permissions for the `/var/www` directory so that the user can read from, write to and change into the directory. Group owners can also read from, write to and change into the directory. All others can read from and change into the directory, but they cannot create or delete a file within it;

```
chmod 775 /var/www
```

This might allow normal users to browse web pages on a server, but prevent them from editing those pages (which is probably a good thing).

The `chmod` command

The `chmod` command allows us to change the permissions for which user is allowed to do what (read, write or execute) to files and directories. It does this by changing the 'mode' (hence `chmod` = **ch**ange file **mod**e) of the file where we can make the assumption that 'mode' = permissions.

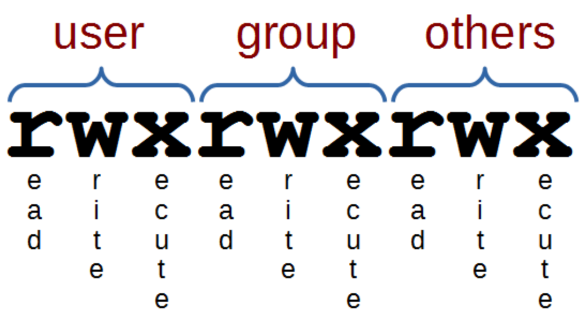
Every file on the computer has an associated set of permissions. Permissions tell the operating system what can be done with that file and by whom. There are three things you can (or can't) do with a given file:

- read it,
- write (modify) it and
- execute it.

Linux permissions specify what the owning user can do, what the members of the owning group can do and what other users can do with the file. For any given user, we need three bits to specify access permissions: the first to denote read (r) access, the second to denote (w) access and the third to denote execute (x) access.

We also have three levels of ownership: 'user', 'group' and 'others' so we need a triplet (three sets of three) for each, resulting in nine bits.

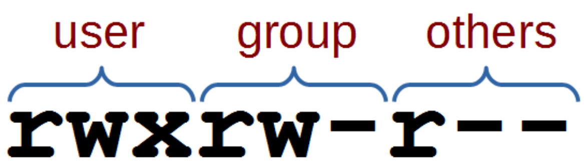
The following diagram shows how this grouping of permissions can be represented on a Linux system where the user, group and others had full read, write and execute permissions;



Linux permissions as rwx

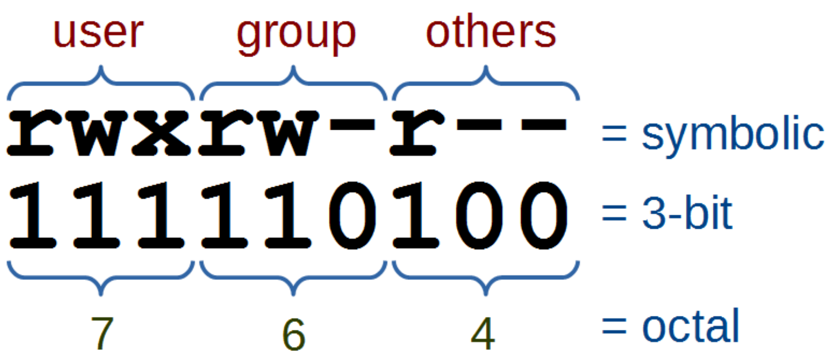
Usually in Linux (when you execute the `ls -l` command) there is also another bit that leads this 9-bit pattern, but we will ignore this in the mean time.

If we had a file with more complex permissions where the user could read, write and execute, the group could read and write, but all other users could only read it would look as follows;



Slightly more complex Linux permissions

This description of permissions is workable, but we will need to be aware that the permissions are also represented as 3 bit values (where each bit is a ‘1’ or a ‘0’ (where a ‘1’ is *yes you can*, or ‘0’ is *no you can’t*)) or as the equivalent octal value.



Linux permissions as symbolic, 3 bit and octal

The full range of possible values for these permission combinations is as follows;

Permission	Symbolic	3-bit	Octal
read, write and execute	rwx	111	7
read and write	rw-	110	6
read and execute	r-w	101	5
read only	r--	100	4
write and execute	-wx	011	3
write only	-w-	010	2
execute only	--x	001	1
none	---	000	0

Another interesting thing to note is that permissions take a different slant for directories.

- read determines if a user can view the directory's contents, i.e. execute `ls` in it.
- write determines if a user can create new files or delete file in the directory. (Note here that this essentially means that a user with write access to a directory can delete files in the directory even if he/she doesn't have write permissions for the file! So be careful.)
- execute determines if the user can cd into the directory.

It's also worth noting at this point that only the owner (or root via `sudo`) of a file may use `chmod` to alter a file's permissions.

