THAKUR COLLEGE OF SCIENCE & COMMERCE

NAAC Accredited with Grade "A" (3rd Cycle)

THAKUR TRUSTS

ISO 9001 : 2015 Certified

Degree College

Computer Journal

CERTIFICATE

SEMESTER III UID No. 2020858

Class SYBSC CS Roll No. 4334 Year 2021-2022

This is to certify that the work entered in this journal is the work of Mst. / Ms. SHAIKH KAYSAN RAZAUDDIN who has worked for the year 2021-2022 in the Computer Laboratory.

Teacher In-Charge

Head of Department

Date :

Examiner

INDEX

Sr. No.	Practical Name	Date
1.	Setting up Raspberry Pi.	**09/07/2021**
2.	Use the Raspberry Pi operating system Raspbian and some of its software, and how to adjust some key settings to specific needs.	**16/07/2021**
3.	Connect the Raspberry Pi Camera Module to Raspberry Pi and take pictures, record video, and apply image effects.	**23/07/2021**
4.	Make your own stop motion animation video using a Raspberry Pi, Python and a camera module to take pictures, controlled by a push button connected to the Pi's GPIO pins.	**30/07/2021**
5.	Use loops to draw a race track and create a racing turtle game.	**06/08/2021**
6.	Use Python to create a program that generates a random story, based on what the user types in.	**13/08/2021**
7.	Installing LAMP stack on Raspberry Pi and testing accessing web application from outside.	**20/08/2021**
8.	Node RED: Connect LED to Internet of Things.	**27/09/2021**
9.	GPIO: LED Grid Module: Program the 8X8 Grid with Different Formulas.	**03/09/2021**

Practical No: 01

Step 1 Introduction

Here you'll learn about your Raspberry Pi, what things you need to use it, and how to set it up.

We also have a three-week online course available **on the FutureLearn platform (**http://rpf.io/rpi-fi), and a **Raspberry Pi forum (**https://www.raspberrypi.org/forums), including the **Beginners (**https://www.raspberrypi.org/forums/viewforum.php?f=91） section, if you want to ask questions and get support from the Raspberry Pi community.

Step 2

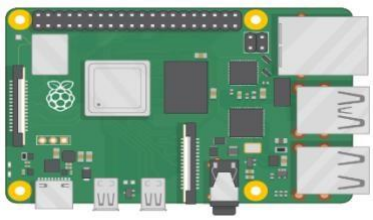
What you will need

Which Raspberry Pi?

There are several **models of Raspberry Pi** (https://www.raspberrypi.org/products/), and for most people Raspberry Pi 4 Model B is the one to choose. Raspberry Pi 4 Model B is the newest, fastest, and easiest to use.

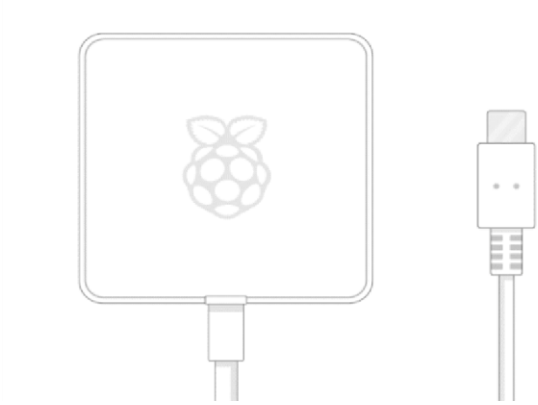
Raspberry Pi 4 comes with 2GB, 4GB, or 8GB of RAM. For most educational purposes and hobbyist projects, and for use as a desktop computer, 2GB is enough.Raspberry Pi Zero, Raspberry Pi Zero W, and Raspberry Pi Zero WH are smaller and require less power, so they're useful for portable projects such as robots. It's generally easier to start a project with Raspberry Pi 4, and to move to Raspberry Pi Zero when you have a working prototype that a smaller Raspberry Pi would be useful for.

If you want to buy a Raspberry Pi, head to **rpf.io/products (**https://rpf.io/products).



A power supply

To connect to a power socket, all Raspberry Pi models have a USB port (the same found on many mobile phones): either USB-C for Raspberry Pi 4, or micro USB for Raspberry Pi 3, 2, and 1.



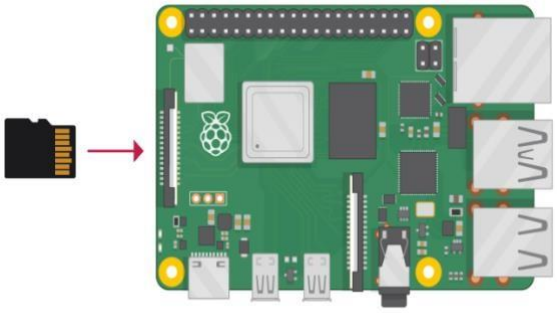
You need a power supply that provides:

- At least 3.0 amps for Raspberry Pi 4

At least 2.5 amps for Raspberry Pi 3

A microSD card

Your Raspberry Pi needs an SD card to store all its files and the Raspberry Pi OS operating system.



You need a microSD card with a capacity of at least 8GB.

Many sellers supply SD cards for Raspberry Pi that are already set up with Raspberry Pi OS and ready to go.

A keyboard and a mouse

To start using your Raspberry Pi, you need a USB keyboard and a USB mouse.

Once you've set up your Raspberry Pi, you can use a Bluetooth keyboard and mouse, but you'll need a USB keyboard and mouse for the first setup.

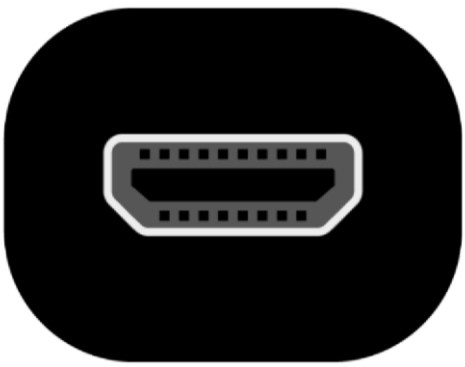
A TV or computer screen

To view the Raspberry Pi OS desktop environment, you need a screen, and a cable to link the screen and your Raspberry Pi. The screen can be a TV or a computer monitor. If the screen has built-in speakers, Raspberry Pi is able to use these to play sound.

HDMI

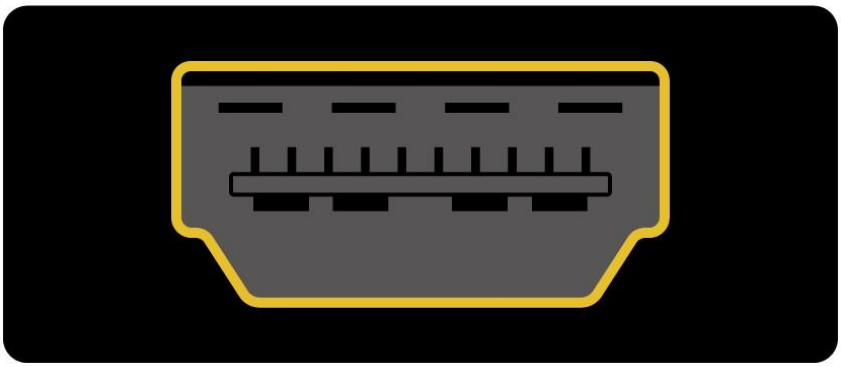
Your Raspberry Pi has an HDMI output port that is compatible with the HDMI port of most modern TVs and computer monitors. Many computer monitors may also have DVI or VGA ports.

Raspberry Pi 4 has two micro HDMI ports, allowing you to connect two separate monitors.



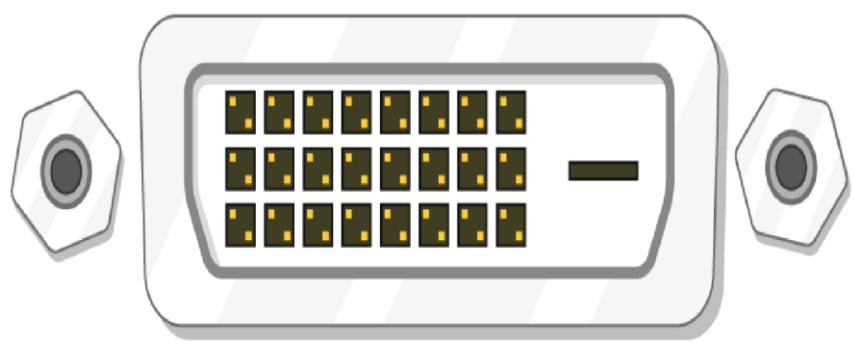
You need either a micro HDMI to HDMI cable, or a standard HDMI to HDMI cable plus a micro HDMI to HDMI adapter, to connect Raspberry Pi 4 to a screen.

Raspberry Pi 1, 2, and 3 have a single full-size HDMI port, so you can connect them to a screen using a standard HDMI to HDMI cable.



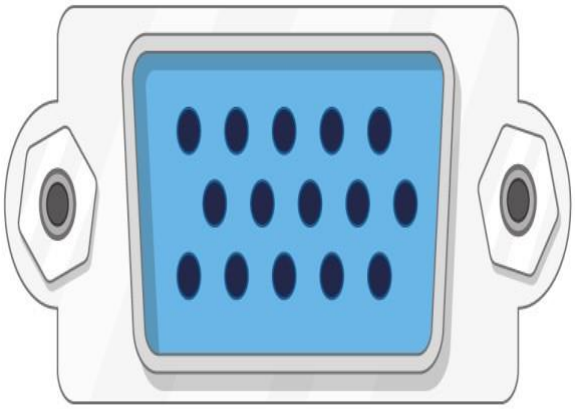
DVI

If your screen has a DVI port, you can connect your Raspberry Pi to it using an HDMI to DVI cable.

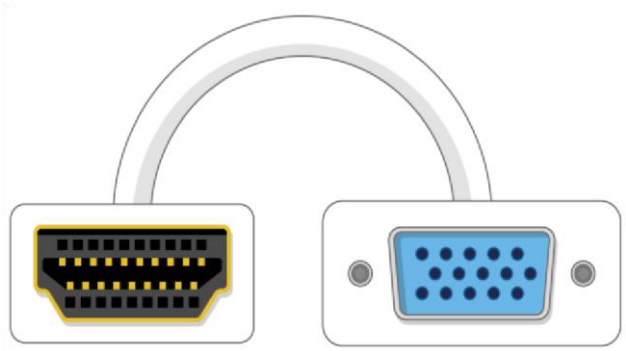


VGA

Some screens only have a VGA port.



To connect your Raspberry Pi to such a screen, you can use an HDMI to VGA adapter.



Optional extras

A case

You may want to put your Raspberry Pi in a case. This is not essential, but it will provide protection for your Raspberry Pi. If you'd like, you can use the official case for Raspberry Pi 4 or Raspberry Pi Zero or Raspberry Pi Zero W.

Headphones or speakers

The large Raspberry Pi models (but not Raspberry Pi Zero or Raspberry Pi Zero W) have a standard audio port like the one on a smartphone or MP3 player. If you want to, you can connect your headphones or speakers so that your Raspberry Pi can play sound. If the screen you're connecting your Raspberry Pi to has built-in speakers, Raspberry Pi can play sound through these.

An Ethernet cable

The large Raspberry Pi models (but not Raspberry Pi Zero or Raspberry Pi Zero W) have a standard Ethernet port to connect them to the internet; to connect Raspberry Pi Zero to the internet, you need a USB to Ethernet adapter.

Raspberry Pi 4, Raspberry Pi 3, and Raspberry Pi Zero W can also be wirelessly connected to the internet.

Set up your SD card

If you have an SD card that doesn't have the Raspberry Pi OS operating system on it yet, or if you want to reset your Raspberry Pi, you can easily install Raspberry Pi OS yourself. To do so, you need a computer that has an SD card port — most laptop and desktop computers have one.

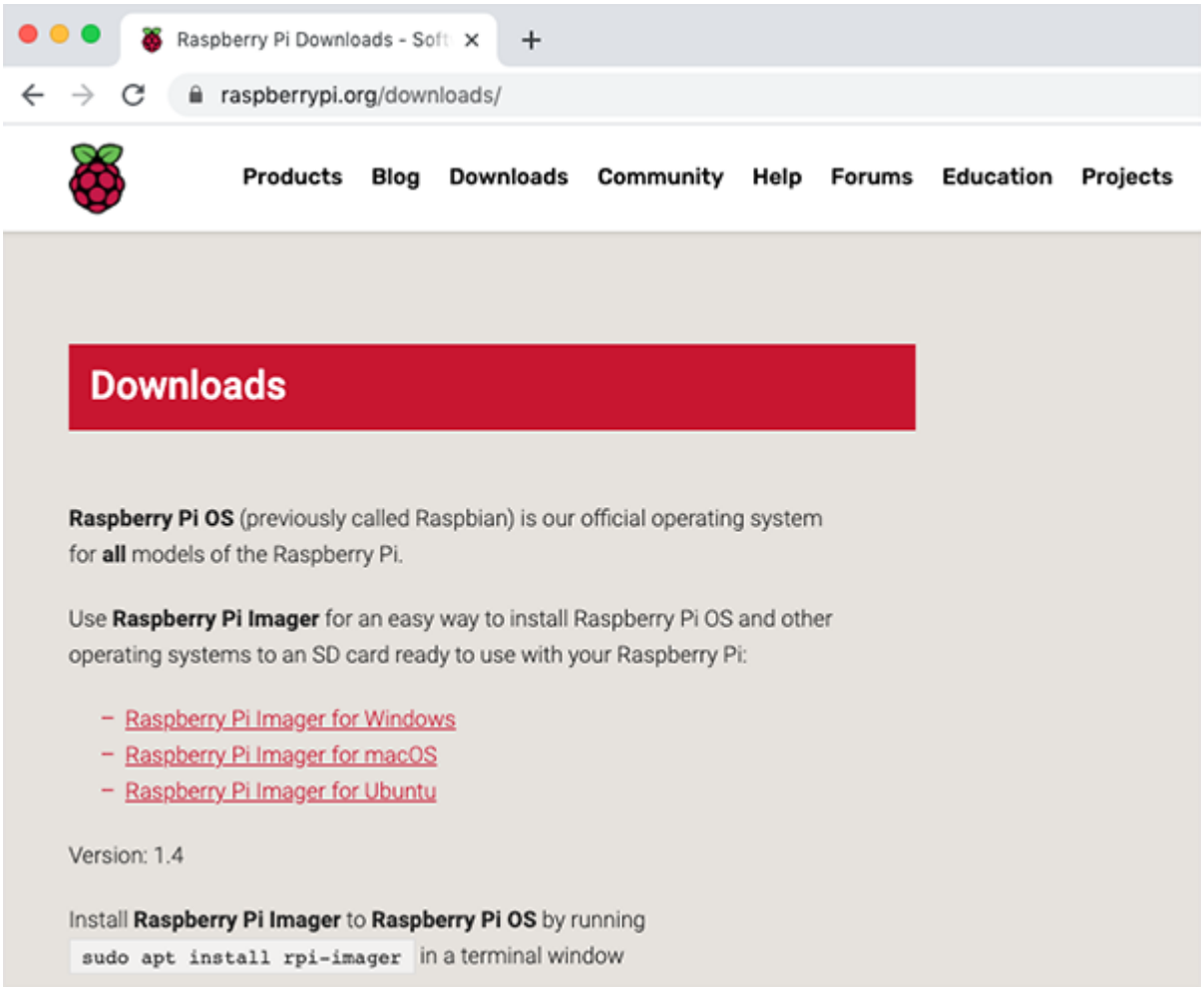
The Raspberry Pi OS operating system via the Raspberry Pi Imager

Using the Raspberry Pi Imager is the easiest way to install Raspberry Pi OS on your SD card.

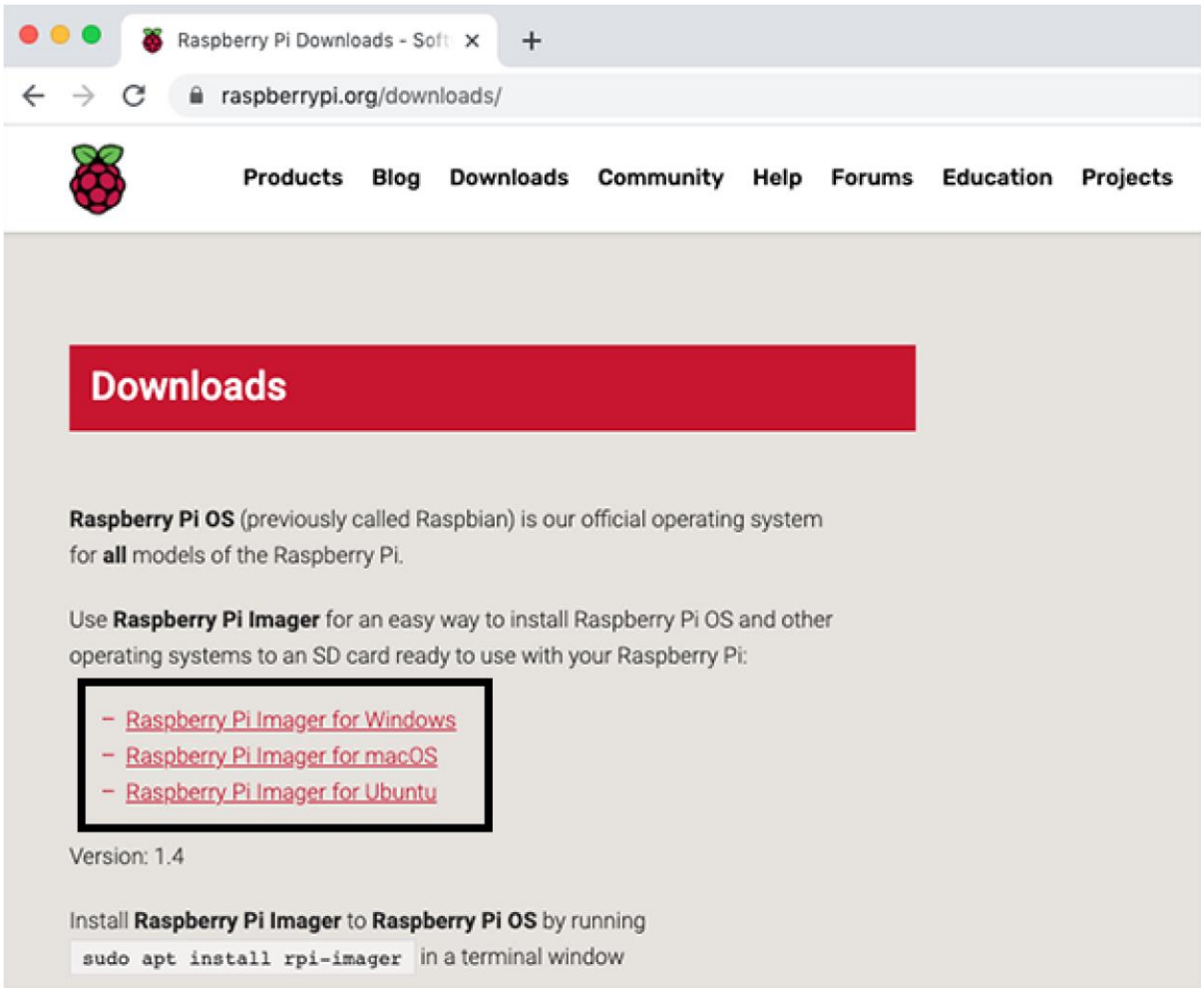
Note: More advanced users looking to install a particular operating system should use this guide to installing operating system images.