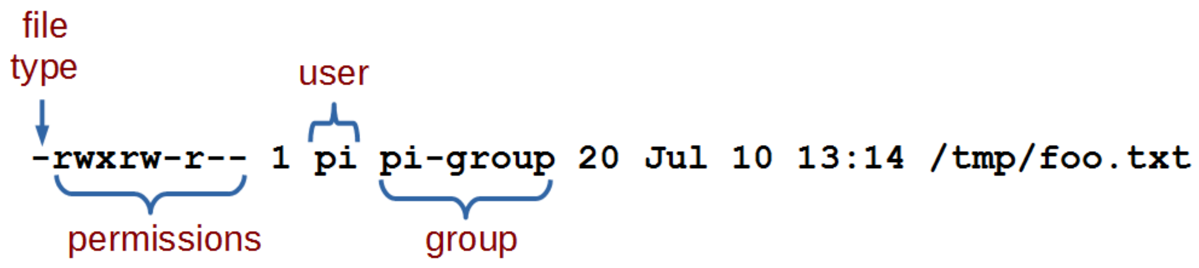
We can check the check the permissions of files using the `ls -l` command which will list files in a long format as follows;

```
ls -l /tmp/foo.txt
```

This command will list the details of the file `foo.txt` that is in the `/tmp` directory as follows

```
pi@raspberrypi ~ $ ls -l /tmp
-rwxrw-r-- 1 pi pi-group 20 Jul 10 13:14 foo.txt
```

The permissions on the file, the user and the group owner can be found as follows;



```
-rwxrw-r-- 1 pi pi-group 20 Jul 10 13:14 /tmp/foo.txt
```

File details

From this information we can see that the file's user ('pi') has permissions to read, write and execute the file. The group owner ('pi-group') can read and write to the file and all other users can read the file.

Options

The main option that is worth remembering is the `-R` option that will **R**ecursively apply permissions on the files in the specified directory and its sub-directories.

The following command will change the permissions for all the files in the `/srv/foo` directory and in all the directories that are under it;

```
chmod -R 764 /srv/foo
```

Arguments

Simplistically (in other words it can be more complicated, but we're simplifying it) there are two main ways that `chmod` is used. In either symbolic mode where the permissions are changed using symbols associated with read, write and execute as well as symbols for the user (`u`), the group owner (`g`), others (`o`) and all users (`a`). Or in numeric mode where we use the octal values for permission combinations.

Symbolic Mode

In symbolic mode we can change the permissions of a file with the following syntax:

- `chmod` [who][op][permissions] *filename*

Where `who` can be the user (`u`), the group owner (`g`) and / or others (`o`). The operator (`op`) is either `+` to add a permission, `-` to remove a permission or `=` to explicitly set permissions. The `permissions` themselves are either readable (`r`), writeable (`w`), or executable (`x`).

For example the following command adds executable permissions (`x`) to the user (`u`) for the file `/tmp/foo.txt`;

```
chmod u+x /tmp/foo.txt
```

This command removes writing (`w`) and executing (`x`) permissions from the group owner (`g`) and all others (`o`) for the same file;

```
chmod go-wx /tmp/foo.txt
```

Hopefully you will note that you can combine the 'who' and 'permissions' fields to allow multiple values.

Note that removing the execute permission from a directory will prevent you from being able to list its contents (although root will override this). If you accidentally remove the execute permission from a directory, you can use the `+X` argument to instruct `chmod` to only apply the execute permission to directories.

```
chmod -R u+X /home/pi/*
```

Numeric Mode

In numeric mode we can explicitly state the permissions using the octal values, so this form of the command is fairly common.

For example, the following command will change the permissions on the file `foo.txt` so that the user can read, write and execute it, the group owner can read and write it and all others can read it;

```
chmod 764 /tmp/foo.txt
```

Examples

To change the permissions in your home directory to remove reading and executing permissions from the group owner and all other users;

```
chmod go-rx ~
```

To make a script executable by the user;

```
chmod u+x foo.sh
```

Windows marks all files as executable by default. If you copy a file or directory from a Windows system (or even a Windows-formatted disk) to your Linux system, you should ideally strip the unnecessary execute permissions from all copied files unless you specifically need to retain it. Note of course we still need it on all //directories// so that we can access their contents! Here's how we can achieve this in one command:

```
chmod -R a-x+X ~/copied_from_windows
```

This instructs chmod to remove the execute permission for each file and directory, and then immediately set execute again if working on a directory.

crontab

The `crontab` command give the user the ability to schedule tasks to be run at a specific time or with a specific interval. If you want to move beyond using Linux from a graphical user interface, you will most likely want to schedule a task to run at a particular time or interval. Even just learning about it might give you ideas of what you might do.

- `crontab` [-u user] [-l | -r | -e] : Schedule a task to run at a particular time or interval

For example, you could schedule a script to run every day to carry our a backup process in the middle of the night. or capture some data every hour to store in a database.

The `crontab` command

The command `crontab` is a concatenation of 'cron table' because it uses the job scheduler `cron` to execute tasks which are stored in a 'table' of sorts in the users crontab file. `cron` is named after 'Khronos', the Greek personification of time.

While each user who sets up a job to run using the crontab creates a crontab file, the file is not intended to be edited by hand. It is in different locations in different flavour of Linux distributions and the most reliable mechanism for editing it is by running the `crontab -e` command. Each user has their own crontab file and the root user can edit another users crontab file. This would be the situation where we would use the `-u` option, but honestly once we get to that stage it can probably be assumed that we know a fair bit about Linux.