Download and launch the Raspberry Pi Imager ○

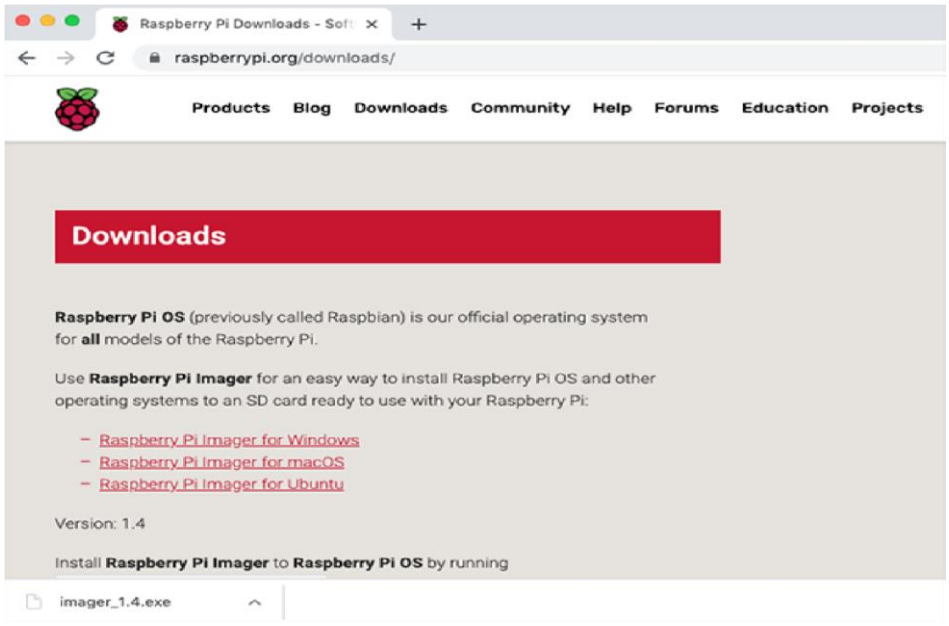
Visit the Raspberry Pi downloads page



- Click on the link for the Raspberry Pi Imager that matches your operating system



- When the download finishes, click it to launch the installer



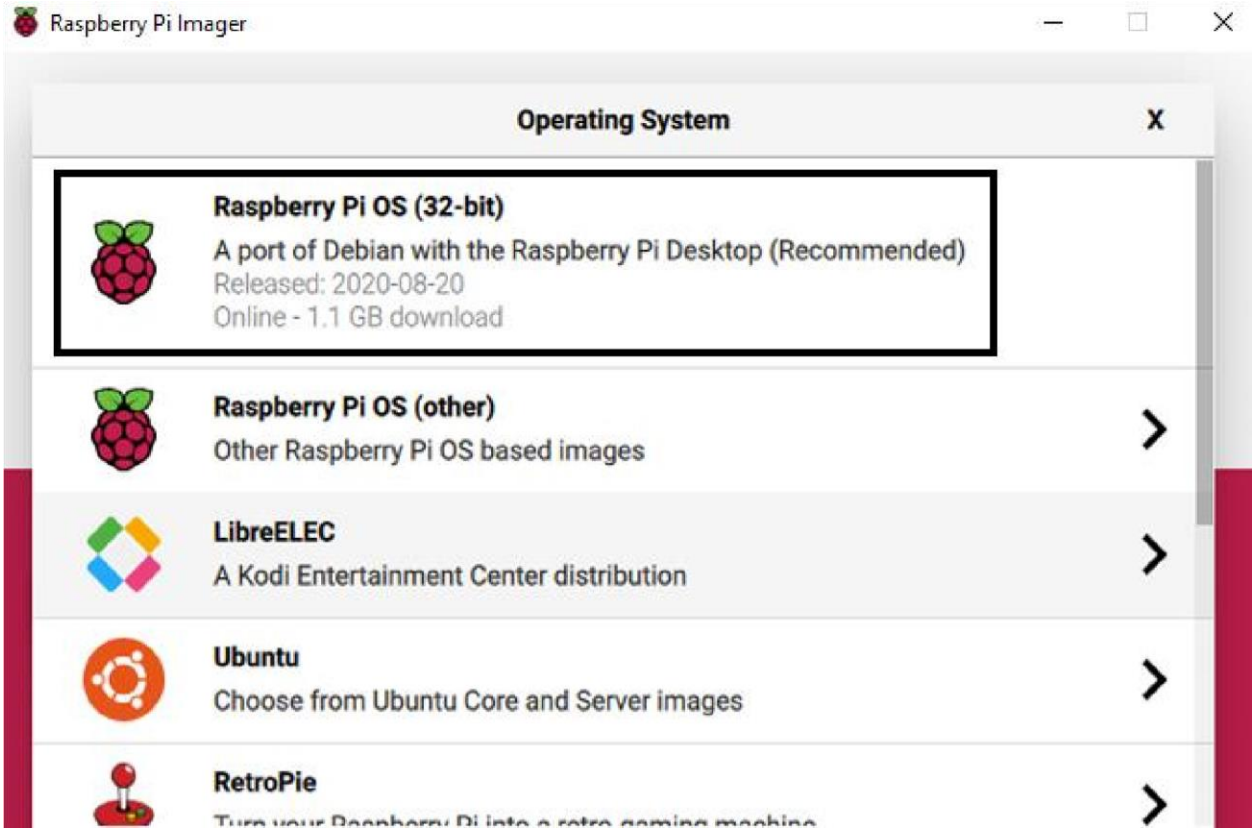
Using the Raspberry Pi Imager

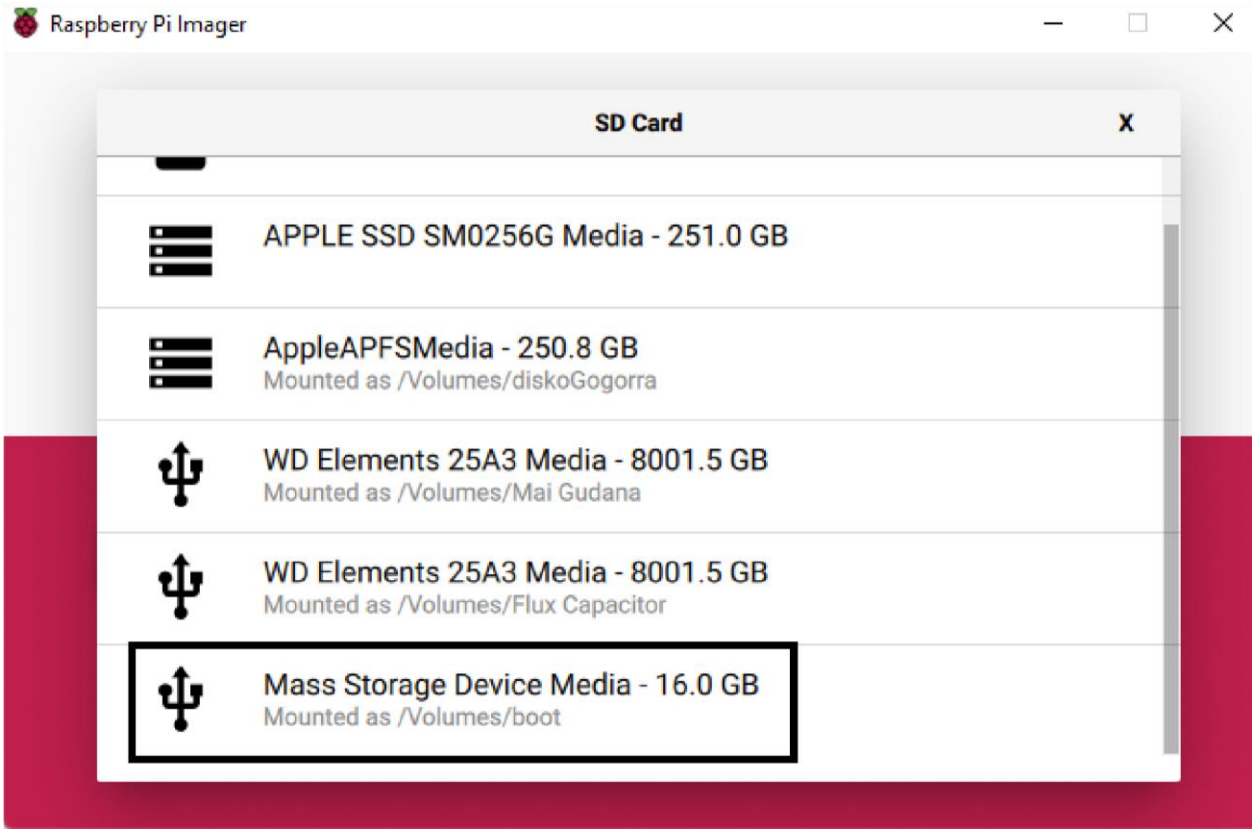
Anything that's stored on the SD card will be overwritten during formatting. If your SD card currently has any files on it, e.g. from an older version of Raspberry Pi OS, you may wish to back up these files first to prevent you from permanently losing them.

When you launch the installer, your operating system may try to block you from running it. For example, on Windows I receive the following message:

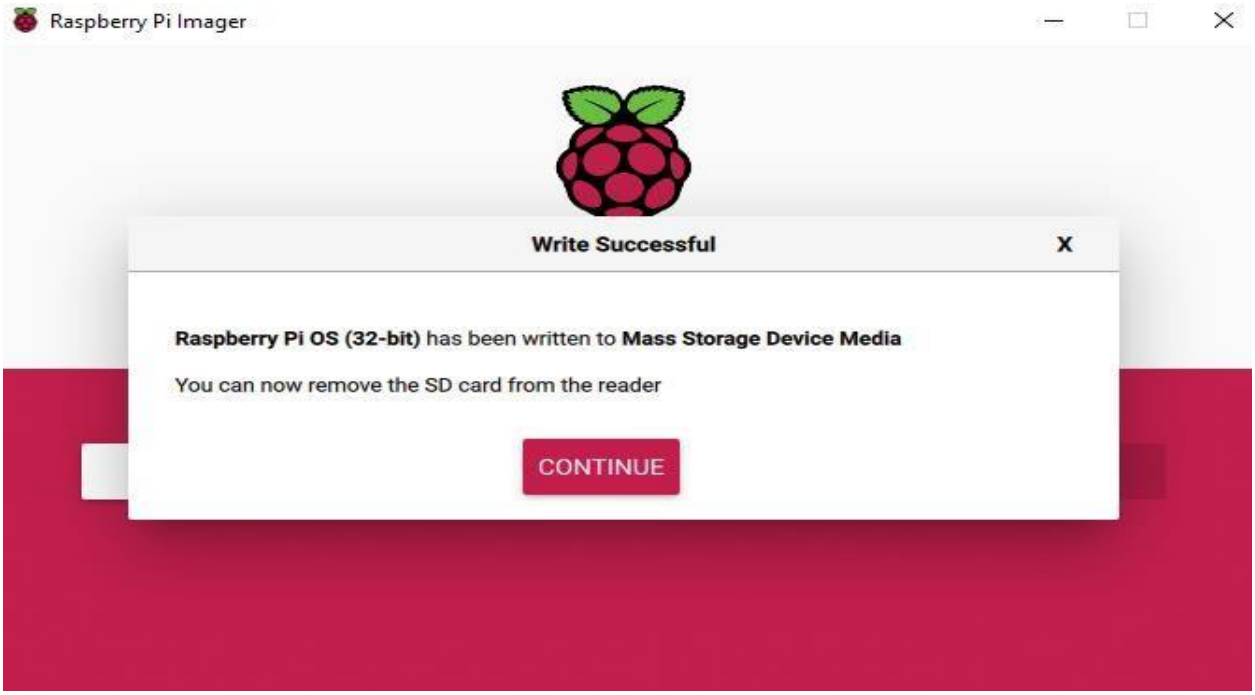
- If this pops up, click on `More info` and then `Run anyway` ○ Follow the instructions to install and run the Raspberry Pi Imager ○ Insert your SD card into the computer or laptop SD card slot ○ In the Raspberry Pi Imager, select the OS that you want to install and the SD card you would like to install it on

Note: You will need to be connected to the internet the first time for the the Raspberry Pi Imager to download the OS that you choose. That OS will then be stored for future offline use. Being online for later uses means that the Raspberry Pi imager will always give you the latest version.


Raspberry Pi Imager
Operating System
X
Raspberry Pi OS (32-bit)
A port of Debian with the Raspberry Pi Desktop (Recommended)
Released: 2020-08-20
Online - 1.1 GB download
Raspberry Pi OS (other)
Other Raspberry Pi OS based images
LibreELEC
A Kodi Entertainment Center distribution
Ubuntu
Choose from Ubuntu Core and Server images
RetroPie

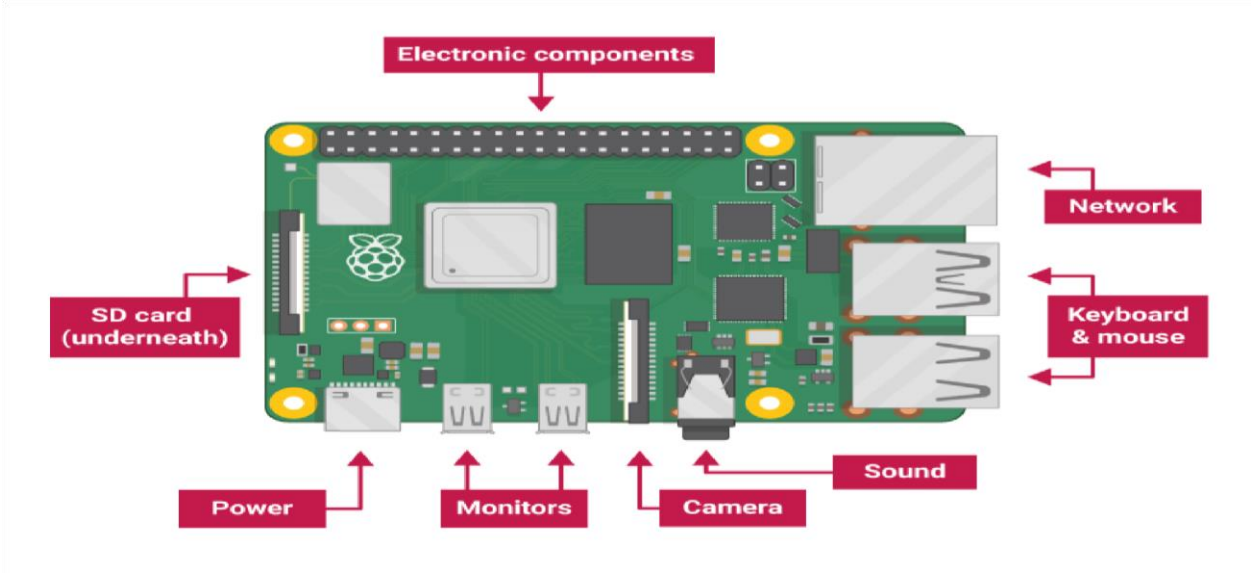

Raspberry Pi Imager
SD Card
X
APPLE SSD SM0256G Media - 251.0 GB
AppleAPFSMedia - 250.8 GB
Mounted as /Volumes/diskoGogorra
WD Elements 25A3 Media - 8001.5 GB
Mounted as /Volumes/Mai Gudana
WD Elements 25A3 Media - 8001.5 GB
Mounted as /Volumes/Flux Capacitor
Mass Storage Device Media - 16.0 GB
Mounted as /Volumes/boot

- Then simply click the `WRITE` button
- Wait for the Raspberry Pi Imager to finish writing
- Once you get the following message, you can eject your SD card

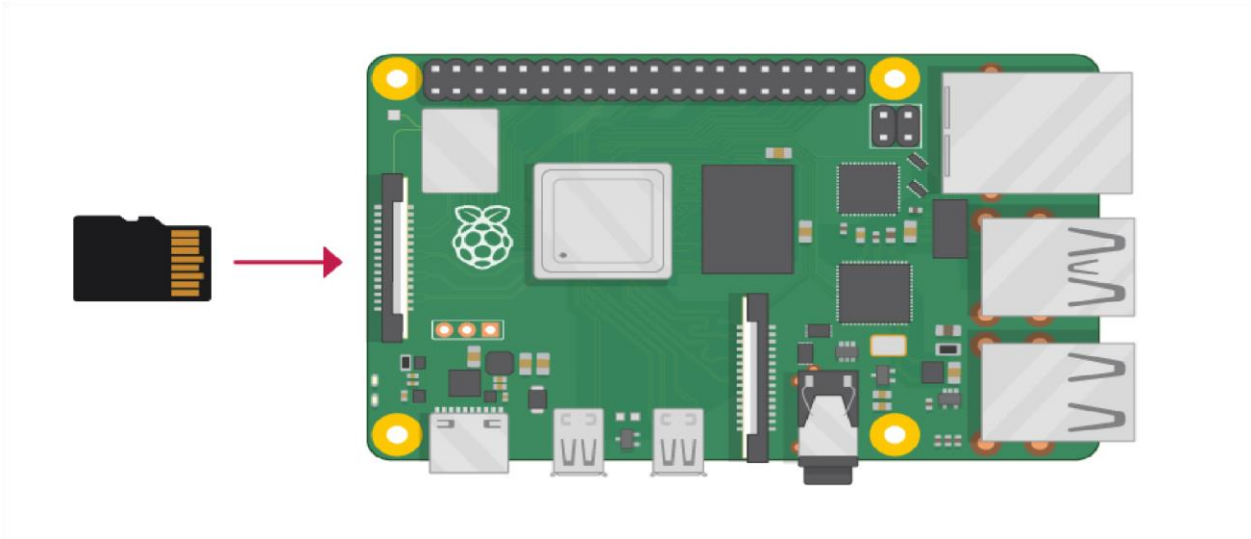


Connect your Raspberry Pi

Now get everything connected to your Raspberry Pi. It's important to do this in the right order, so that all your components are safe.

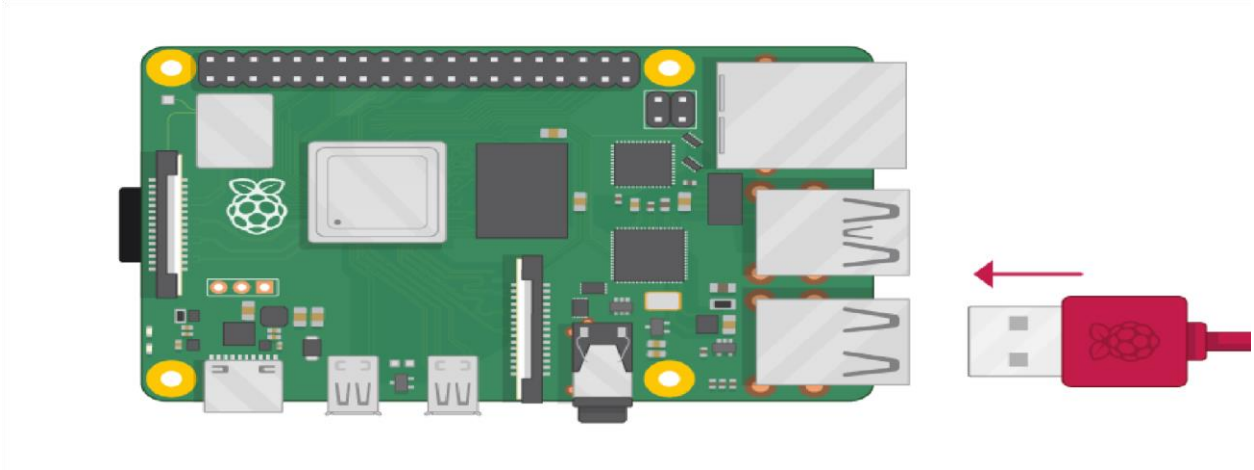


- Insert the SD card you've set up with Raspberry Pi OS into the microSD card slot on the underside of your Raspberry Pi.

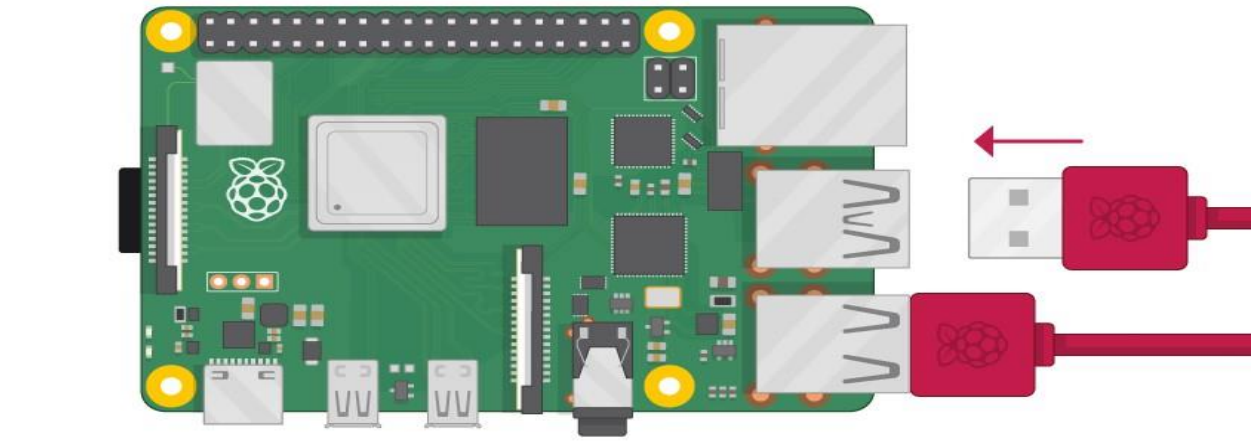


Note: Many microSD cards come inside a larger adapter — you can slide the smaller card out using the lip at the bottom.