There are only three main options that are used with `crontab`.

Options

The first option that we should examine is the `-l` option which allows us to list the crontab file;

```
crontab -l
```

Once run it will list the contents of the crontab file directly to the screen. The output will look something like;

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

*/10 * * * * /usr/bin/php /home/pi/scrape-books.php
```

Here we can see that the main part of the file (in fact everything except the final line) is comments that explain how to include an entry into the crontab file.

The entry in this case is specified to run every 10 minutes and when it does, it will run the PHP script `scrape-books.php` (we'll explain how this is encoded later in the examples section).

If we want to remove the current crontab we can use the `-r` option. Probably not something that we would do an a regular basis, as it would be more likely to be editing the content rather than just removing it wholesale.

Lastly there is the option to edit the crontab file which is initiated using `-e`. This is the main option that would be used and the one we will cover in detail in the examples below.

Examples

As an example, consider that we wish to run a Python script every day at 6am. The following command will let us edit the crontab;

```
crontab -e
```

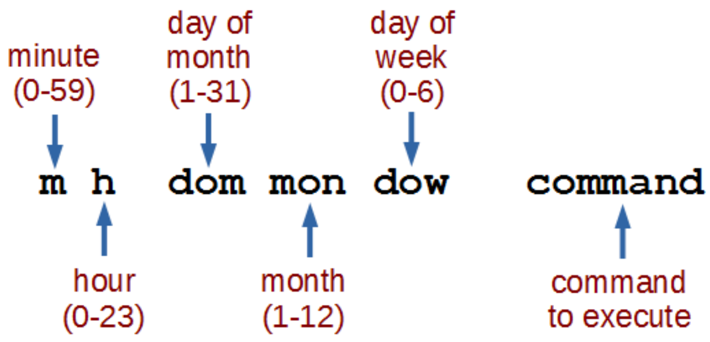
Once run it will open the crontab in the default editor on your system (most likely 'vi', 'vim' or 'nano'). The file will look as follows;

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
```

As stated earlier, the default file obviously includes some explanation of how to format an entry in the crontab. In our case we wish to add in an entry that tells the script to start at 6 hours and 0 minutes each day. The crontab accepts six pieces of information that will allow that action to be performed. each of those pieces is separated by a space.

1. A number (or range of numbers), m, that represents the minute of the hour (valid values 0-59);
2. A number (or range of numbers), h, that represents the hour of the day (valid values 0-23);
3. A number (or range of numbers), dom, that represents the day of the month (valid values 0-31);
4. A number (or list, or range), or name (or list of names), mon, that represents the month of the year (valid values 1-12 or Jan-Dec);
5. A number (or list, or range), or name (or list of names), dow, that represents the day of the week (valid values 0-6 or Sun-Sat); and
6. command, which is the command to be run, exactly as it would appear on the command line.

The layout is therefore as follows;



Linux permissions as rwx

Assuming that we want to run a Python script called 'm_temp.py' which was in the 'pi' home directory the line that we would want to add would be as follows;

```
0 6 * * * /usr/bin/python /home/pi/m_temp.py
```

So at minute 0, hour 6, every day of the month (where the asterisk denotes 'everything'), every month, every day of the week we run the command `/usr/bin/python /home/pi/m_temp.py` (which, if we were at the command line in the pi home directory we would run as `python m_temp.py`, but since we can't guarantee where we will be when running the script, we are supplying the full path to the `python` command and the `m_temp.py` script.

If we wanted to run the command twice a day (6am and 6pm (1800hrs)) we can supply a comma separated value in the hours (`h`) field as follows;

```
0 6,18 * * * /usr/bin/python /home/pi/m_temp.py
```

If we wanted to run the command at 6am but only on weekdays (Monday through Friday) we can supply a range in the `dow` field as follows (remembering that 0 = Sunday);

```
0 6 * * 1-5 /usr/bin/python /home/pi/m_temp.py
```

If we want to run the same command every 2 hours we can use the `*/2` notation, so that our line in the crontab would look like the following;

```
0 */2 * * * /usr/bin/python /home/pi/m_temp.py
```

It's important to note that we need to include the `0` at the start (instead of the `*`) so that it doesn't run every minute every 2 hours (every minute in other words)

Test yourself

1. How could you set up a schedule job in crontab that ran every second?
2. Create a crontab line to run a command on the 20th of July every year at 2 minutes past midnight.

ifconfig

The `ifconfig` command can be used to view the configuration of, or to configure a network interface. Networking is a fundamental function of modern computers. `ifconfig` allows us to configure the network interfaces to allow that connection.

- `ifconfig` [arguments] [interface]

or

- `ifconfig` [arguments] *interface* [options]

Used with no 'interface' declared `ifconfig` will display information about all the operational network interfaces. For example running;

```
ifconfig
```

… produces something similar to the following on a simple Raspberry Pi.

```
eth0      Link encap:Ethernet  HWaddr 76:12:45:56:47:53
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan0     Link encap:Ethernet  HWaddr 09:87:65:54:43:32
          inet addr:10.1.1.8  Bcast:10.1.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3978 errors:0 dropped:898 overruns:0 frame:0
          TX packets:347 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:859773 (839.6 KiB)  TX bytes:39625 (38.6 KiB)
```

The output above is broken into three sections; eth0, lo and wlan0.

- `eth0` is the first Ethernet interface and in our case represents the RJ45 network port on the Raspberry Pi (in this specific case on a B+ model). If we had more than one Ethernet interface, they would be named `eth1`, `eth2`, etc.
- `lo` is the loopback interface. This is a special network interface that the system uses to communicate with itself. You can notice that it has the IP address 127.0.0.1 assigned to it. This is described as designating the 'localhost'.
- `wlan0` is the name of the first wireless network interface on the computer. This reflects a wireless USB adapter (if installed). Any additional wireless interfaces would be named `wlan1`, `wlan2`, etc.

The `ifconfig` command

The `ifconfig` command is used to read and manage a servers network interface configuration (hence `ifconfig` = **i**nter**f**ace **config**uration).

We can use the `ifconfig` command to display the current network configuration information, set up an ip address, netmask or broadcast address on an network interface, create an alias for network interface, set up hardware addresses and enable or disable network interfaces.

`ifconfig` has been 'deprecated' in some Linux distributions, which means that the software has been superseded and where practical an alternative used. Although deprecated, the command is still available, although its use may raise warning messages recommending alternative practices, and deprecation may indicate that the feature will be removed in the future. Features are deprecated rather than immediately removed in order to provide backward compatibility. In the case of `ifconfig` the alternative is `ip`.

To view the details of a specific interface we can specify that interface as an argument;

```
ifconfig eth0
```

Which will produce something similar to the following;

```
eth0      Link encap:Ethernet  HWaddr b8:27:eb:2c:bc:62
          inet addr:10.1.1.8  Bcast:10.1.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:119833 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8279 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8895891 (8.4 MiB)  TX bytes:879127 (858.5 KiB)
```

The configuration details being displayed above can be interpreted as follows;

- `Link encap:Ethernet` - This tells us that the interface is an Ethernet related device.

- `HWaddr b8:27:eb:2c:bc:62` - This is the hardware address or Media Access Control (MAC) address which is unique to each Ethernet card. Kind of like a serial number.
- `inet addr:10.1.1.8` - indicates the interfaces IP address.
- `Bcast:10.1.1.255` - denotes the interfaces broadcast address
- `Mask:255.255.255.0` - is the network mask for that interface.
- `UP` - Indicates that the kernel modules for the Ethernet interface have been loaded.
- `BROADCAST` - Tells us that the Ethernet device supports broadcasting (used to obtain IP address via DHCP).
- `RUNNING` - Lets us know that the interface is ready to accept data.
- `MULTICAST` - Indicates that the Ethernet interface supports multicasting.
- `MTU:1500` - Short for for **M**aximum **T**ransmission **U**nit is the size of each packet received by the Ethernet card.
- `Metric:1` - The value for the Metric of an interface decides the priority of the device (to designate which of more than one devices should be used for routing packets).
- `RX packets:119833 errors:0 dropped:0 overruns:0 frame:0` and `TX packets:8279 errors:0 dropped:0 overruns:0 carrier:0` - Show the total number of packets received and transmitted with their respective errors, number of dropped packets and overruns respectively.
- `collisions:0` - Shows the number of packets which are colliding while traversing the network.
- `txqueuelen:1000` - Tells us the length of the transmit queue of the device.
- `RX bytes:8895891 (8.4 MiB)` and `TX bytes:879127 (858.5 KiB)` - Indicates the total amount of data that has passed through the Ethernet interface in transmit and receive.

Options

The main option that would be used with `ifconfig` is `-a` which will will display **a**ll of the interfaces on the interfaces available (ones that are 'up' (active) and 'down' (shut down). The default use of the `ifconfig` command without any arguments or options will display only the active interfaces.

```
ifconfig -a
```

Arguments

We can disable an interface (turn it down) by specifying the interface name and using the suffix 'down' as follows;

```
ifconfig eth0 down
```

Or we can make it active (bring it up) by specifying the interface name and using the suffix 'up' as follows;

```
ifconfig eth0 up
```

To assign a IP address to a specific interface we can specify the interface name and use the IP address as the suffix;

```
ifconfig eth0 10.1.1.8
```

To add a netmask to a a specific interface we can specify the interface name and use the `netmask` argument followed by the netmask value;

```
ifconfig eth0 netmask 255.255.255.0
```

To assign an IP address and a netmask at the same time we can combine the arguments into the same command;

```
ifconfig eth0 10.1.1.8 netmask 255.255.255.0
```

Test yourself

1. List all the network interfaces on your server.
2. Why might it be a bad idea to turn down a network interface while working on a server remotely?
3. Display the information about a specific interface, turn it down, display the information about it again then turn it up. What differences do you see?

`ls`

The `ls` command lists the contents of a directory and can show the properties of those objects it lists. It is one of the fundamental commands for knowing what files are where and the properties of those files.