- Find the USB connector end of your mouse's cable, and connect the mouse to a USB port on Raspberry Pi (it doesn't matter which port you use).



- Connect the keyboard in the same way.



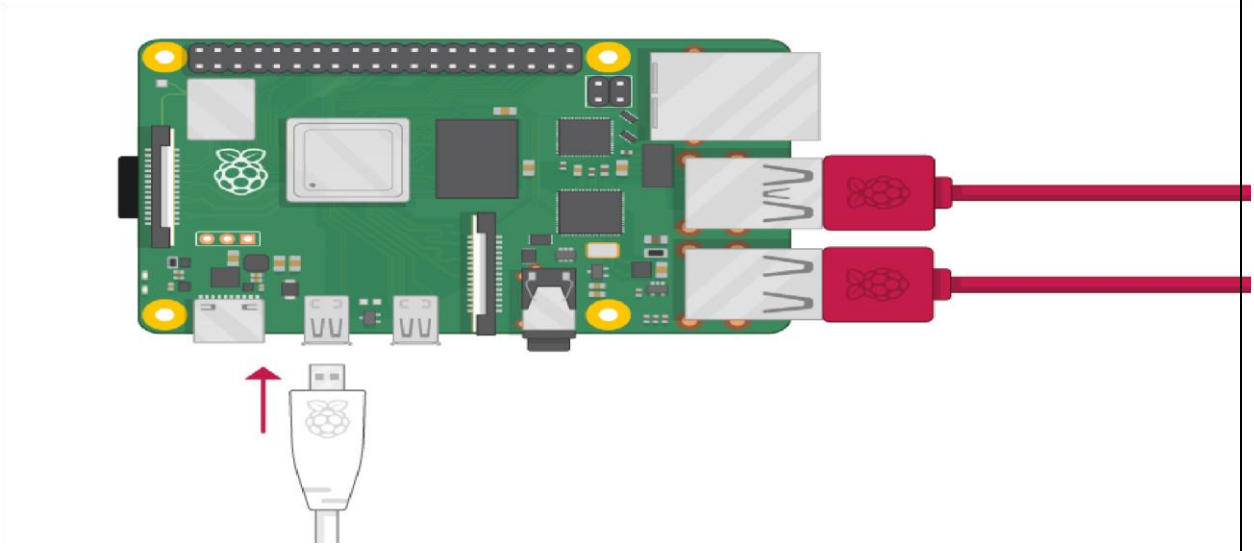
- Make sure your screen is plugged into a wall socket and switched on. ○ Look at the HDMI port(s) on your Raspberry Pi — notice that they have a flat side on top.

- Use a cable to connect the screen to Raspberry Pi's HDMI port — use an adapter if necessary.

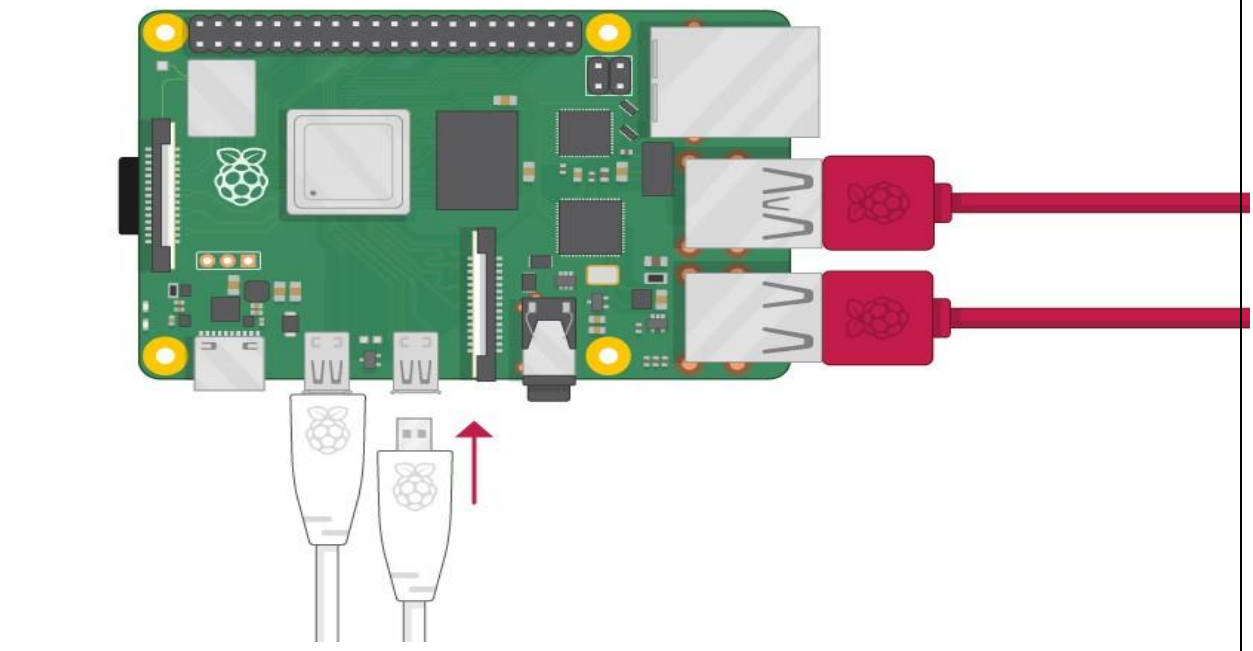
  Raspberry Pi 4

  Connect your screen to the first of Raspberry Pi 4's HDMI ports, labelled HDMI0.

  Note: Make sure you have used HDMI0 (nearest the power in port) rather than HDMI1.

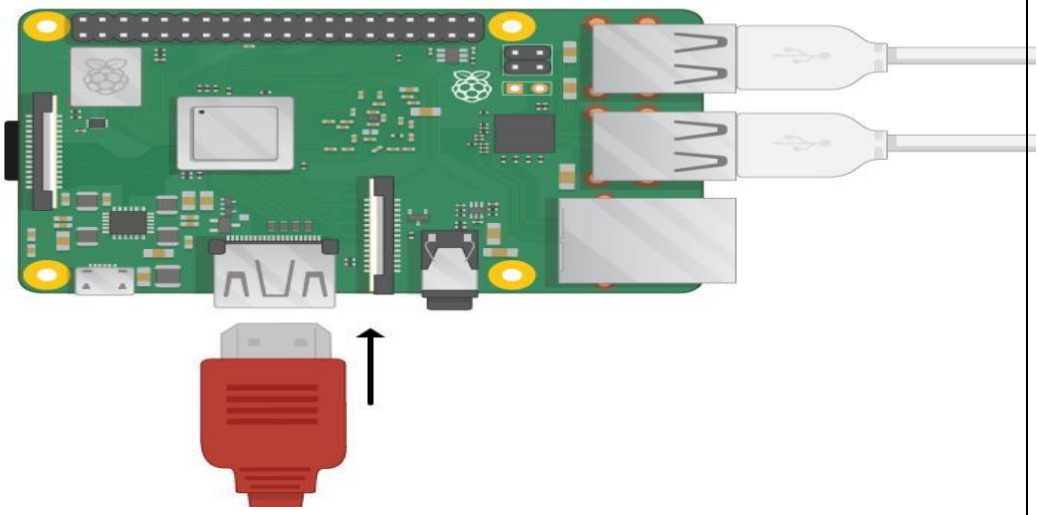


You can connect an optional second screen in the same way.



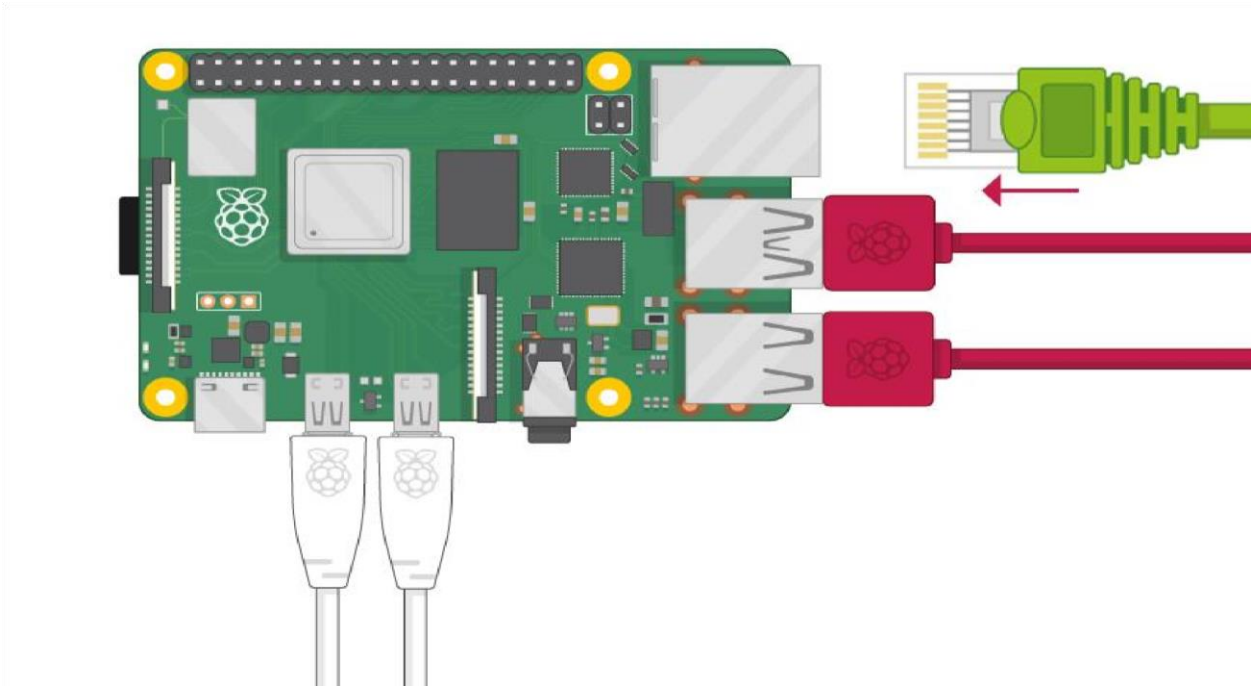
Raspberry Pi 1, 2, 3

Connect your screen to the single HDMI port.

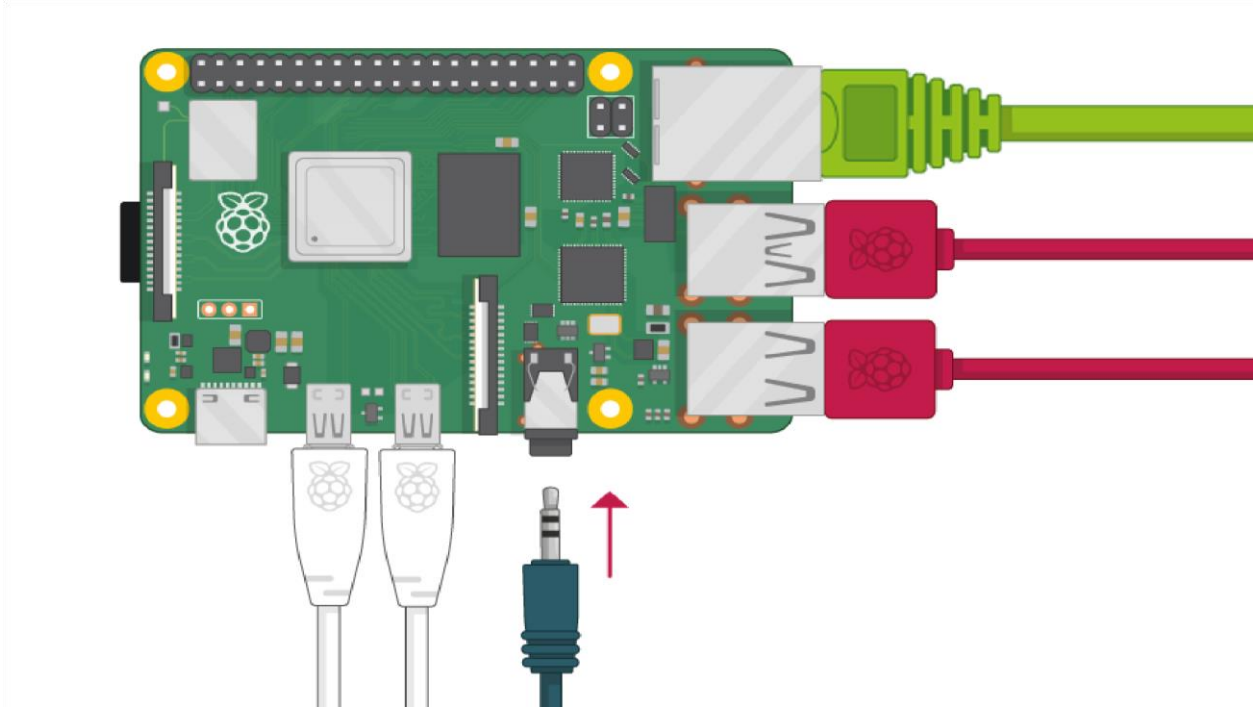


Note: Nothing will display on the screen, because your Raspberry Pi is not running yet.

- If you want to connect your Raspberry Pi to the internet via Ethernet, use an Ethernet cable to connect the Ethernet port on Raspberry Pi to an Ethernet socket on the wall or on your internet router. You don't need to do this if you want to use wireless connectivity, or if you don't want to connect to the internet.



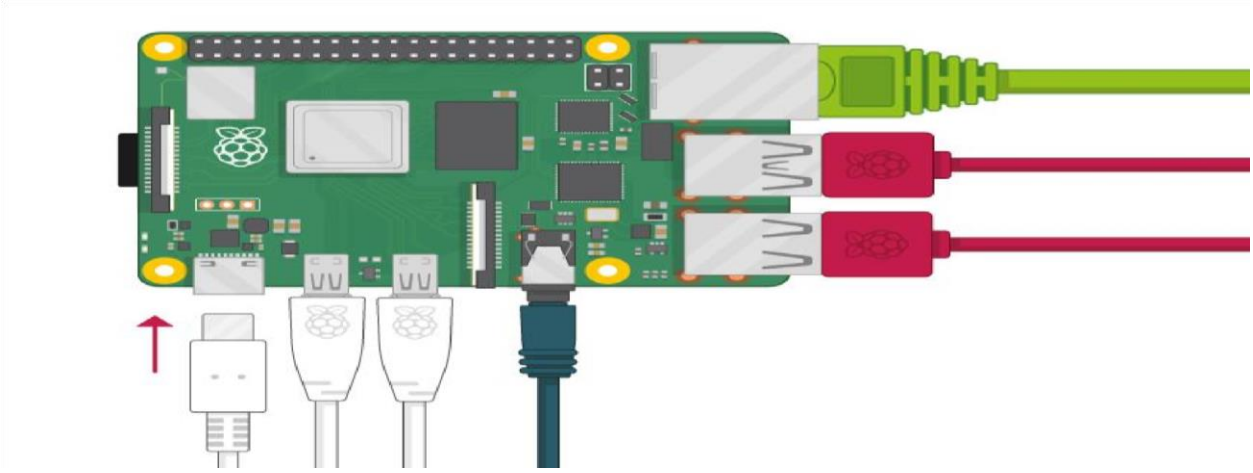
- If the screen you are using has speakers, sound will play through those. Alternatively, connect headphones or speakers to the audio port if you prefer.



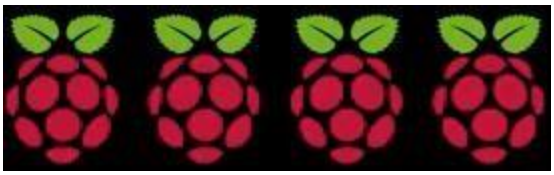
Start up your Raspberry Pi

Your Raspberry Pi doesn't have a power switch. As soon as you connect it to a power outlet, it will turn on.

- Plug the power supply into a socket and connect it to your Raspberry Pi's power port.



You should see a red LED light up on the Raspberry Pi, which indicates that Raspberry Pi is connected to power. As it starts up (this is also called booting), you will see raspberries appear in the top left-hand corner of your screen.



After a few seconds the Raspberry Pi OS desktop will appear.

Finishing the setup

When you start your Raspberry Pi for the first time, the Welcome to Raspberry Pi application will pop up and guide you through the initial setup.

- Click on Next to start the setup.
- Set your Country, Language, and Time zone, then click on Next again.

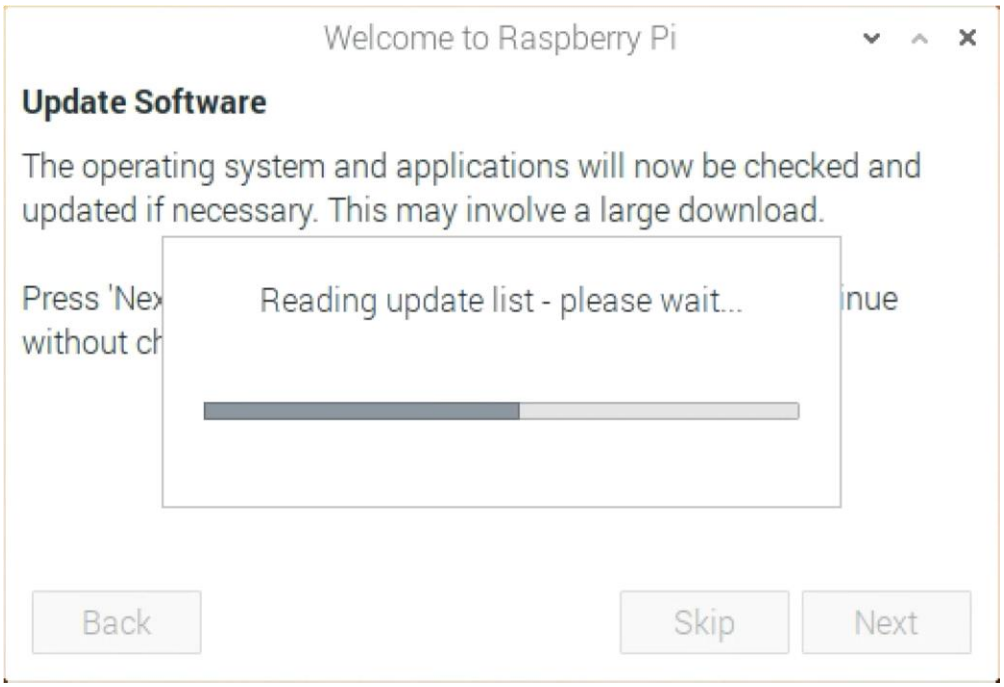
- Enter a new password for your Raspberry Pi and click on Next.

- Connect to your wireless network by selecting its name, entering the password, and clicking on Next.

Note: If your model of Raspberry Pi doesn't have wireless connectivity, you won't see this screen.

Note: Wait until the wireless connection icon appears and the correct time is shown before trying to update the software.

- Click on Next, and let the wizard check for updates to Raspberry Pi OS and install them (this might take a little while).



- Click on Restart to finish the setup.

Note: You will only need to reboot if that's necessary to complete an update.

Practical No: 02

Configuration:

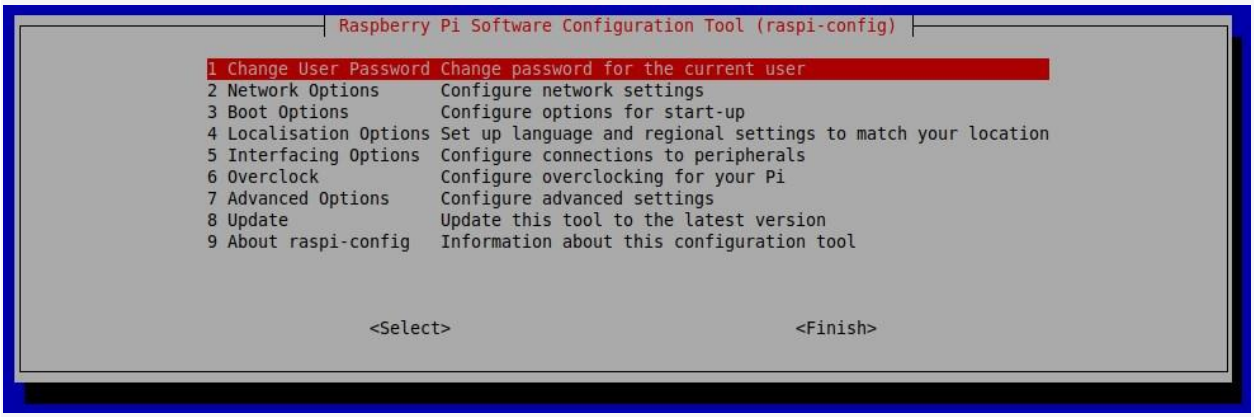
The `raspi-config` Tool

`raspi-config` is the Raspberry Pi configuration tool originally written by Alex Bradbury. To open the configuration tool, type the following on the command line:

```
sudo raspi-config
```

The `sudo` is required because you will be changing files that you do not own as the `pi` user.

You should then see a blue screen with options in a grey box:



Use the `up` and `down` arrow keys to move the highlighted selection between the options available.

Pressing the `right` arrow key will jump out of the Options menu and take you to the `<Select>` and `<Finish>` buttons. Pressing `left` will take you back to the options. Alternatively, you can use the `Tab` key to switch between these.

Generally speaking, `raspi-config` aims to provide the functionality to make the most common configuration changes. This may result in automated edits to `/boot/config.txt` and various standard Linux configuration files. Some options require a reboot to take effect. If you changed any of those, `raspi-config` will ask if you wish to reboot now when you select the `<Finish>` button.

List of Options

System Options

The system options submenu allows you to make configuration changes to various parts of the boot, login and networking process, along with some other system level changes.

Wireless LAN

Allows setting of the wireless LAN SSID and passphrase.

Audio

Specify the audio output destination.

Password

The default user on Raspberry Pi OS is `pi` with the password `raspberry`. You can change that here.

Read about other users.

Hostname

Set the visible name for this Pi on a network.

Boot / Auto login

From this submenu you can select whether to boot to console or desktop and whether you need to log in or not. If you select automatic login, you will be logged in as the `pi` user.

Network at Boot

Use this option to wait for a network connection before letting boot proceed.

Splash Screen

Enable or disable the splash screen displayed at boot time

Power LED

If the model of Pi permits it, you can change the behaviour of the power LED using this option.

Display Options

Resolution

Define the default HDMI/DVI video resolution to use when the system boots without a TV or monitor being connected. This can have an effect on RealVNC if the VNC option is enabled.

Underscan

Old TV sets had a significant variation in the size of the picture they produced; some had cabinets that overlapped the screen. TV pictures were therefore given a black border so that none of the picture was lost; this is called overscan. Modern TVs and monitors don't need the border, and the

signal doesn't allow for it. If the initial text shown on the screen disappears off the edge, you need to enable overscan to bring the border back.

Any changes will take effect after a reboot. You can have greater control over the settings by editing config.txt.
On some displays, particularly monitors, disabling overscan will make the picture fill the whole screen and correct the resolution. For other displays, it may be necessary to leave overscan enabled and adjust its values.

Pixel Doubling

Enable/disable 2x2 pixel mapping.

Composite Video

On the Raspberry Pi4, enable composite video. On models prior to the Raspberry Pi4, composite video is enabled by default so this option is not displayed.

Screen Blanking

Enable or disable screen blanking.

Interfacing Options

In this submenu there are the following options to enable/disable: Camera, SSH, VNC, SPI, I2C, Serial, 1wire, and Remote GPIO.

Camera

Enable/disable the CSI camera interface.

SSH

Enable/disable remote command line access to your Pi using SSH.

SSH allows you to remotely access the command line of the Raspberry Pi from another computer. SSH is disabled by default. Read more about using SSH on the SSH documentation page. If connecting your Pi directly to a public network, you should not enable SSH unless you have set up secure passwords for all users.

VNC

Enable/disable the RealVNC virtual network computing server.

SPI

Enable/disable SPI interfaces and automatic loading of the SPI kernel module, needed for products such as PiFace.

I2C

Enable/disable I2C interfaces and automatic loading of the I2C kernel module.

Serial

Enable/disable shell and kernel messages on the serial connection.

1-wire

Enable/disable the Dallas 1-wire interface. This is usually used for DS18B20 temperature sensors.

Remote GPIO

Enable or disable remote access to the GPIO pins.

Performance Options

Overclock

On some models it is possible to overclock your Raspberry Pi's CPU using this tool. The overclocking you can achieve will vary; overclocking too high may result in instability. Selecting this option shows the following warning:

Be aware that overclocking may reduce the lifetime of your Raspberry Pi. If overclocking at a certain level causes system instability, try a more modest overclock. Hold down the Shift key during boot to temporarily disable overclocking.

GPU Memory

Change the amount of memory made available to the GPU.

Overlay File System

Enable or disable a read-only filesystem

Fan

Set the behaviour of a GPIO connected fan

Localisation Options

The localisation submenu gives you these options to choose from: keyboard layout, time zone, locale, and wireless LAN country code.

Locale

Select a locale, for example `en_GB.UTF-8 UTF-8`.

Time Zone

Select your local time zone, starting with the region, e.g. Europe, then selecting a city, e.g. London. Type a letter to skip down the list to that point in the alphabet.

Keyboard

This option opens another menu which allows you to select your keyboard layout. It will take a long time to display while it reads all the keyboard types. Changes usually take effect immediately, but may require a reboot.

WLAN Country

This option sets the country code for your wireless network.

Advanced Options

Expand Filesystem

This option will expand your installation to fill the whole SD card, giving you more space to use for files.

You will need to reboot the Raspberry Pi to make this available.

WARNING

There is no confirmation: selecting the option begins the partition expansion immediately.

GL Driver

Enable/disable the experimental GL desktop graphics drivers.

GL (Full KMS)

Enable/disable the experimental OpenGL Full KMS (kernel mode setting) desktop graphics driver.

GL (Fake KMS)

Enable/disable the experimental OpenGL Fake KMS desktop graphics driver.

Legacy

Enable/disable the original legacy non-GL VideoCore desktop graphics driver.

Compositor

Enable/Display the xcompmgr composition manager

Network Interface Names

Enable or disable predictable network interface names.

Network Proxy Settings

Configure the network's proxy settings.

Boot Order

On the Raspberry Pi4, you can specify whether to boot from USB or network if the SD card isn't inserted.

See this page for more information.

Bootloader Version

On the Raspberry Pi4, you can tell the system to use the very latest boot ROM software, or default to the factory default if the latest version causes problems.

Update

Update this tool to the latest version.

About raspi-config

Selecting this option shows the following text:

```
This tool provides a straightforward way of doing initial configuration of
the Raspberry Pi.
Although it can be run at any time, some of the options may have difficulties
if you have heavily customised your installation.
```

Finish

Use this button when you have completed your changes. You will be asked whether you want to reboot or not. When used for the first time, it's best to reboot. There will be a delay in rebooting if you have chosen to resize your SD card.

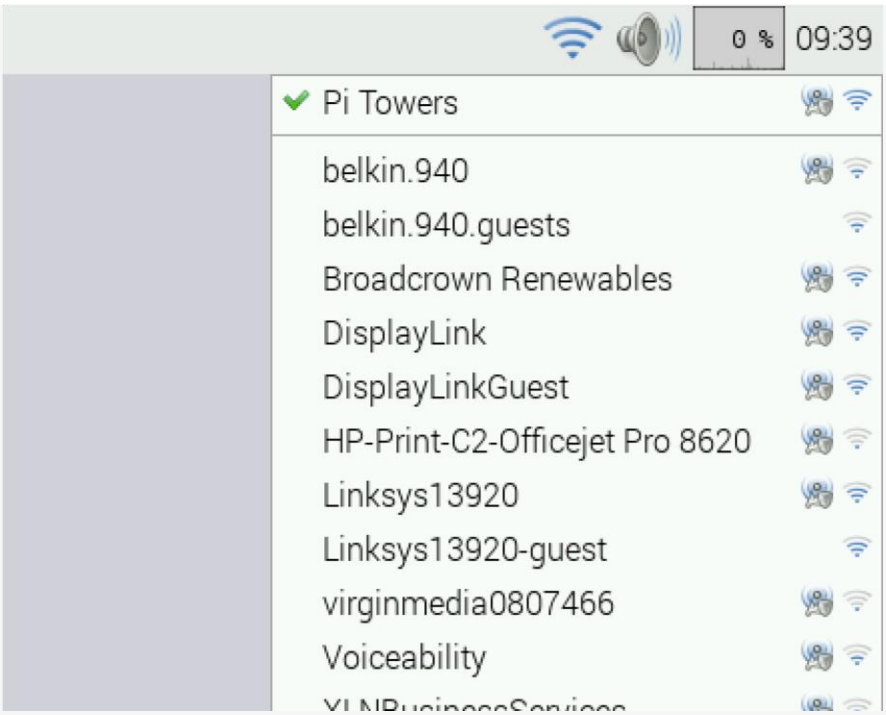
Configuring Networking

A GUI is provided for setting up wireless connections in Raspberry Pi OS within the Raspberry Pi Desktop. However if you are not using the Raspberry Pi Desktop, you can set up wireless networking from the command line.

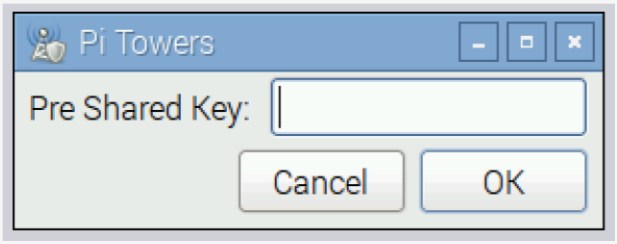
Using the Desktop

Wireless connections can be made via the network icon at the right-hand end of the menu bar. If you are using a Pi with built-in wireless connectivity, or if a wireless dongle is plugged in, left-clicking this icon will bring up a list of available wireless networks, as shown below. If no networks are found, it will show the message 'No APs found - scanning…'. Wait a few seconds without closing the menu, and it should find your network.

Note that on Raspberry Pi devices that support the 5GHz band (Pi3B+, Pi4, CM4, Pi400), wireless networking is disabled for regulatory reasons, until the country code has been set. To set the country code, open the `Raspberry Pi Configuration` application from the Preferences Menu, select **Localisation** and set the appropriate code.



The icons on the right show whether a network is secured or not, and give an indication of its signal strength. Click the network that you want to connect to. If it is secured, a dialogue box will prompt you to enter the network key:



Enter the key and click **OK**, then wait a couple of seconds. The network icon will flash briefly to show that a connection is being made. When it is ready, the icon will stop flashing and show the signal strength.

Using the Command Line

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

Using raspi-config

The quickest way to enable wireless networking is to use the command line `raspi-config` tool.

```
sudo raspi-config
```

Select the **Localisation Options** item from the menu, then the **Change wireless country** option. On a fresh install, for regulatory purposes, you will need to specify the country in which the device is being used. Then set the SSID of the network, and the passphrase for the network. If you do not know the SSID of the network you want to connect to, see the next section on how to list available networks prior to running `raspi-config`.

Note that `raspi-config` does not provide a complete set of options for setting up wireless networking; you may need to refer to the extra sections below for more details if `raspi-config` fails to connect the Pi to your requested network.

Getting Wireless LAN Network Details

To scan for wireless networks, use the command `sudo iwlist wlan0 scan`. This will list all available wireless networks, along with other useful information. Look out for:

1. 'ESSID:"testing"' is the name of the wireless network.
2. 'IE: IEEE 802.11i/WPA2 Version 1' is the authentication used. In this case it's WPA2, the newer and more secure wireless standard which replaces WPA. This guide should work for WPA or WPA2, but may not work for WPA2 enterprise. You'll also need the password for the wireless network. For most home routers, this is found on a sticker on the back of the router. The ESSID (ssid) for the examples below is `testing` and the password (psk) is `testingPassword`.

Adding the Network Details to your Raspberry Pi

Open the `wpa-supplicant` configuration file in nano:

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

Go to the bottom of the file and add the following:

```
network={
ssid="testing"
psk="testingPassword" }
```

The password can be configured either as the ASCII representation, in quotes as per the example above, or as a pre-encrypted 32 byte hexadecimal number. You can use the `wpa_passphrase` utility to generate an encrypted PSK. This takes the SSID and the password, and generates the encrypted PSK. With the example from above, you can generate the PSK with `wpa_passphrase "testing"`. Then you will be asked for the password of the wireless network (in this case `testingPassword`). The output is as follows:

```
  network={
        ssid="testing"
#psk="testingPassword"

psk=131e1e221f6e06e3911a2d11ff2fac9182665c004de85300f9cac208a6a80531    }
```

Note that the plain text version of the code is present, but commented out. You should delete this line from the final `wpa_supplicant` file for extra security.

The `wpa_passphrase` tool requires a password with between 8 and 63 characters. To use a more complex password, you can extract the content of a text file and use it as input for `wpa_passphrase`. Store the password in a text file and input it to `wpa_passphrase` by calling `wpa_passphrase "testing" < file_where_password_is_stored`. For extra security, you should delete the `file_where_password_is_stored` afterwards, so there is no plain text copy of the original password on the system.

To use the `wpa_passphrase` encrypted PSK, you can either copy and paste the encrypted PSK into `wpa_supplicant.conf` file, or redirect the tool's output to the configuration file in one of two ways:

- Either change to `root` by executing `sudo su`, then call `wpa_passphrase "testing" >> /etc/wpa_supplicant/wpa_supplicant.conf` and enter the testing password when asked
- Or use `wpa_passphrase "testing" | sudo tee -a /etc/wpa_supplicant/wpa_supplicant.conf > /dev/null` and enter the testing password when asked; the redirection to `/dev/null` prevents `tee` from **also** outputting to the screen (standard output).

If you want to use one of these two options, **make sure you use >>, or use `-a` with `tee`** — either will **append** text to an existing file. Using a single chevron `>`, or omitting `-a` when using `tee`, will erase all contents and **then** append the output to the specified file.

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

Reconfigure the interface with `wpa_cli -i wlan0 reconfigure`.

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

On the Raspberry Pi 3B+ and Raspberry Pi 4B, you will also need to set the country code, so that the 5GHz networking can choose the correct frequency bands. You can do this using the `raspi-config` application: select the 'Localisation Options' menu, then 'Change Wi-Fi Country'. Alternatively, you

can edit the `wpa_supplicant.conf` file and add the following. (Note: you need to replace 'GB' with the 2 letter ISO code of your country. See Wikipedia for a list of 2 letter ISO 3166-1 country codes.)

```
country=GB
```

Note that with the latest Buster Raspberry Pi OS release, you must ensure that the `wpa_supplicant.conf` file contains the following information at the top:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev update_config=1
country=<Insert 2 letter ISO 3166-1 country code here>
```

Using Unsecured Networks

If the network you are connecting to does not use a password, the `wpa_supplicant` entry for the network will need to include the correct `key_mgmt` entry. e.g.

```
network={
ssid="testing"
key_mgmt=NONE }
```

WARNING

You should be careful when using unsecured wireless networks.

Hidden Networks

If you are using a hidden network, an extra option in the `wpa_supplicant file`, `scan_ssid`, may help connection.

```
network={
    ssid="yourHiddenSSID"
scan_ssid=1
    psk="Your_wireless_network_password" }
```

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

Adding Multiple Wireless Network Configurations

On recent versions of Raspberry Pi OS, it is possible to set up multiple configurations for wireless networking. For example, you could set up one for home and one for school.

For example

```
network={
    ssid="SchoolNetworkSSID"
psk="passwordSchool"     id_str="school"
}
 network={
    ssid="HomeNetworkSSID"
psk="passwordHome"     id_str="home"
}
```

If you have two networks in range, you can add the priority option to choose between them. The network in range, with the highest priority, will be the one that is connected.

```
network={
```

```
    ssid="HomeOneSSID"
psk="passwordOne"
priority=1     id_str="homeOne"
}
 network={
    ssid="HomeTwoSSID"
psk="passwordTwo"
priority=2     id_str="homeTwo"
}
```

The DHCP Daemon

The Raspberry Pi uses `dhcpcd` to configure TCP/IP across all of its network interfaces.

The `dhcpcd` daemonis intended to be an all-in-one ZeroConf client for UNIX-like systems. This includes assigning each interface an IP address, setting netmasks, and configuring DNS resolution via the Name Service Switch (NSS) facility.

By default, Raspberry Pi OS attempts to automatically configure all network interfaces by DHCP, falling back to automatic private addresses in the range 169.254.0.0/16 if DHCP fails. This is consistent with the behaviour of other Linux variants and of Microsoft Windows.

Static IP Addresses

If you wish to disable automatic configuration for an interface and instead configure it statically, add the details to `/etc/dhcpcd.conf`. For example:

```
interface eth0
static ip_address=192.168.0.4/24 static
routers=192.168.0.254
static domain_name_servers=192.168.0.254 8.8.8.8
```

You can find the names of the interfaces present on your system using the `ip link` command.

Note that if you have several Raspberry Pis connected to the same network, you may find it easier instead to set address reservations on your DHCP server. In this way, each Pi will keep the same IP address, but they will all be managed in one place, making reconfiguring your network in the future more straightforward.

On Raspberry Pi systems where the graphical desktop is installed, a GUI tool called `lxplug-network` is used to allow the user to make changes to the configuration of `dhcpcd`, including setting static IP addresses. The `lxplug-network` tool is based on `dhcpcd-ui`, which was also developed by Roy Marples.

Setting up a Headless Raspberry Pi

If you do not use a monitor or keyboard to run your Pi (known as headless), but you still need to do some wireless setup, there is a facility to enable wireless networking and SSH when creating a image.

Once an image is created on an SD card, by inserting it into a card reader on a Linux or Windows machines the boot folder can be accessed. Adding certain files to this folder will activate certain setup features on the first boot of the Raspberry Pi.

Configuring Networking

You will need to define a `wpa_supplicant.conf` file for your particular wireless network. Put this file onto the boot folder of the SD card. When the Raspberry Pi boots for the first time, it will copy that file into the correct location in the Linux root file system and use those settings to start up wireless networking.

The Raspberry Pi's IP address will not be visible immediately after power on, so this step is crucial to connect to it headlessly. Depending on the OS and editor you are creating this on, the file could have incorrect newlines or the wrong file extension so make sure you use an editor that accounts for this.

Linux expects the line feed (LF) newline character.

WARNING

After your Raspberry Pi is connected to power, make sure to wait a few (up to 5) minutes for it to boot up and registe network.

A `wpa_supplicant.conf` file example:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
country=<Insert 2 letter ISO 3166-1 country code here> update_config=1
 network={
 ssid="<Name of your wireless LAN>"
 psk="<Password for your wireless LAN>" }
```

Where the country code should be set the two letter ISO/IEC alpha2 code for the country in which you are using, e.g.