- `ls` [options] *directory* : List the files in a particular directory

For example: If we execute the `ls` command with the `-l` option to show the properties of the listings in **l**ong format and with the argument `/var` so that it lists the content of the `/var` directory…

```
ls -l /var
```

… we should see the following;

```
pi@raspberrypi ~ $ ls -l /var
total 102440
drwxr-xr-x  2 root     root          4096 Mar  7 06:25 backups
drwxr-xr-x 12 root     root          4096 Feb 20 08:33 cache
drwxr-xr-x 43 root     root          4096 Feb 20 08:33 lib
drwxrwsr-x  2 root     uucp          4096 Jan 11 00:02 local
lrwxrwxrwx  1 root     root             9 Feb 15 11:23 lock -> /run/lock
drwxr-xr-x 11 root     root          4096 Jul  7 06:25 log
drwxrwsr-x  2 root     mail          4096 Feb 15 11:23 mail
drwxr-xr-x  2 root     root          4096 Feb 15 11:23 opt
lrwxrwxrwx  1 root     root             4 Feb 15 11:23 run -> /run
drwxr-xr-x  4 root     root          4096 Feb 15 11:26 spool
-rw-------  1 root     root     104857600 Feb 16 14:03 swap
drwxrwxrwt  2 root     root          4096 Jan 11 00:02 tmp
drwxrwxr-x  2 www-data www-data      4096 Feb 20 08:21 www
```

What’s the information in the long list format?

- 1st column will give detailed information regarding file permission,
- 2nd column will tell you about the number of links to the file,
- 3rd and 4th columns are associated with owner and group of the file,
- 5th column will be displaying the size of the file in bytes,
- 6th column will display the recent time and date at which the file was modified,
- and the last and 7th column is the actual file/directory/link name.

The `ls` command

The `ls` command will be one of the first commands that someone starting with Linux will use. It is used to list the contents of a directory (hence `ls` = **l**i**s**t). It has a large number of options for displaying listings and their properties in different ways. The arguments used are normally the name of the directory or file that we want to show the contents of.

By default the `ls` command will show the contents of the current directory that the user is in and just the names of the files that it sees in the directory. So if we execute the `ls` command on its own from the pi users home directory (where we would be after booting up the Raspberry Pi), this is the command we would use;

```
ls
```

… and we should see the following;

```
pi@raspberrypi ~ $ ls
Desktop  python_games
```

This shows two directories (`Desktop` and `python_games`) that are in pi’s home directory, but there are no details about the directories themselves. To get more information we need to include some options.

Options

There are a very large number of options available to use with the `ls` command. For a full listing type `man ls` on the command line. Some of the most commonly used are;

- `-l` gives us a **l**ong listing (as explained above)
- `-a` shows us **a**LL the files in the directory, including hidden files
- `-s` shows us the **s**ize of the files (in blocks, not bytes)
- `-h` shows the size in “**h**uman readable format” (ie: 4K, 16M, 1G etc). (must be used in conjunction with the -s option).
- `-S` sorts by file **S**ize
- `-t` sorts by modification **t**ime
- `-r` **r**everses order while sorting

A useful combination of options could be a long listing (`-l`) that shows all (`-a`) the files with the file size being reported in human readable (`-h`) block size (`-s`).

```
ls -lash
```

… will produce something like the following;

```
pi@raspberrypi ~ $ ls -lash
total 84K
4.0K drwxr-xr-x 13 pi   pi   4.0K May  7 11:46 .
4.0K drwxr-xr-x  3 root root 4.0K May  7 10:20 ..
4.0K -rw-r--r--  1 pi   pi     69 May  7 11:46 .asoundrc
4.0K -rw-------  1 pi   pi    854 Jul  8 12:55 .bash_history
4.0K -rw-r--r--  1 pi   pi   3.2K May  7 10:20 .bashrc
4.0K drwxr-xr-x  4 pi   pi   4.0K May  7 11:46 .cache
4.0K drwxr-xr-x  7 pi   pi   4.0K May  7 11:46 .config
4.0K drwxr-xr-x  2 pi   pi   4.0K May  7 11:46 Desktop
4.0K drwxr-xr-x  2 pi   pi   4.0K May  7 11:46 .fontconfig
4.0K drwxr-xr-x  2 pi   pi   4.0K May  7 11:46 .gstreamer-0.10
4.0K drwx------  3 pi   pi   4.0K May  7 11:46 .local
4.0K -rw-r--r--  1 pi   pi    675 May  7 10:20 .profile
4.0K drwxrwxr-x  2 pi   pi   4.0K Jan 27 21:34 python_games
4.0K drwxr-xr-x  3 pi   pi   4.0K May  7 11:46 .themes
```

The size of the reported files when using the human readable option are designated by the following respective letters;

- K = Kilobyte
- M = Megabyte
- G = Gigabyte
- T = Terabyte
- P = Petabyte
- E = Exabyte
- Z = Zettabyte
- Y = Yottabyte

Arguments

The default argument (if none is included) is to list the contents of the directory that the user is currently in. Otherwise we can specify the directory to list. This might seem like a simple task, but there are a few tricks that can make using ls *really* versatile.

The simplest example of using a specific directory for an argument is to specify the location with the full address. For example, if we wanted to list the contents of the `/var` directory (and it doesn't matter which directory we run this command from) we simply type;

```
ls /var
```

… will produce the following;

```
pi@raspberrypi ~ $ ls /var
backups  cache  lib  local  lock  log  mail  opt  run  spool  swap  tmp  www
```

We can also use some of the relative addressing characters to shortcut our listing. We can list the home directory by using the tilde (`ls ~`) and the parent directory by using two full stops (`ls ..`).

The asterisk (*) can be used as a wildcard to list files with similar names. E.g. to list all the `png` file in a directory we can use `ls *.png`.

If we just want to know the details of a specific file we can use its name explicitly. For example if we wanted to know the details of the `swap` file in `/var` we would use the following command;

```
ls -l /var/swap
```

... which will produce the following;

```
pi@raspberrypi ~ $ ls -l /var/swap
-rw------- 1 root root 104857600 May  7 11:29 /var/swap
```

Examples

List all the configuration (`.conf`) files in the `/etc` directory;

```
ls /etc/*.conf
```

... which will produce the following;

```
pi@raspberrypi ~ $ ls /etc/*.conf
/etc/adduser.conf          /etc/host.conf       /etc/ntp.conf
/etc/ca-certificates.conf  /etc/idmapd.conf     /etc/pam.conf
/etc/debconf.conf          /etc/insserv.conf    /etc/resolv.conf
/etc/deluser.conf          /etc/ld.so.conf      /etc/resolvconf.conf
/etc/dhcpcd.conf           /etc/libaudit.conf   /etc/rsyslog.conf
/etc/fuse.conf             /etc/logrotate.conf  /etc/sysctl.conf
/etc/gai.conf              /etc/mke2fs.conf     /etc/ts.conf
/etc/gssapi_mech.conf      /etc/nsswitch.conf   /etc/ucf.conf
```

`ping`

The `ping` command allows us to check the network connection between the local computer and a remote server. It does this by sending a request to the remote server to reply to a message (kind of like a read-request in email). This allows us to test network connectivity to the remote server and to see if the server is operating. The `ping` command is a simple and commonly used network troubleshooting tool.

- `ping` [options] *remote server* : checks the connection to a remote server.

To check the connection to the server at CNN for example we can simple execute the following command (assuming that we have a connection to the internet);

```
ping cnn.com
```

Which will return something like the following;

```
PING cnn.com (157.166.226.25) 56(84) bytes of data.
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=1 ttl=111 time=265 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=2 ttl=111 time=257 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=3 ttl=111 time=257 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=4 ttl=111 time=258 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=5 ttl=111 time=257 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=6 ttl=111 time=257 ms
^C
--- cnn.com ping statistics ---
14 packets transmitted, 13 received, 7% packet loss, time 13016ms
rtt min/avg/max/mdev = 257.199/267.169/351.320/24.502 ms
```

The first thing to note is that by default the `ping` command will just keep running. When we want to stop it we need to press CTRL-c to get it to stop.

The information presented is extremely useful and tells us that www.cnn.com’s IP address is 157.166.226.25 and that the time taken for a ping send and return message took about 250 milliseconds.

As an aside, the ping command is one of those commands that has illustrated nicely for me the need to write a book that has some simpler language in it to explain things. For example, the following is the description of the `ping` command from the `man` pages;

> "ping uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams ('pings') have an IP and ICMP header, followed by a struct timeval and then an arbitrary number of 'pad' bytes used to fill out the packet."

Don't get me wrong. It's a great description and I'm not suggesting that it be changed, but it's not entirely friendly to a new user.

The `ping` command

The `ping` command is a very simple network / connectivity checking tool that is one of the default 'go-to' commands for system administrators. You might be wondering about how the name has come about. It is reminiscent of the echo-location technique used by dolphins, whales and bats to send out a sound and to judge their surroundings by the returned echo. In the dramatised world of the submariner, a ping is the sound emitted by a submarine in the same way to judge the distance and direction to an object. It was illustrated to best effect in the book by Tom Clancy and the subsequent movie "The Hunt for Red October[59]" where the submarine commander makes the request for "One Ping Only".

One Ping Only

It works by sending message called an 'Echo Request' to a specific network location (which we specify as part of the command). When (or if) the server receives the request it sends an 'Echo Reply' to the originator that includes the exact payload received in the request. The command will continue to send and (hopefully) receive these echoes until the command completes its requisit number of attempts or the command is stopped by the user (with a CTRL-c). Once complete, the command summarises the effort.

From the example used above we can see the output as follows;

[59]https://en.wikipedia.org/wiki/The_Hunt_for_Red_October

```
PING cnn.com (157.166.226.25) 56(84) bytes of data.
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=1 ttl=111 time=265 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=2 ttl=111 time=257 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=3 ttl=111 time=257 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=4 ttl=111 time=258 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=5 ttl=111 time=257 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=6 ttl=111 time=257 ms
^C
--- cnn.com ping statistics ---
14 packets transmitted, 13 received, 7% packet loss, time 13016ms
rtt min/avg/max/mdev = 257.199/267.169/351.320/24.502 ms
```

We can see from the returned pings that the IP address of the server that is designated as ‘www.cnn.com’ is ‘157.166.226.25’ The resolution of the IP address would be made possible by DNS, but using a straight IP address is perfectly fine). The `icmp_seq=` column tells us the sequence of the returned replies and ttl indicates how many IP routers the packet can go through before being thrown away. The time provides the measured return trip of the request and reply.