- GB (United Kingdom)
- FR (France)
- DE (Germany)
- US (United States)
- SE (Sweden)

Here is a more elaborate example that should work for most typical wpa2 personal networks. This

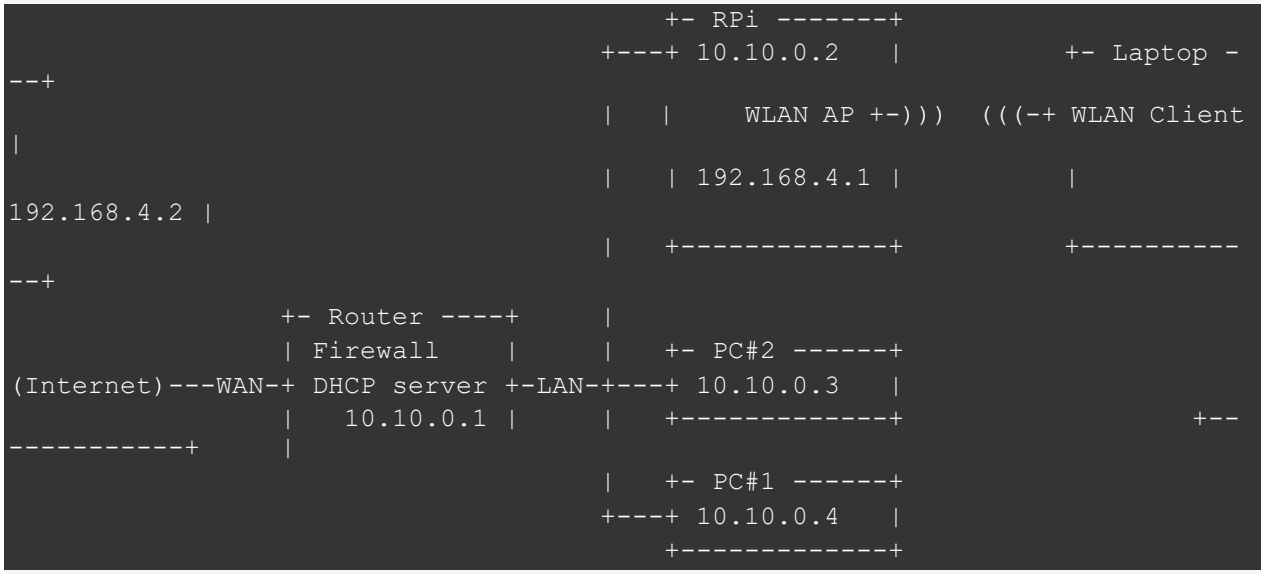
template below works for 2.4ghz/5ghz hidden or not networks. The utilization of quotes around the ssid - psk can help avoid any oddities if your network ssid or password has special chars (! @ # $ etc)

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev update_config=1
country=<Insert 2 letter ISO 3166-1 country code here>
 network={
scan_ssid=1
        ssid="<Name of your wireless LAN>"
psk="<Password for your wireless LAN>"
proto=RSN          key_mgmt=WPA-PSK
pairwise=CCMP          auth_alg=OPEN
}
```

Setting up a Routed Wireless Access Point

A Raspberry Pi within an Ethernet network can be used as a wireless access point, creating a secondary network. The resulting new wireless network is entirely managed by the Raspberry Pi.

If you wish to extend an existing Ethernet network to wireless clients, consider instead setting up a bridged access point.



A routed wireless access point can be created using the inbuilt wireless features of the Raspberry Pi 4, Raspberry Pi 3 or Raspberry Pi Zero W, or by using a suitable USB wireless dongle that supports access point mode. It is possible that some USB dongles may need slight changes to their settings. If you are having trouble with a USB wireless dongle, please check the forums.

This documentation was tested on a Raspberry Pi 3B running a fresh installation of Raspberry Pi OS Buster.

Before you Begin

- Ensure you have administrative access to your Raspberry Pi. The network setup will be modified as part of the installation: local access, with screen and keyboard connected to your Raspberry Pi,

is recommended.

- Connect your Raspberry Pi to the Ethernet network and boot the Raspberry Pi OS.
- Ensure the Raspberry Pi OS on your Raspberry Pi is up-to-date and reboot if packages were installed in the process.
- Take note of the IP configuration of the Ethernet network the Raspberry Pi is connected to:
  - In this document, we assume IP network `10.10.0.0/24` is configured on the Ethernet LAN, and the Raspberry Pi is going to manage IP network `192.168.4.0/24` for wireless clients.
  - Please select another IP network for wireless, e.g. `192.168.10.0/24`, if IP network `192.168.4.0/24` is already in use by your Ethernet LAN.
- Have a wireless client (laptop, smartphone, …) ready to test your new access point.

Install AP and Management Software

In order to work as an access point, the Raspberry Pi needs to have the `hostapd` access point software package installed:

```
sudo apt install hostapd
```

Enable the wireless access point service and set it to start when your Raspberry Pi boots:

```
sudo systemctl unmask hostapd sudo
systemctl enable hostapd
```

In order to provide network management services (DNS, DHCP) to wireless clients, the Raspberry Pi needs to have the `dnsmasq` software package installed:

```
sudo apt install dnsmasq
```

Finally, install `netfilter-persistent` and its plugin `iptables-persistent`. This utilty helps by saving firewall rules and restoring them when the Raspberry Pi boots:

```
sudo DEBIAN_FRONTEND=noninteractive apt install -y netfilter-persistent
iptables-persistent
```

Software installation is complete. We will configure the software packages later on.

Set up the Network Router

The Raspberry Pi will run and manage a standalone wireless network. It will also route between the wireless and Ethernet networks, providing internet access to wireless clients. If you prefer, you can choose to skip the routing by skipping the section "Enable routing and IP masquerading" below, and run the wireless network in complete isolation.

Define the Wireless Interface IP Configuration

The Raspberry Pi runs a DHCP server for the wireless network; this requires static IP configuration for the wireless interface (`wlan0`) in the Raspberry Pi. The Raspberry Pi also acts as the router on the wireless network, and as is customary, we will give it the first IP address in the network: `192.168.4.1`.

To configure the static IP address, edit the configuration file for `dhcpcd` with:

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

Go to the end of the file and add the following:

```
interface wlan0
    static ip_address=192.168.4.1/24
nohook wpa_supplicant
```

Enable Routing and IP Masquerading

This section configures the Raspberry Pi to let wireless clients access computers on the main (Ethernet) network, and from there the internet.

NOTE

If you wish to block wireless clients from accessing the Ethernet network and the internet, skip this section.

To enable routing, i.e. to allow traffic to flow from one network to the other in the Raspberry Pi, create a file using the following command, with the contents below:

```
sudo nano /etc/sysctl.d/routed-ap.conf
```

File contents:

```
# Enable IPv4 routing net.ipv4.ip_forward=1
```

Enabling routing will allow hosts from network `192.168.4.0/24` to reach the LAN and the main router towards the internet. In order to allow traffic between clients on this foreign wireless network and the internet without changing the configuration of the main router, the Raspberry Pi can substitute the IP address of wireless clients with its own IP address on the LAN using a "masquerade" firewall rule.

- The main router will see all outgoing traffic from wireless clients as coming from the Raspberry Pi, allowing communication with the internet.
- The Raspberry Pi will receive all incoming traffic, substitute the IP addresses back, and forward traffic to the original wireless client.

This process is configured by adding a single firewall rule in the Raspberry Pi:

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

Now save the current firewall rules for IPv4 (including the rule above) and IPv6 to be loaded at boot by the `netfilter-persistent` service:

```
sudo netfilter-persistent save
```

Filtering rules are saved to the directory `/etc/iptables/`. If in the future you change the configuration of your firewall, make sure to save the configuration before rebooting.

Configure the DHCP and DNS services for the wireless network

The DHCP and DNS services are provided by `dnsmasq`. The default configuration file serves as a template for all possible configuration options, whereas we only need a few. It is easier to start from an empty file.

Rename the default configuration file and edit a new one:

```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig sudo
nano /etc/dnsmasq.conf
```

Add the following to the file and save it:

```
interface=wlan0 # Listening interface
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
# Pool of IP addresses served via DHCP domain=wlan
# Local wireless DNS domain
address=/gw.wlan/192.168.4.1
                # Alias for this router
```

The Raspberry Pi will deliver IP addresses between `192.168.4.2` and `192.168.4.20`, with a lease time of 24 hours, to wireless DHCP clients. You should be able to reach the Raspberry Pi under the name `gw.wlan` from wireless clients.

There are many more options for `dnsmasq`; see the default configuration file (`/etc/dnsmasq.conf`) or the online documentation for details.

Ensure Wireless Operation

Countries around the world regulate the use of telecommunication radio frequency bands to ensure interference-free operation. The Linux OS helps users comply with these rules by allowing applications to be configured with a two-letter "WiFi country code", e.g. `US` for a computer used in the United States.

In the Raspberry Pi OS, 5 GHz wireless networking is disabled until a WiFi country code has been configured by the user, usually as part of the initial installation process (see wireless configuration pages in this section for details.)

To ensure WiFi radio is not blocked on your Raspberry Pi, execute the following command:

```
sudo rfkill unblock wlan
```

This setting will be automatically restored at boot time. We will define an appropriate country code in the access point software configuration, next.

Configure the AP Software

Create the `hostapd` configuration file, located at `/etc/hostapd/hostapd.conf`, to add the various parameters for your new wireless network.

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

Add the information below to the configuration file. This configuration assumes we are using channel 7, with a network name of `NameOfNetwork`, and a password `AardvarkBadgerHedgehog`. Note that the name and password should **not** have quotes around them. The passphrase should be between 8 and 64 characters in length.

```
country_code=GB
interface=wlan0
```

```
ssid=NameOfNetwork
hw_mode=g channel=7
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=AardvarkBadgerHedgehog
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP rsn_pairwise=CCMP
```

Note the line `country_code=GB`: it configures the computer to use the correct wireless frequencies in the United Kingdom. **Adapt this line** and specify the two-letter ISO code of your country. See Wikipedia for a list of two-letter ISO 3166-1 country codes.

To use the 5 GHz band, you can change the operations mode from `hw_mode=g` to `hw_mode=a`. Possible values for `hw_mode` are:

- a = IEEE 802.11a (5 GHz) (Raspberry Pi 3B+ onwards)
- b = IEEE 802.11b (2.4 GHz)
- g = IEEE 802.11g (2.4 GHz)

Note that when changing the `hw_mode`, you may need to also change the `channel` - see Wikipedia for a list of allowed combinations.

Running the new Wireless AP

Now restart your Raspberry Pi and verify that the wireless access point becomes automatically available.

```
sudo systemctl reboot
```

Once your Raspberry Pi has restarted, search for wireless networks with your wireless client. The network SSID you specified in file `/etc/hostapd/hostapd.conf` should now be present, and it should be accessible with the specified password.

If SSH is enabled on the Raspberry Pi, it should be possible to connect to it from your wireless client as follows, assuming the `pi` account is present: `ssh pi@192.168.4.1` or `ssh pi@gw.wlan`

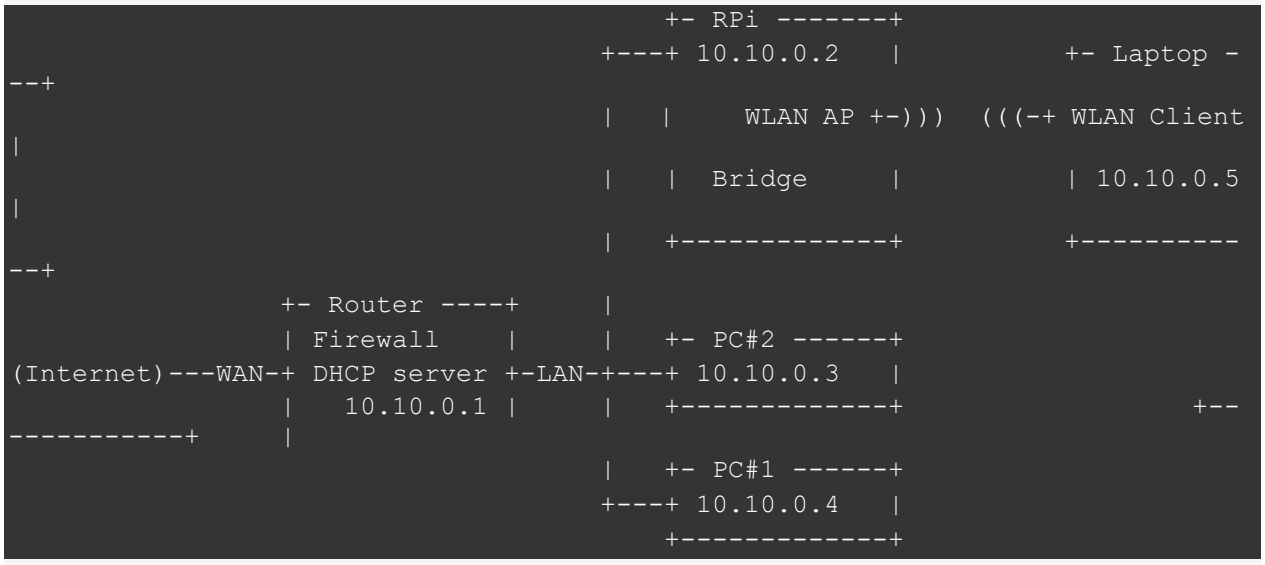
If your wireless client has access to your Raspberry Pi (and the internet, if you set up routing), congratulations on setting up your new access point!

If you encounter difficulties, contact the forums for assistance. Please refer to this page in your message.

Setting up a Bridged Wireless Access Point

The Raspberry Pi can be used as a bridged wireless access point within an existing Ethernet network.

This will extend the network to wireless computers and devices.
If you wish to create a standalone wireless network, consider instead setting up a routed access point.



A bridged wireless access point can be created using the inbuilt wireless features of the Raspberry Pi 4, Raspberry Pi 3 or Raspberry Pi Zero W, or by using a suitable USB wireless dongle that supports access point mode. It is possible that some USB dongles may need slight changes to their settings. If you are having trouble with a USB wireless dongle, please check the forums.

This documentation was tested on a Raspberry Pi 3B running a fresh installation of Raspberry Pi OS Buster.

Before you Begin

- Ensure you have administrative access to your Raspberry Pi. The network setup will be entirely reset as part of the installation: local access, with screen and keyboard connected to your Raspberry Pi, is recommended.

NOTE

If installing remotely via SSH, connect to your Raspberry Pi **by name** rather than by IP address, e.g. `ssh pi@raspberrypi.local`, as the address of your Raspberry Pi on the network will probably ch installation. You should also be ready to add screen and keyboard if needed in case you lose contact wit Raspberry Pi after installation.

- Connect your Raspberry Pi to the Ethernet network and boot the Raspberry Pi OS.
- Ensure the Raspberry Pi OS on your Raspberry Pi is up-to-date and reboot if packages were installed in the process.
- Have a wireless client (laptop, smartphone, …) ready to test your new access point.

Install AP and Management Software

In order to work as a bridged access point, the Raspberry Pi needs to have the `hostapd` access point software package installed:

```
sudo apt install hostapd
```

Enable the wireless access point service and set it to start when your Raspberry Pi boots:

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
```

Software installation is complete. We will configure the access point software later on.

Setup the Network Bridge

A bridge network device running on the Raspberry Pi will connect the Ethernet and wireless networks using its built-in interfaces.

Create a bridge device and populate the bridge

Add a bridge network device named `br0` by creating a file using the following command, with the contents below:

```
sudo nano /etc/systemd/network/bridge-br0.netdev
```

File contents:

```
[NetDev]
Name=br0
Kind=bridge
```

In order to bridge the Ethernet network with the wireless network, first add the built-in Ethernet interface (`eth0`) as a bridge member by creating the following file:

```
sudo nano /etc/systemd/network/br0-member-eth0.network
```

File contents:

```
[Match]
Name=eth0
[Network]
Bridge=br0
```

NOTE

The access point software will add the wireless interface `wlan0` to the bridge when the service starts. There is no need to for that interface. This situation is particular to wireless LAN interfaces.

Now enable the `systemd-networkd` service to create and populate the bridge when your Raspberry Pi boots:

```
sudo systemctl enable systemd-networkd
```

Define the bridge device IP configuration

Network interfaces that are members of a bridge device are never assigned an IP address, since they communicate via the bridge. The bridge device itself needs an IP address, so that you can reach your Raspberry Pi on the network.

`dhcpcd`, the DHCP client on the Raspberry Pi, automatically requests an IP address for every active interface. So we need to block the `eth0` and `wlan0` interfaces from being processed, and let `dhcpcd` configure only `br0` via DHCP.

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

Add the following line near the beginning of the file (above the first `interface xxx` line, if any):

```
denyinterfaces wlan0 eth0
```

Go to the end of the file and add the following:

```
interface br0
```

With this line, interface `br0` will be configured in accordance with the defaults via DHCP. Save the file to complete the IP configuration of the machine.

Ensure Wireless Operation

Countries around the world regulate the use of telecommunication radio frequency bands to ensure interference-free operation. The Linux OS helps users comply with these rules by allowing applications to be configured with a two-letter "WiFi country code", e.g. `US` for a computer used in the United States.

In the Raspberry Pi OS, 5 GHz wireless networking is disabled until a WiFi country code has been configured by the user, usually as part of the initial installation process (see wireless configuration pages in this section for details.)

To ensure WiFi radio is not blocked on your Raspberry Pi, execute the following command:

```
sudo rfkill unblock wlan
```

This setting will be automatically restored at boot time. We will define an appropriate country code in the access point software configuration, next.

Configure the AP Software

Create the `hostapd` configuration file, located at `/etc/hostapd/hostapd.conf`, to add the various parameters for your new wireless network.

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

Add the information below to the configuration file. This configuration assumes we are using channel 7, with a network name of `NameOfNetwork`, and a password `AardvarkBadgerHedgehog`. Note that the name and password should **not** have quotes around them. The passphrase should be between 8 and 64 characters in length.

```
country_code=GB
interface=wlan0
bridge=br0
ssid=NameOfNetwork
hw_mode=g
channel=7
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=AardvarkBadgerHedgehog
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP rsn_pairwise=CCMP
```

Note the lines `interface=wlan0` and `bridge=br0`: these direct `hostapd` to add the `wlan0` interface as a bridge member to `br0` when the access point starts, completing the bridge between Ethernet and wireless.

Note the line `country_code=GB`: it configures the computer to use the correct wireless frequencies in the United Kingdom. **Adapt this line** and specify the two-letter ISO code of your country. See Wikipedia for a list of two-letter ISO 3166-1 country codes.

To use the 5 GHz band, you can change the operations mode from `hw_mode=g` to `hw_mode=a`. Possible values for `hw_mode` are:

- a = IEEE 802.11a (5 GHz) (Raspberry Pi 3B+ onwards)
- b = IEEE 802.11b (2.4 GHz)
- g = IEEE 802.11g (2.4 GHz)

Note that when changing the `hw_mode`, you may need to also change the `channel` - see Wikipedia for a list of allowed combinations.

Run the new Wireless AP

Now restart your Raspberry Pi and verify that the wireless access point becomes automatically available.

```
sudo systemctl reboot
```

Once your Raspberry Pi has restarted, search for wireless networks with your wireless client. The network SSID you specified in file `/etc/hostapd/hostapd.conf` should now be present, and it should be accessible with the specified password.

If your wireless client has access to the local network and the internet, congratulations on setting up your new access point!

If you encounter difficulties, contact the forums for assistance. Please refer to this page in your message.

Using a Proxy Server

If you want your Raspberry Pi to access the Internet via a proxy server (perhaps from a school or other workplace), you will need to configure your Pi to use the server before you can get online.

You will need:

- The IP address or hostname and port of your proxy server
- A username and password for your proxy (if required)

Configuring your Raspberry Pi

You will need to set up three environment variables (`http_proxy`, `https_proxy`, and `no_proxy`) so your Raspberry Pi knows how to access the proxy server.

Open a terminal window, and open the file `/etc/environment` using nano:

```
sudo nano /etc/environment
```

Add the following to the `/etc/environment` file to create the `http_proxy` variable:

```
export http_proxy="http://proxyipaddress:proxyport"
```

Replace `proxyipaddress` and `proxyport` with the IP address and port of your proxy.