The summary at completion tells us how many packets were sent and how many received back. This forms a percentage of lost packets which is established over the specified time. The final line provides a minimum, average maximum and standard deviation from the mean.

Options

There are a few different options for use, but the more useful are as follows;

- `-c` only ping the connection a certain number (**c**ount) of times
- `-i` change the time **i**nterval between pings

It’s really useful to have ping running continuously so that we can make changes to networking while watching the results, but it’s also useful to run the command for a limited amount of time. This is where the `-c` option comes in. This will simply restrict the number of pings that are sent out and will then cease and summarise the effort. This can be used as follows;

```
ping -c 4 cnn.com
```

Which will return something like the following;

```
PING cnn.com (157.166.226.25) 56(84) bytes of data.
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=1 ttl=111 time=266 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=2 ttl=111 time=263 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=3 ttl=111 time=288 ms
64 bytes from www.cnn.com (157.166.226.25): icmp_seq=4 ttl=111 time=280 ms

--- cnn.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 263.283/274.586/288.637/10.386 ms
```

Sometimes it can be convenient to set our own time interval between pings. This can be accomplished with the `-i` option which will let us vary the repeat time. The default is 1 second, however the value cannot be set below 0.2 seconds without doing so as the superuser. Interestingly there is an option to flood the network with pings (flood mode) to test the network infrastructure. However, this would be something typically left to research *carefully* when you really need it.

Test yourself

1. How does the ping command to a server name know how to return an IP address?
2. What does 'ttl' stand for?

sudo

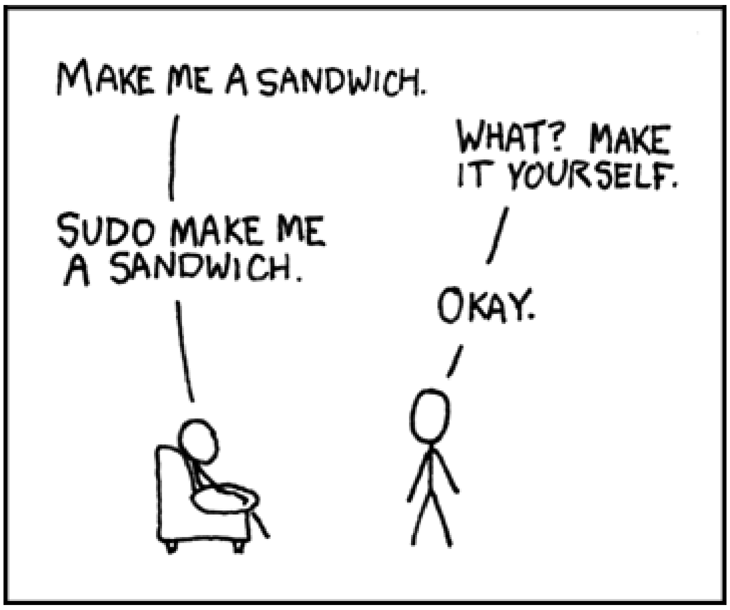
The sudo command allows a user to execute a command as the 'superuser' (or as another user). It is a vital tool for system administration and management.

- sudo [options] [command] : Execute a command as the superuser

For example, if we want to update and upgrade our software packages, we will need to do so as the super user. All we need to do is prefix the command apt-get with sudo as follows;

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

One of the best illustrations of this is via the excellent cartoon work of the xkcd comic strip[60] (Buy his stuff[61], it's awesome!).



Sudo courtesy xkcd

The sudo command

The sudo command is shorthand for '**s**uper**u**ser **do**'.

The sudo command allows the user to run programs or give commands that should only be executed with a degree of caution as they could potentially affect the normal operation of the computer. However, a user can only use this command if they have the correct permissions to do so.

[60] https://xkcd.com/149/
[61] http://store.xkcd.com/

When we use `sudo` an authorised user is determined by the contents of the file `/etc/sudoers`.

As an example of usage we should check out the file `/etc/sudoers`. If we use the `cat` command to list the file like so;

```
cat /etc/sudoers
```

We get the following response;

```
pi@raspberrypi ~ $ cat /etc/sudoers
cat: /etc/sudoers: Permission denied
```

That's correct, the 'pi' user does not have permissions to view the file

Let's confirm that with `ls`;

```
ls /etc/sudoers
```

Which will result in the following;

```
pi@raspberrypi ~ $ ls -l /etc/sudoers
-r--r----- 1 root root 696 May  7 10:39 /etc/sudoers
```

It would appear that only the root user can read the file!