NOTE

if your proxy requires a username and password, add them using the following format:

```
export http_proxy="http://username:password@proxyipaddress:proxyport"
```

Enter the same information for the environment variable `https_proxy`:

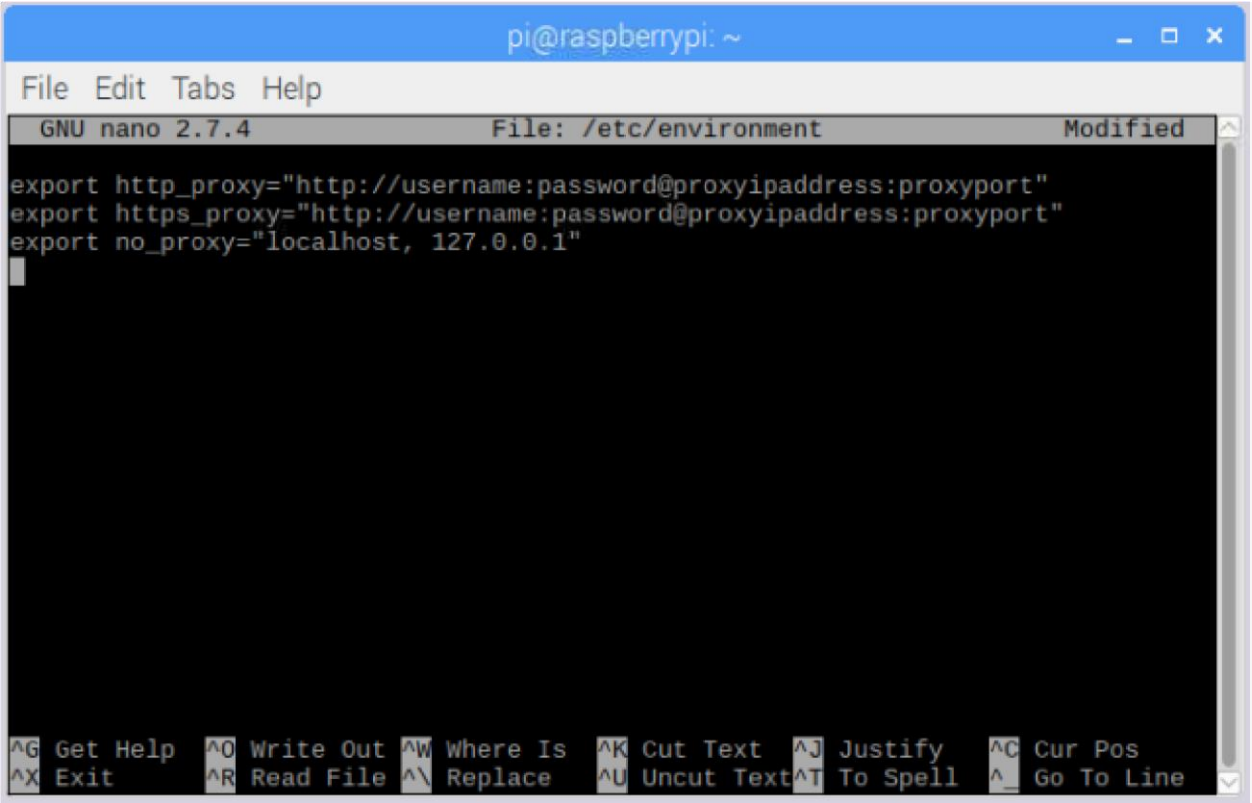
```
export https_proxy="http://username:password@proxyipaddress:proxyport"
```

Create the `no_proxy` environment variable, which is a comma-separated list of addresses your Pi should not use the proxy for:

```
export no_proxy="localhost, 127.0.0.1"
```

Your `/etc/environment` file should now look like this:

```
export http_proxy="http://username:password@proxyipaddress:proxyport" export
https_proxy="http://username:password@proxyipaddress:proxyport" export
no_proxy="localhost, 127.0.0.1"
```



Press `Ctrl + X` to save and exit.

Update the `sudoers` File

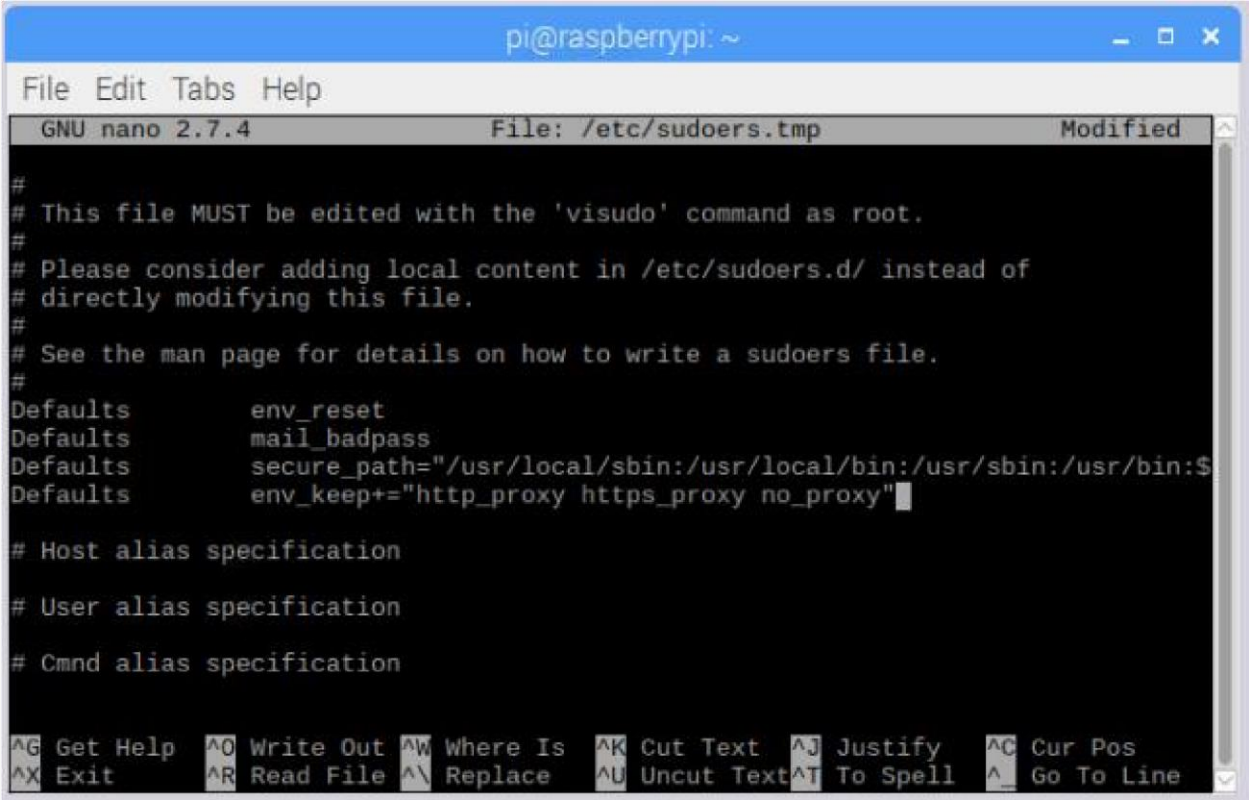
In order for operations that run as `sudo` (e.g. downloading and installing software) to use the new environment variables, you'll need to update `sudoers`.

Use the following command to open `sudoers`:

```
sudo visudo
```

Add the following line to the file so `sudo` will use the environment variables you just created:

```
Defaults    env_keep+="http_proxy https_proxy no_proxy"
```



Press `Ctrl + X` to save and exit.

Reboot your Raspberry Pi

Reboot your Raspberry Pi for the changes to take effect. You should now be able to access the internet via your proxy server.

HDMI Configuration

In the vast majority of cases, simply plugging your HDMI-equipped monitor into the Raspberry Pi using a standard HDMI cable will automatically lead to the Pi using the best resolution the monitor supports. The Raspberry Pi Zero uses a mini HDMI port, so you will need a mini-HDMI-to-full-size-HDMI lead or adapter. On the Raspberry Pi 4 there are two micro HDMI ports, so you will need either one or two micro-HDMI-to-full-size-HDMI leads or adapters, depending on how many displays you wish to attach. You should connect any HDMI leads before turning on the Raspberry Pi.

The Raspberry Pi 4 can drive up to two displays, with a resolution up to 1080p at a 60Hz refresh rate. At 4K resolution, if you connect two displays then you are limited to a 30Hz refresh rate. You can also drive a single display at 4K with a 60Hz refresh rate: this requires that the display is attached to the HDMI port adjacent to the USB-C power input (labelled HDMI0). You must also enable 4Kp60 output by setting

the `hdmi_enable_4kp60=1` flag in config.txt. This flag can also be set using the 'Raspberry Pi Configuration' tool within the desktop environment.

If you are running the 3D graphics driver (also known as the FKMS driver), then in the Preferences menu you will find a graphical application for setting up standard displays, including multi-display setups.

If you are using legacy graphics drivers, or find yourself in circumstances where the Raspberry Pi may not be able to determine the best mode, or you may specifically wish to set a non-default resolution, the rest of this page may be useful.

HDMI Groups and Mode

HDMI has two common groups: CEA (Consumer Electronics Association, the standard typically used by TVs) and DMT (Display Monitor Timings, the standard typically used by monitors). Each group advertises a particular set of modes, where a mode describes the resolution, frame rate, clock rate, and aspect ratio of the output.

What Modes does my Device Support?

You can use the `tvservice` application on the command line to determine which modes are supported by your device, along with other useful data:

- `tvservice -s` displays the current HDMI status, including mode and resolution
- `tvservice -m CEA` lists all supported CEA modes
- `tvservice -m DMT` lists all supported DMT modes

If you are using a Pi 4 with more than one display attached, then `tvservice` needs to be told which device to ask for information. You can get display IDs for all attached devices by using:

```
tvservice -l
```

You can specify which display `tvservice` uses by adding `-v <display id>` to the `tvservice` command, e.g:

- `tvservice -v 7 -m CEA`, lists all supported CEA modes for display ID 7

Setting a Specific HDMI Mode

Setting a specific mode is done using the `hdmi_group` and `hdmi_mode` config.txt entries. The group entry selects between CEA or DMT, and the mode selects the resolution and frame rate. You can find tables of

modes on the config.txt Video Configuration page, but you should use the `tvservice` command described above to find out exactly which modes your device supports.

On the Pi 4, to specify the HDMI port, add an index identifier to the `hdmi_group` or `hdmi_mode` entry in config.txt, e.g. `hdmi_mode:0` or `hdmi_group:1`.

Setting a Custom HDMI Mode

There are two options for setting a custom mode: `hdmi_cvt` and `hdmi_timings`.

`hdmi_cvt` sets a custom Coordinated Video Timing entry, which is described fully here: Video Configuration

In certain rare cases it may be necessary to define the exact clock requirements of the HDMI signal. This is a fully custom mode, and it is activated by setting `hdmi_group=2` and `hdmi_mode=87`. You can then use the `hdmi_timings` config.txt command to set the specific parameters for your display. `hdmi_timings` specifies all the timings that an HDMI signal needs to use. These timings are usually found in the datasheet of the display being used.

Timing	Purpose
`h_active_pixels`	The horizontal resolution
`h_sync_polarity`	0 or 1 to define the horizontal sync polarity
`h_front_porch`	Number of horizontal front porch pixels
`h_sync_pulse`	Width of horizontal sync pulse
`h_back_porch`	Number of horizontal back porch pixels
`v_active_lines`	The vertical resolution `v_sync_polarity`
	0 or 1 to define the vertical sync polarity

Timing	Purpose
`v_front_porch`	Number of vertical front porch pixels
`v_sync_pulse`	Width of vertical sync pulse
`v_back_porch`	Number of vertical back porch pixels
`v_sync_offset_a`	Leave at 0
`v_sync_offset_b`	Leave at 0
`pixel_rep`	Leave at 0
`frame_rate`	Frame rate of mode
`interlaced`	0 for non-interlaced, 1 for interlaced
`pixel_freq`	The mode pixel frequency
`aspect_ratio`	The aspect ratio required

`aspect_ratio` should be one of the following:

Ratio	`aspect_ratio` ID
`4:3`	1
`14:9`	2
`16:9`	3
`5:4`	4
`16:10`	5

`15:9`	6
`21:9`	7
`64:27`	8

For the Pi4, to specify the HDMI port, you can add an index identifier to the config.txt.

e.g. `hdmi_cvt:0=...` or `hdmi_timings:1=...`. If no port identifier is specified, the settings are applied to port 0.

Troubleshooting your HDMI

In some rare cases you may need to increase the HDMI drive strength, for example when there is speckling on the display or when you are using very long cables. There is a config.txt item to do this, `config_hdmi_boost`, which is documented on the config.txt video page.

Rotating your Display

The options to rotate the display of your Raspberry Pi depend on which display driver software it is running, which may also depend on which Raspberry Pi you are using.

Fake or Full KMS Graphics Driver

If you are running the Raspberry Pi desktop then rotation is achieved by using the `Screen Configuration Utility` from the desktop `Preferences` menu. This will bring up a graphical representation of the display or displays connected to the Raspberry Pi. Right click on the display you wish to rotate and select the required option.

It is also possible to change these settings using the command line `xrandr` option. The following commands give 0°, -90°, +90° and 180° rotations respectively.

```
xrandr --output HDMI-1 --rotate normal xrandr
--output HDMI-1 --rotate left xrandr --output
HDMI-1 --rotate right xrandr --output HDMI-1
--rotate inverted
```

Note that the `--output` entry specifies to which device the rotation applies. You can determine the device name by simply typing `xrandr` on the command line which will display information, including the name, for all attached devices.

You can also use the command line to mirror the display using the `--reflect` option. Reflection can be one of 'normal' 'x', 'y' or 'xy'. This causes the output contents to be reflected across the specified axes. For example:

```
xrandr -output HDMI-1 --reflect x
```

If you are using the console only (no graphical desktop) then you will need to set the appropriate kernel command line flags. Change the console settings as described on the this page.

Legacy Graphics Driver

There are `config.txt` options for rotating when using the legacy display drivers.

`display_hdmi_rotate` is used to rotate the HDMI display, `display_lcd_rotate` is used to rotate any attached LCD panel (using the DSI or DPI interface). These options rotate both the desktop and console. Each option takes one of the following parameters :

display_*_rotate	result
0	no rotation
1	rotate 90 degrees clockwise
2	rotate 180 degrees clockwise
3	rotate 270 degrees clockwise
0x10000	horizontal flip
0x20000	vertical flip

Note that the 90 and 270 degree rotation options require additional memory on the GPU, so these will not

work with the 16MB GPU split.

You can combine the rotation settings with the flips by adding them together. You can also have both horizontal and vertical flips in the same way. E.g. A 180 degree rotation with a vertical and horizontal flip will be 0x20000 + 0x10000 + 2 = 0x30002.

Audio Configuration

The Raspberry Pi has up to three audio output modes: HDMI 1 and 2, if present, and a headphone jack. You can switch between these modes at any time.

If your HDMI monitor or TV has built-in speakers, the audio can be played over the HDMI cable, but you can switch it to a set of headphones or other speakers plugged into the headphone jack. If your display claims to have speakers, sound is output via HDMI by default; if not, it is output via the headphone jack. This may not be the desired output setup, or the auto-detection is inaccurate, in which case you can manually switch the output.

Changing the Audio Output

There are two ways of setting the audio output; using the Desktop volume control, or using `raspiconfig` command line tool.

Using the Desktop

Right-clicking the volume icon on the desktop taskbar brings up the audio output selector; this allows you to select between the internal audio outputs. It also allows you to select any external audio devices, such as USB sound cards and Bluetooth audio devices. A green tick is shown against the currently selected audio output device — simply left-click the desired output in the pop-up menu to change this. The volume control and mute operate on the currently selected device.

Using raspi-config

Open up raspi-config by entering the following into the command line:

```
sudo raspi-config
```

This will open the configuration screen:

Select `System Options` (Currently option 1, but yours may be different) and press `Enter`.

Now select the Option named, `Audio` (Currently option S2, but yours may be different) and press `Enter`:

Select your required mode, press `Enter` and press the right arrow key to exit the options list, then select `Finish` to exit the configuration tool.

After you have finished modifying your audio settings, you need to restart your Raspberry Pi in order for your changes to take effect.

Troubleshooting your HDMI

In some rare cases, it is necessary to edit `config.txt` to force HDMI mode (as opposed to DVI mode, which does not send sound). You can do this by editing `/boot/config.txt` and setting `hdmi_drive=2`, then rebooting for the change to take effect.

External Storage Configuration

You can connect your external hard disk, SSD, or USB stick to any of the USB ports on the Raspberry Pi, and mount the file system to access the data stored on it.

By default, your Raspberry Pi automatically mounts some of the popular file systems such as FAT, NTFS, and HFS+ at the `/media/pi/<HARD-DRIVE-LABEL>` location.

To set up your storage device so that it always mounts to a specific location of your choice, you must mount it manually.

Mounting a Storage Device

You can mount your storage device at a specific folder location. It is conventional to do this within the `/mnt` folder, for example `/mnt/mydisk`. Note that the folder must be empty.

1. Plug the storage device into a USB port on the Raspberry Pi.

2. List all the disk partitions on the Pi using the following command:

```
sudo lsblk -o UUID,NAME,FSTYPE,SIZE,MOUNTPOINT,LABEL,MODEL
```

The Raspberry Pi uses mount points `/` and `/boot`. Your storage device will show up in this list, along with any other connected storage.

3. Use the SIZE, LABEL, and MODEL columns to identify the name of the disk partition that points to your storage device. For example, `sda1`.

4. The FSTYPE column contains the filesystem type. If your storage device uses an exFAT file system, install the exFAT driver:

5.
```
sudo apt update
sudo apt install exfat-fuse
```

6. If your storage device uses an NTFS file system, you will have read-only access to it. If you want to write to the device, you can install the ntfs-3g driver :

7.
```
sudo apt update
sudo apt install ntfs-3g
```

8. Run the following command to get the location of the disk partition:

```
sudo blkid
```

For example, `/dev/sda1`.

9. Create a target folder to be the mount point of the storage device. The mount point name used in this case is `mydisk`. You can specify a name of your choice:

```
sudo mkdir /mnt/mydisk
```

10. Mount the storage device at the mount point you created:

```
sudo mount /dev/sda1 /mnt/mydisk
```

11. Verify that the storage device is mounted successfully by listing the contents:

```
ls /mnt/mydisk
```

Setting up Automatic Mounting

You can modify the `fstab` file to define the location where the storage device will be automatically mounted when the Raspberry Pi starts up. In the `fstab` file, the disk partition is identified by the universally unique identifier (UUID).

1. Get the UUID of the disk partition:

```
sudo blkid
```

2. Find the disk partition from the list and note the UUID. For example, `5C24-1453`.

3. Open the fstab file using a command line editor such as nano:

```
sudo nano /etc/fstab
```

4. Add the following line in the `fstab` file:

```
UUID=5C24-1453 /mnt/mydisk fstype defaults,auto,users,rw,nofail 0 0
```

Replace `fstype` with the type of your file system, which you found in step 2 of 'Mounting a storage device' above, for example: `ntfs`.

5. If the filesystem type is FAT or NTFS, add `,umask=000` immediately after `nofail` - this will allow all users full read/write access to every file on the storage device.

Now that you have set an entry in `fstab`, you can start up your Raspberry Pi with or without the storage device attached. Before you unplug the device you must either shut down the Pi, or manually unmount it using the steps in 'Unmounting a storage device' below.

NOTE if you do not have the storage device attached when the Pi starts, the Pi will take an extra 90 seconds to start up. You can s by adding `,x-systemd.device-timeout=30` immediately after `nofail` in step 4. This will change the timeout to 3 meaning the system will only wait 30 seconds before giving up trying to mount the disk.

For more information on each Linux command, refer to the specific manual page using the `man` command. For example, `man fstab`.

Unmounting a Storage Device

When the Raspberry Pi shuts down, the system takes care of unmounting the storage device so that it is safe to unplug it. If you want to manually unmount a device, you can use the following command:

```
sudo umount /mnt/mydisk
```

If you receive an error that the 'target is busy', this means that the storage device was not unmounted. If no error was displayed, you can now safely unplug the device.

Dealing with 'target is busy'

The 'target is busy' message means there are files on the storage device that are in use by a program. To close the files, use the following procedure.

1. Close any program which has open files on the st orage device.
2. If you have a terminal open, make sure that you are not in the folder where the storage device is mounted, or in a sub -folder of it.
3. If you are still unable to unmount the storage device, you can use the `lsof` tool to check which program has f iles open on the device. You need to first install `lsof` using `apt`:

```
sudo apt update
sudo apt install lsof
```

To use lsof:

```
lsof /mnt/mydisk
```

Localising your Raspberry Pi

You can set your Raspberry Pi up to match your regional settings.

Changing the Language

If you want to select a different language use raspi-config.

Configuring the Keyboard

If you want to select a different keyboard use raspi-config.

Changing the Timezone

Once again, this is something you can change using the raspi-config tool.

Changing the default pin configuration

As of July 2014, the Raspberry Pi firmware supports custom default pin configurations through a userprovided Device Tree blob file.

To find out whether your firmware is recent enough, please run `vcgencmd version`.

Device Pins During Boot Sequence

During the bootup sequence, the GPIO pins go through various actions.

1. Power-on — pins default to inputs with default pulls; the default pulls for each pin are described in the datasheet
2. Setting by the bootrom
3. Setting by `bootcode.bin`
4. Setting by `dt-blob.bin` (this page)
5. Setting by the GPIO command in `config.txt`
6. Additional firmware pins (e.g. UARTS)
7. Kernel/Device Tree

On a soft reset, the same procedure applies, except for default pulls, which are only applied on a power-on reset.

Note that it may take a few seconds to get from stage 1 to stage 4. During that time, the GPIO pins may not be in the state expected by attached peripherals (as defined in `dtblob.bin` or `config.txt`). Since different GPIO pins have different default pulls, you should do **one of the following** for your peripheral:

- Choose a GPIO pins that defaults to pulls as required by the peripheral on reset
- Delay the peripheral's startup until stage 4/5 has been reached
- Add an appropriate pull-up/-down resistor

Providing a Custom Device Tree Blob

In order to compile a Device Tree source (`.dts`) file into a Device Tree blob (`.dtb`) file, the Device Tree compiler must be installed by running `sudo apt install device-tree-compiler`.

The `dtc` command can then be used as follows:

```
sudo dtc -I dts -O dtb -o /boot/dt-blob.bin dt-blob.dts
```

Similarly, a `.dtb` file can be converted back to a `.dts` file, if required.

```
dtc -I dtb -O dts -o dt-blob.dts /boot/dt-blob.bin
```

Sections of the `dt-blob`

The `dt-blob.bin` is used to configure the binary blob (VideoCore) at boot time. It is not currently used by the Linux kernel, but a kernel section will be added at a later stage, when we reconfigure the Raspberry Pi kernel to use a dt-blob for configuration. The dt-blob can configure all versions of the Raspberry Pi, including the Compute Module, to use the alternative settings. The following sections are valid in the dtblob:

1. `videocore`

   This section contains all of the VideoCore blob information. All subsequent sections must be enclosed within this section.

2. `pins_*`

   `There are a number of separate `pins_*` sections, based on particular Raspberry Pi models, namely:`

   - **pins_rev1** Rev1 pin setup. There are some differences because of the moved I2C pins.
   - **pins_rev2** Rev2 pin setup. This includes the additional codec pins on P5.
   - **pins_bplus1** Model B+ rev 1.1, including the full 40pin connector.
   - **pins_bplus2** Model B+ rev 1.2, swapping the low-power and lan-run pins.
   - **pins_aplus** Model A+, lacking Ethernet.
   - **pins_2b1** Pi 2 Model B rev 1.0; controls the SMPS via I2C0.
   - **pins_2b2** Pi 2 Model B rev 1.1; controls the SMPS via software I2C on 42 and 43.
   - **pins_3b1** Pi 3 Model B rev 1.0
   - **pins_3b2** Pi 3 Model B rev 1.2
   - **pins_3bplus** Pi 3 Model B+
   - **pins_3aplus** Pi 3 Model A+
   - **pins_pi0** The Pi Zero
   - **pins_pi0w** The Pi Zero W
   - **pins_cm** The Compute Module. The default for this is the default for the chip, so it is a useful source of information about default pull ups/downs on the chip.

- **pins_cm3** The Compute Module version 3

Each `pins_*` section can contain `pin_config` and `pin_defines` sections.

3. `pin_config`

The `pin_config` section is used to configure the individual pins. Each item in this section must be a named pin section, such as `pin@p32`, meaning GPIO32. There is a special section `pin@default`, which contains the default settings for anything not specifically named in the pin_config section.

4. `pin@pinname`

This section can contain any combination of the following items:

- `polarity`
  - `active_high`
  - `active_low`
- `termination`
  - `pull_up`
  - `pull_down`
  - `no_pulling`
- `startup_state`
  - `active`
  - `inactive`
- `function`
  - `input`
  - `output`
  - `sdcard`
  - `i2c0`

        - `i2c1`
        - `spi`
        - `spi1`
        - `spi2`
        - `smi`
        - `dpi`
        - `pcm`
        - `pwm`
        - `uart0`
        - `uart1`
        - `gp_clk`
        - `emmc`
        - `arm_jtag`

    - `drive_strength_mA` The drive strength is used to set a strength for the pins. Please note that you can only specify a single drive strength for the bank. <8> and <16> are valid values.

5. `pin_defines`
   This section is used to set specific VideoCore functionality to particular pins. This enables the user to move the camera power enable pin to somewhere different, or move the HDMI hotplug position: things that Linux does not control. Please refer to the example DTS file below.

Practical No: 03

Camera Module:

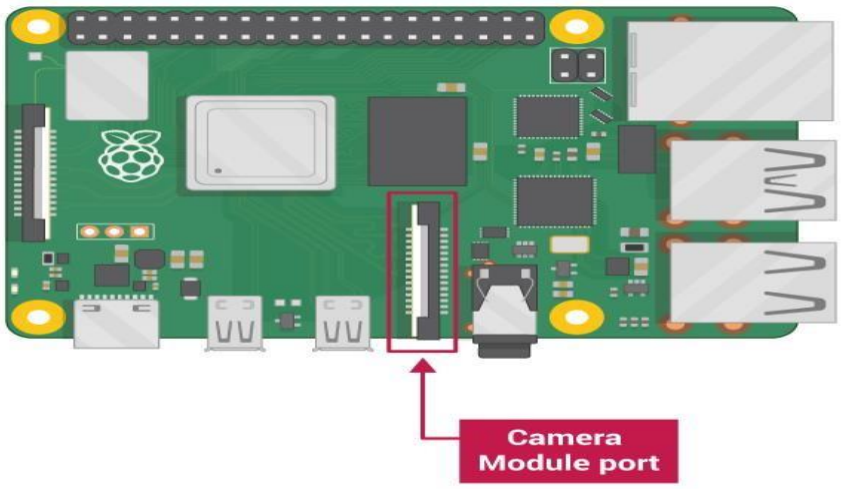
Introduction

Learn how to connect the Raspberry Pi Camera Module to your Raspberry Pi and take pictures, record video, and apply image effects.

What you will need

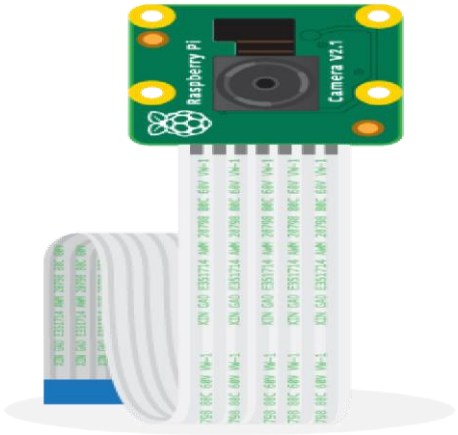
Raspberry Pi computer with a Camera Module port

All current models of Raspberry Pi have a port for connecting the Camera Module.



Note: If you want to use a Raspberry Pi Zero, you need a Camera Module ribbon cable that fits the Raspberry Pi Zero's smaller Camera Module port.

Raspberry Pi Camera Module



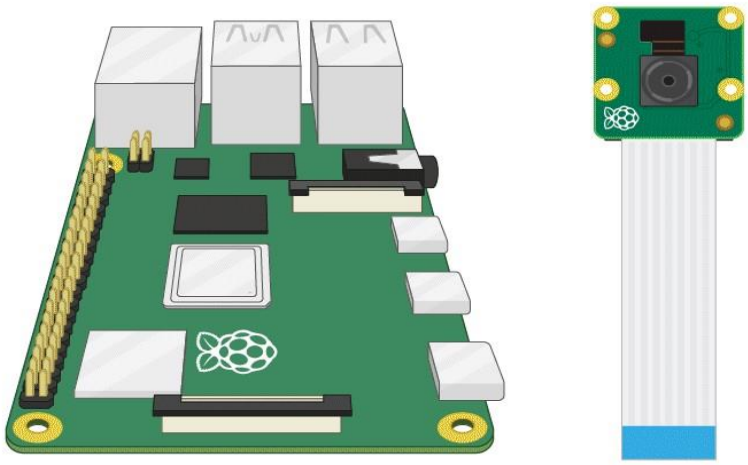
There are two versions of the Camera Module:

- The standard version, which is designed to take pictures in normal light
- The NoIR version, which doesn't have an infrared filter, so you can use it together with an infrared light source to take pictures in the dark

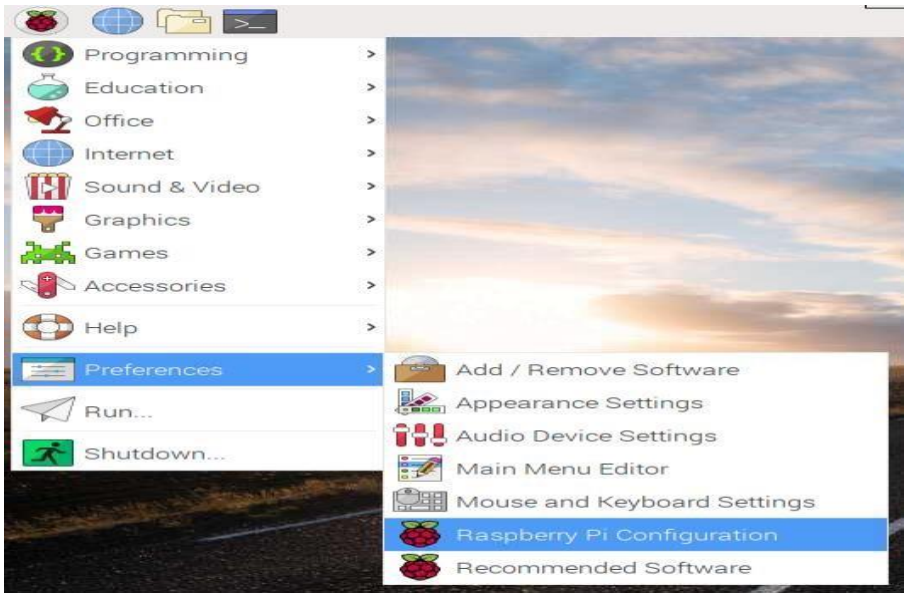
Connect the Camera Module

Ensure your Raspberry Pi is turned off.

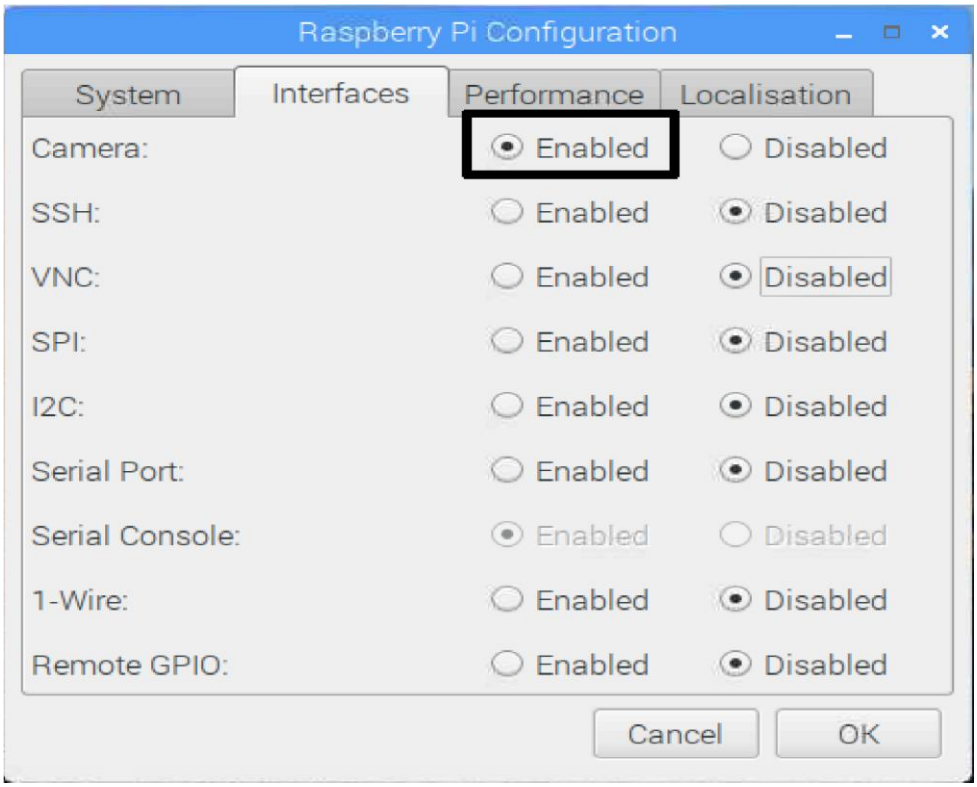
1. Locate the Camera Module port
2. Gently pull up on the edges of the port's plastic clip
3. Insert the Camera Module ribbon cable; make sure the connectors at the bottom of the ribbon cable are facing the contacts in the port.
4. Push the plastic clip back into place



- Start up your Raspberry Pi. ○ Go to the main menu and open the Raspberry Pi Configuration tool.



- Select the Interfaces tab and ensure that the camera is enabled:



- Reboot your Raspberry Pi.

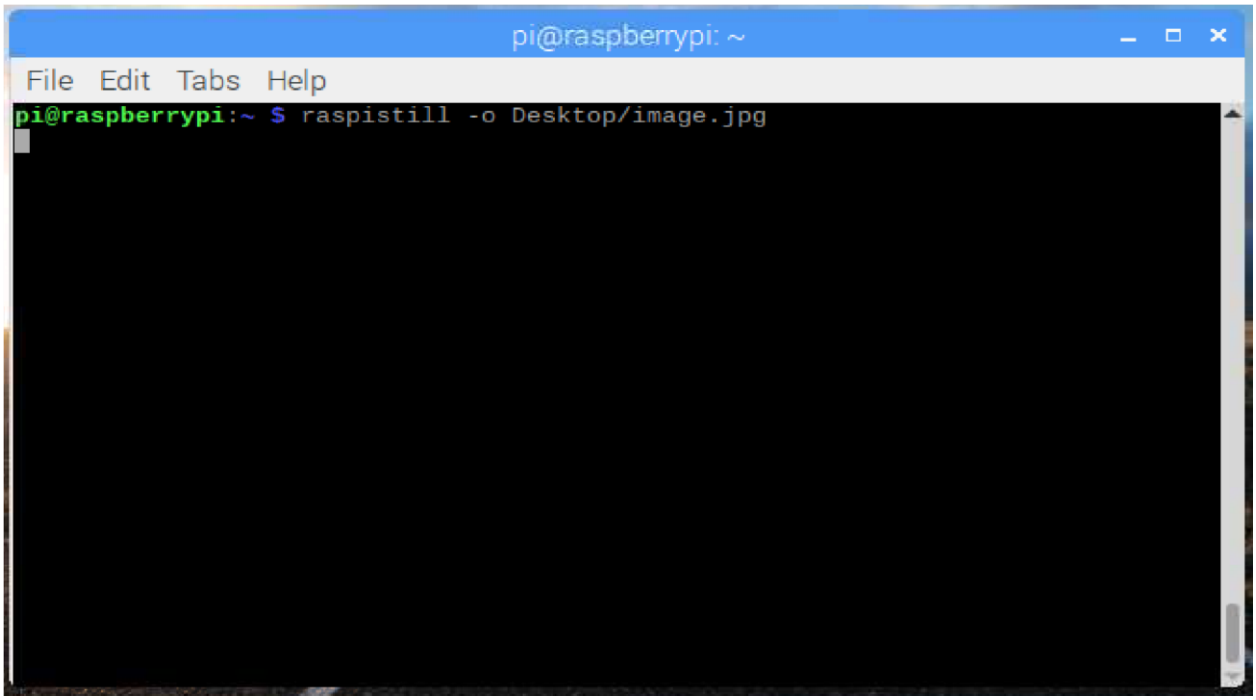
How to control the Camera Module via the command line

Now your Camera Module is connected and the software is enabled, try out the command line tools `raspistill` and `raspivid`

- Open a terminal window by clicking the black monitor icon in the taskbar:

- Type in the following command to take a still picture and save it to the Desktop:

```
raspistill -o Desktop/image.jpg
```



- Press `Enter` to run the command.

When the command runs, you can see the camera preview open for five seconds before a still picture is taken.

- Look for the picture file icon on the Desktop, and double-click the file icon to open the picture.

By adding different options, you can set the size and look of the image the `raspistill` command takes.

- For example, add `-h` and `-w` to change the height and width of the image:

```
raspistill -o Desktop/image-small.jpg -w 640 -h 480
```

- Now record a video with the Camera Module by using the following `raspivid` command:

```
raspivid -o Desktop/video.h264
```

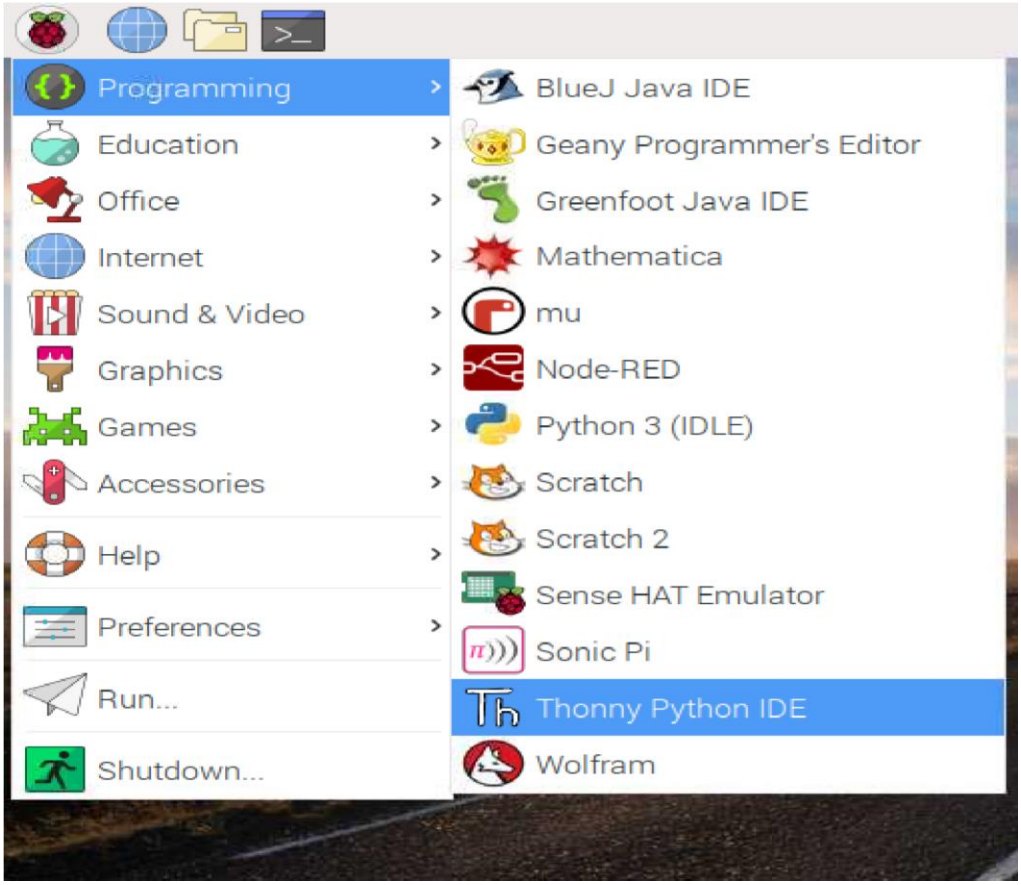
- In order to play the video file, double-click the file icon on the Desktop to open it in VLC Media Player.

For more information and other options you can use with these commands, read the documentation for raspistill and the documentation for raspivid.

How to control the Camera Module with Python code

The Python `picamera` library allows you to control your Camera Module and create amazing projects.