So let's use `sudo` to `cat`](#cat) the file as follows;

```
sudo cat /etc/sudoers
```

That will result in the following output;

```
pi@raspberrypi ~ $ sudo cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
```

```
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/s\
bin:/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
pi ALL=(ALL) NOPASSWD: ALL
```

DO NOT edit this file with a text editor! Always use the `visudo` command like the instructions say! (Interestingly, Raspbian has the nano editor (not vi) configured as the default editor.)

There's a lot of information in the file, but there, right at the bottom is the line that determines the privileges for the 'pi' user;

```
pi ALL=(ALL) NOPASSWD: ALL
```

We can break down what each section means;

`pi`

pi `ALL=(ALL) NOPASSWD: ALL`

The `pi` portion is the user that this particular rule will apply to.

`ALL`

`pi` **ALL**`=(ALL) NOPASSWD: ALL`

The first `ALL` portion tells us that the rule applies to all hosts.

`ALL`

`pi ALL=(`**ALL**`) NOPASSWD: ALL`

The second `ALL` tells us that the user 'pi' can run commands as all users and all groups.

`NOPASSWD`

`pi ALL=(ALL)` **NOPASSWD**`: ALL`

The `NOPASSWD` tells us that the user ‘pi’ won’t be asked for their password when executing a command with `sudo`.

`All`

`pi ALL=(ALL) NOPASSWD:` **ALL**‘

The last `ALL` tells us that the rules on the line apply to all commands.

Under normal situations the use of `sudo` would require a user to be authorised and then enter their password. By default the Raspbian operating system has the ‘pi’ user configured in the `/etc/sudoers` file to avoid entering the password every time.

If your curious about what privileges (if any) a user has, we can execute `sudo` with the `-l` option to list them;

```
sudo -l
```

This will result in output that looks similar to the following;

```
pi@raspberrypi ~ $ sudo -l
Matching Defaults entries for pi on this host:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User pi may run the following commands on this host:
    (ALL : ALL) ALL
    (ALL) NOPASSWD: ALL
```

The ‘sudoers’ file

As mentioned above, the file that determines permissions for users is `/etc/sudoers`. **DO NOT EDIT THIS BY HAND**. Use the `visudo` command to edit. Of course you will be required to run the command using `sudo`;

```
sudo visudo
```

I’m going to reinforce the point a bit more that starting to configure the `sudoers` file is a task that should be taken very seriously and with full knowledge of the security implications.

If you open up a `sudoers` file to edit it and you see something like the following;

```
bob ALL=(ALL) ALL
john ALL=(ALL) ALL
eve ALL=(ALL) ALL
someguy ALL=(ALL) ALL
```

It would indicate that there are a fair number of people with full administration rights to this server and perhaps this should be reviewed?

`sudo` vs `su`

There is a degree of confusion about the roles of the `sudo` command vs the `su` command. While both can be used to gain root privileges, the `su` command actually **s**witches the user to another user, while sudo only runs the specified command with different privileges. While there will be a degree of debate about their use, it is widely agreed that for simple on-off elevation, `sudo` is ideal.

Test yourself

1. Write an entry for the `sudoers` file that provides sudo priviledges to a user for only the `cat` command.
2. Under what circumstances can you edit the `sudoers` file with a standard text editor.

Directory Structure Cheat Sheet

- `/` : The ‘root’ directory which contains all other files and directories
- `/bin` : Common commands / programs, shared by all users
- `/boot` : Contains the files needed to successfully start the computer during the boot process
- `/dev` : Holds device files that represent physical and ‘logical’ devices
- `/etc` : Contains configuration files that control the operation of programs
- `/etc/cron.d`: One of the directories that allow programs to be run on a regular schedule
- `/etc/rc?.d` : Directories containing files that control the mode of operation of a computer
- `/home` : A directory that holds subdirectories for each user to store user specific files
- `/lib` : Contains shared library files and kernel modules
- `/lost+found` : Will hold recoverable data in the event of an an improper shut-down
- `/media` : Used to temporarily mount removable devices
- `/mnt` : A mount point for filesystems or temporary mount point for system administrators
- `/opt` : Contains third party or additional software that is not part of the default installation
- `/proc` : Holds files that contain information about running processes and system resources
- `/root` : The home directory of the System Administrator, or the ‘root’ user
- `/sbin` : Contains binary executables / commands used by the system administrator
- `/srv` : Provides a consistent location for storing data for specific services
- `/tmp` : A temporary location for storing files or data
- `/usr` : Is the directory where user programs and data are stored and shared
- `/usr/bin` : Contains binary executable files for users
- `/usr/lib` : Holds shared library files to support executables in `/usr/bin` and `/usr/sbin`
- `/usr/local` : Contains users programs that are installed locally from source code
- `/usr/sbin` : The directory for non-essential system administration binary executables
- `/var` : Holds variable data files which are expected to grow under normal circumstances
- `/var/lib` : Contains dynamic state information that programs modify while they run
- `/var/log` : Stores log files from a range of programs and services
- `/var/spool` : Contains files that are held (spooled) for later processing
- `/var/tmp` : A temporary store for data that needs to be held between reboots (unlike `/tmp`)