- Open a Python 3 editor, such as Thonny Python IDE:



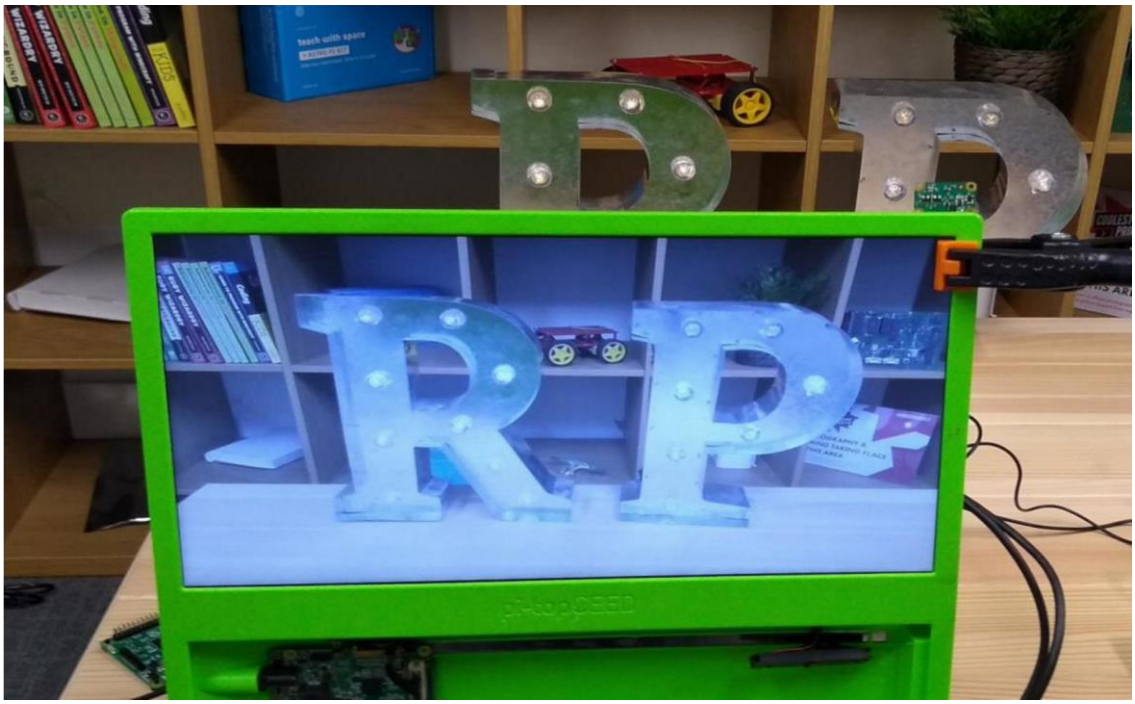
- Open a new file and save it as `camera.py`.
  Note: it's important that you never save the file as `picamera.py`.
- Enter the following code:

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()

camera.start_preview()
sleep(5) camera.stop_preview()
```

- Save and run your program. The camera preview should be shown for five seconds and then close again.



Note: the camera preview only works when a monitor is connected to your Raspberry Pi. If you are using remote access (such as SSH or VNC), you won't' see the camera preview.

- If your preview is upside-down, you can rotate it by 180 degrees with the following code:

```
camera = PiCamera() camera.rotation = 180
```

You can rotate the image by 90, 180, or 270 degrees. To reset the image, set to degrees.

It's best to make the preview slightly see-through so you can see whether errors occur in your program while the preview is on.

- Make the camera preview see-through by setting an level:

```
camera.start_preview(alpha=200)
```

The alpha value can be any number between 0 and 255.

Take still pictures with Python code

Now use the Camera Module and Python to take some still pictures.

- Amend your code to add a `camera.capture()` line:

```
camera.start_preview() sleep(5)
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
```

Note: it's important to `sleep` for at least two seconds before capturing an image, because this gives the camera's sensor time to sense the light levels.

- Run the code.

You should see the camera preview open for five seconds, and then a still picture should be captured. As the picture is being taken, you can see the preview briefly adjust to a different resolution.

Your new image should be saved to the Desktop.

- Now add a loop to take five pictures in a row:

```
camera.start_preview()
for i in range(5):
sleep(5)
camera.capture('/home/pi/Desktop/image%s.jpg' % i)
camera.stop_preview()
```

The variable counts how many times the loop has run, from to . Therefore, the images get saved as , , and so on.

Run the code again and hold the Camera Module in position.

The camera should take one picture every five seconds. Once the fifth picture is taken, the preview closes.

- Look at your Desktop to find the five new pictures.

Recording video with Python code

Now record a video!

- Amend your code to remove `capture()` and instead add `start_recording()` and `stop_recording()`
  Your code should look like this now:

```
camera.start_preview()
camera.start_recording('/home/pi/Desktop/video.h264')
sleep(5)
camera.stop_recording()
camera.stop_preview()
```

- Run the code.

Your Raspberry Pi should open a preview, record 5 seconds of video, and then close the preview.

How to change the image settings and add image effects

The Python `picamera` software provides a number of effects and configurations to change how your images look.
Note: some settings only affect the preview and not the captured image, some affect only the captured image, and many others affect both.

Set the image resolution

You can change the `resolution` of the image that the Camera Module takes.

By default, the image resolution is set to the resolution of your monitor. The maximum resolution is 2592×1944 for still photos, and 1920×1080 for video recording.

- Use the following code to set the `resolution` to maximum and take a picture.
  Note: you also need to set the frame rate to 15 to enable this maximum resolution.

```
camera.resolution = (2592, 1944)
camera.framerate = 15
camera.start_preview() sleep(5)
camera.capture('/home/pi/Desktop/max.jpg')
camera.stop_preview()
```

The minimum resolution is 64×64.

- Try taking a picture with the minimum resolution.

Add text to your image

You can add text to your image using the command `annotate_text`.

- Run this code to try it:

```
camera.start_preview()
camera.annotate_text = "Hello world!"
sleep(5)
camera.capture('/home/pi/Desktop/text.jpg')
camera.stop_preview()
```

Change the look of the added text

- Set the text size with the following code:

```
camera.annotate_text_size = 50
```

You can set the text size to anything between 6 to 160. The default size is 32. It's also possible to change the text colour.

- First of all, add `Color` to your `import` line at the top of the program:

```
from picamera import PiCamera, Color
```

Then below the `import` line, amend the rest of your code so it looks like this:

```
camera.start_preview()
camera.annotate_background = Color('blue')
camera.annotate_foreground = Color('yellow')
camera.annotate_text = " Hello world "
sleep(5)
camera.stop_preview()
```

Change the brightness of the preview

You can change how bright the preview appears. The default brightness is 50, and you can set it to any value between and .

- Run the following code to try this out:

```
camera.start_preview()
camera.brightness = 70
sleep(5)
camera.capture('/home/pi/Desktop/bright.jpg')
camera.stop_preview()
```

- The following loop adjusts the brightness and also adds text to display the current brightness level:

```
camera.start_preview()
for i in range(100):
    camera.annotate_text = "Brightness: %s" % i
    camera.brightness = i
    sleep(0.1)
camera.stop_preview()
```

Change the contrast of the preview

Similarly to the preview brightness, you can change the contrast of the preview.

- Run the following code to try this out:

```
camera.start_preview()
for i in range(100):
    camera.annotate_text = "Contrast: %s" % i
    camera.contrast = i
    sleep(0.1)
camera.stop_preview()
```

Add cool image effects

You can use `camera.image_effect` to apply a particular image effect.
The image effect options are:

- none
- negative
- solarize
- sketch
- denoise
- emboss
- oilpaint
- hatch
- gpen
- pastel
- watercolor
- film
- blur
- saturation
- colorswap
- washedout
- posterise
- colorpoint
- colorbalance
- cartoon
- deinterlace1
- deinterlace2

The default effect is `none`.

- Pick an image effect and try it out:

```
camera.start_preview()
camera.image_effect = 'colorswap'
sleep(5)
camera.capture('/home/pi/Desktop/colorswap.jpg')
camera.stop_preview()
```

- Run this code to loop over all the image effects with `camera.IMAGE_EFFECTS`:

```
camera.start_preview()
for effect in camera.IMAGE_EFFECTS:
```

```
    camera.image_effect = effect
    camera.annotate_text = "Effect: %s" % effect
    sleep(5)
camera.stop_preview()
```

Set the image exposure mode

You can use `camera.exposure_mode` to set the exposure to a particular mode.
The exposure mode options are:

- `off`
- `auto`
- `night`
- `nightpreview`
- `backlight`
- `spotlight`
- `sports`
- `snow`
- `beach`
- `verylong`
- `fixedfps`
- `antishake`
- `fireworks`

The default mode is `auto`.

- Pick an exposure mode and try it out:

```
camera.start_preview()
camera.exposure_mode = 'beach'
sleep(5)
camera.capture('/home/pi/Desktop/beach.jpg')
camera.stop_preview()
```

-
  You can loop over all the exposure modes with `camera.EXPOSURE_MODES`, like you did for the image effects.

Change the image white balance

You can use `camera.awb_mode` to set the auto white balance to a preset mode.
The available auto white balance modes are:

- `off`
- `auto`
- `sunlight`
- `cloudy`
- `shade`
- `tungsten`
- `fluorescent`
- `incandescent`
- `flash`
- `horizon`

The default is `auto`.

- Pick an auto white balance mode and try it out:

```
camera.start_preview()
camera.awb_mode = 'sunlight'
sleep(5)
camera.capture('/home/pi/Desktop/sunlight.jpg')
camera.stop_preview()
```

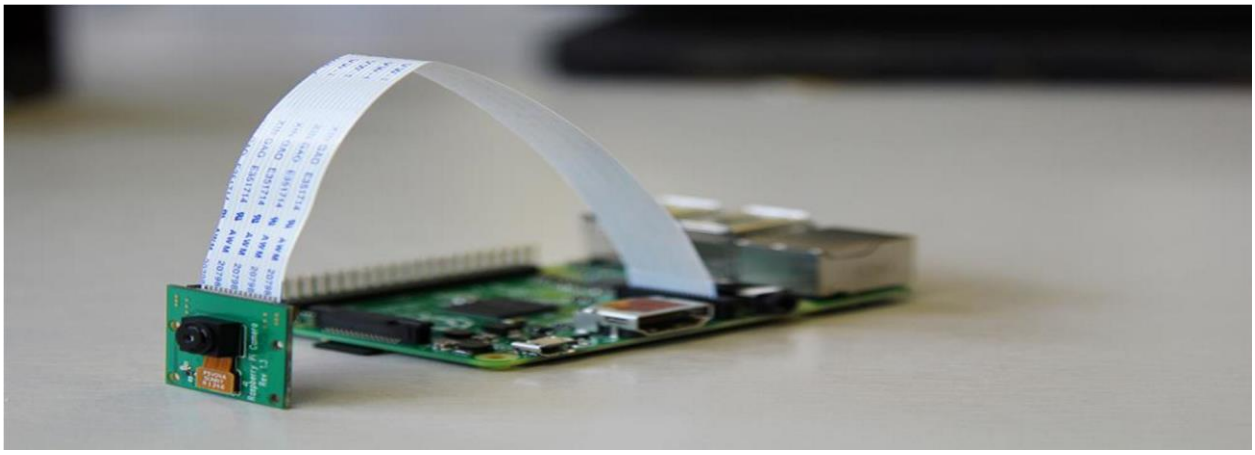
- You can loop over all the auto white balance modes with `camera.AWB_MODES`, like you did for the image effects.

Practical No: 04

Push Button Stop Motion:

Connect the camera

Before booting your Pi, you'll need to connect the camera.



- Loc ate the camera port next to the Ethernet port. Lift the tab on the top.
- Plac e the strip in the connector, with the blue side facing the Ethernet port. While holding the strip in plac e, push down the tab.
- Turn the power on to boot the Pi.

Test the camera

- Ope n a terminal window from the application menu. Enter the following command:

```
ras pistill -k
```

You should see a preview appear on the screen. It doesn't matter if the picture is upside-down; you can conf igure this later. Press `Ctrl + C` to exit the preview.

- To save an image you can use the following command:

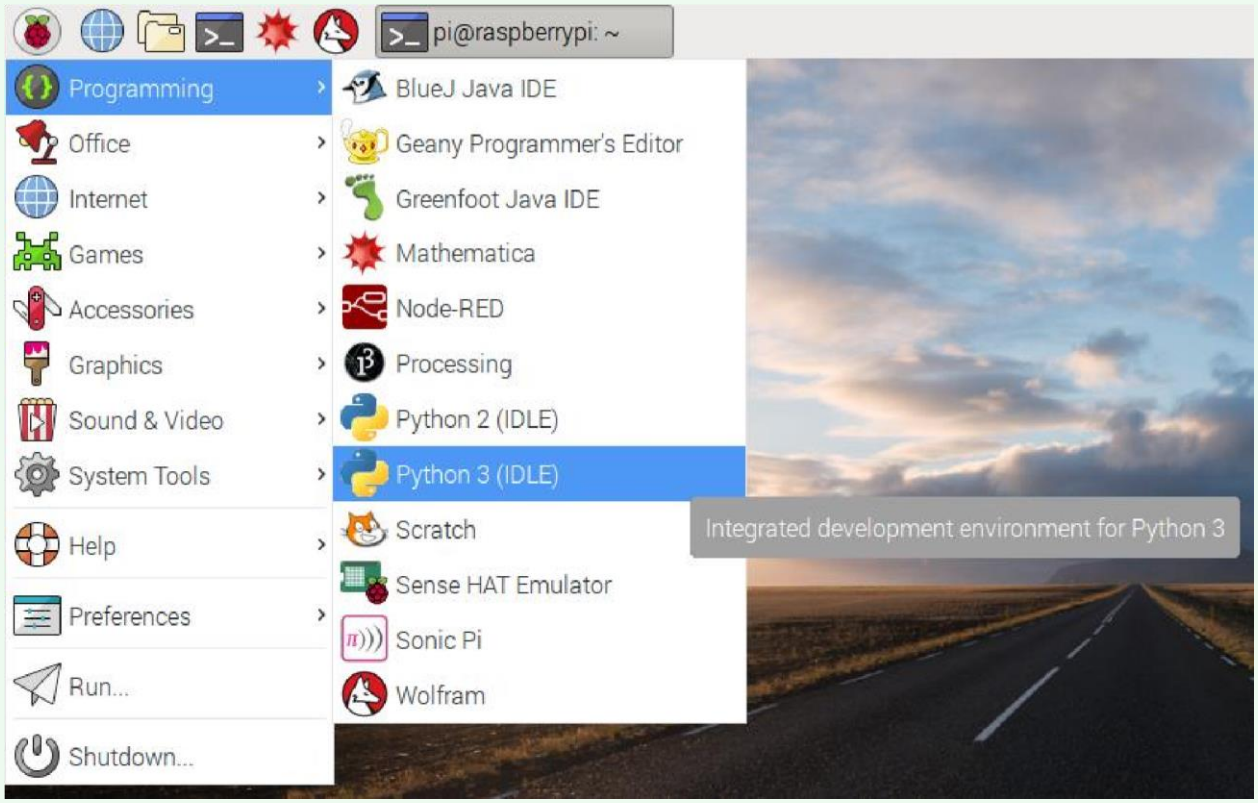
```
raspistill -o image1.jpg
```

Run the command `ls` to see the files in your home directory; you should see `image1.jpg` listed.

Click the file manager icon in the taskbar and you should see some folders and files. Double-click `image1.jpg` to preview it.

Take a picture with Python

Open Python 3 (IDLE) from the main menu:



Select `File > New Window` from the menu to open a Python file editor.

Carefully enter the following code into the new window (case is important!):

```
from picamera import PiCamera from time
import sleep

camera = PiCamera()

camera.start_preview() sleep(3)
camera.capture('/home/pi/Desktop/image.jpg') camera.stop_preview()
```

Sele ct `File > Save` from the menu (or `Ctrl + S`) and save as `animation.py`. press

Pres s `F5` to run your program.

You should see `image.jpg` saved on your Desktop. Double e picture is upside down you can either reposition your camera using a mount, or leave it as it is and click the icon to open the image.

If th - ython to flip the image. To do this, add the following lines:

tell P

```
cam era.rotation = 180
```

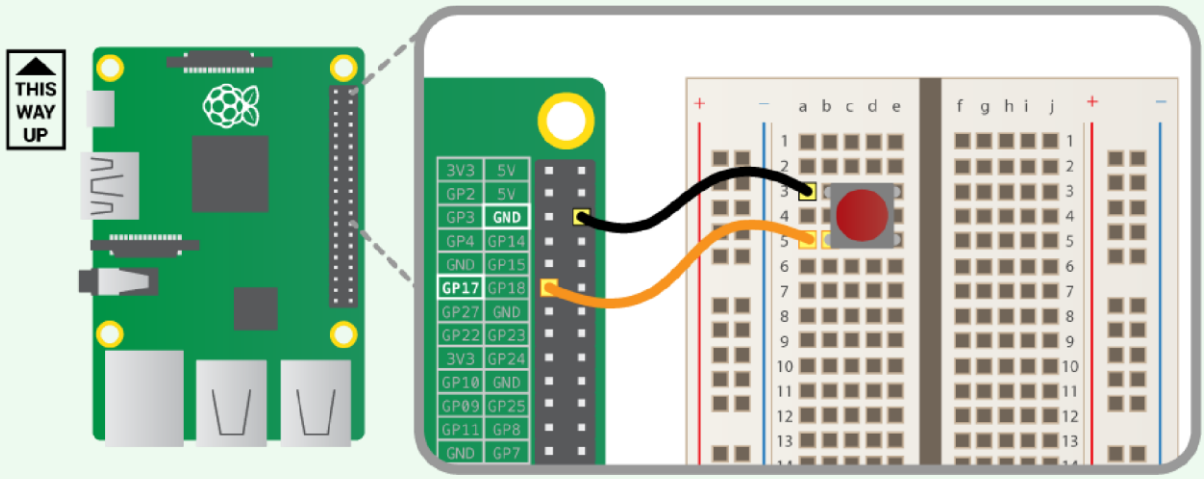
after `camera = PiCamera()`, so it becomes:

```
from  picamera import PiCamera  time
from import sleep era = PiCamera()
 cam
cam
cam era.rotation = 180
cam era.start_preview() ep(3)
sle era.capture('/home/pi/Desktop/image.jpg') era.stop_preview()
cam
cam
```

Run the file again and it will overwrite `image.jpg` with a new image in the correct orientation. Remember to keep these lines in your code while you alter it in the next few steps.

Connect a hardware button

Using your breadboard and jumper leads, connect the Pi to the button as shown in the diagram below:



Import `Button` from the `gpiozero` module at the top of the code, create up a `Button` connected to pin 17, and change the `sleep` line to use `button.wait_for_press` like so:

```
fro  picamera import PiCamera
m   time import sleep  gpiozero
fro import Button
m
fro ton = Button(17)
m   era = PiCamera()

but era.start_preview()
cam ton.wait_for_press()
cam era.capture('/home/pi/image.jpg')
but era.stop_preview()
cam
cam
```

Save and run your program.

Once the preview has started, press the button connected to your Pi to capture an image.

Return to the file manager window and you should see `image.jpg`. Again, double-click to view your

Take a selfie

If you want to take a photograph of yourself with the camera board, you are going to have to add in a delay to enable you to get into position. You can do this by modifying your program.

- Add a line to your code to tell the program to sleep briefly before capturing an image, as below:

```
camera.start_preview() button.wait_for_press()

sleep(3)
camera.capture('/home/pi/Desktop/image.jpg') camera.stop_preview()
```

Save and run your program.

- Pres s the button and try to take a selfie. Be sure to keep the camera still! Ideally, it should be mounted in posi tion.
- Again, feel free to check the image in the file manager. You can run the program again to take another selfie.

Stop motion animation

Now that you have successfully taken individual photographs with your camera, it's time to try combining a series of still images to make a stop motion animation.

- IMP ORTANT You must create a new folder to store your stills. In the terminal window, enter `mkdir ani mation`.
- Mod ify your code to add a loop to keep taking pictures every time the button is pressed:

```
cam era.start_preview()
fra me = 1 le True:
whi  try:
    button.wait_for_press()
         camera.capture('/home/pi/animation/frame%03d.jpg' % frame)
    frame += 1
     except KeyboardInterrupt:
    camera.stop_preview()
    break
```

Because while True goes on forever, you have to be able to make it exit gracefully. Using try and except means it can deal with an exceptional circumstance - if you force it to stop with Ctrl + C it will close the camera preview and exit the loop frame%03d means the file will be saved as the name "frame" followed by a 3-digit number with leading zeroes - 001, 002, 003, etc. This allows them to be easily sorted into the correct order for the video.

Now set up your animation subject (e.g. LEGO), ready to start the stop motion animation.

Press the button to capture the first frame, then rearrange the animation subject and press the button again to capture each subsequent frame.

Once all the frames have been captured, press Ctrl + C to terminate the program.

Open the `animation` folder in the file manager to see your stills collection.

Generate the video

To generate the video, begin by returning to the terminal window.

Run the video rendering command:

```
ffmpeg -r 10 -i animation/frame%03d.jpg -qscale 2 animation.mp4
```

Note you're using %03d again - this is a common format which both Python and ffmpeg understand, and means the photos will be passed in to the video in order.

Play your video using vlc.

```
vlc animation.mp4
```

You can adjust the frame rate by editing the rendering command. Try changing -r 10 (10 frames per second) to another number.

You can also change the filename of the rendered video to stop it from overwriting your first attempt. To do this, change `animation.h264` to something else.

Practical No: 05

Turtle Race:

Introduction

Use loops to draw a race track and create a racing turtle game.

What you will make

This project introduces for loops through a fun turtle race game. Loops are used to draw the race track and to make the turtles move a random number of steps each turn. If you have a group of people to play the game, each person pick a turtle and the one that gets the furthest is the winner.

What you will need

Hardware ○ An internet-connected

computer

Software

This project uses Python 3. We recommend using Trinket, which allows you to write Python code online.

Race track

You're going to create a game with racing turtles. First they'll need a race track.

- Open the blank Python template Trinket: jumpto.cc/python-new. ○ Add the following code to draw a line using the 'turtle':

```
from turtle import *
forward(100)
```

- Now let's use the turtle to draw some track markings for the race.

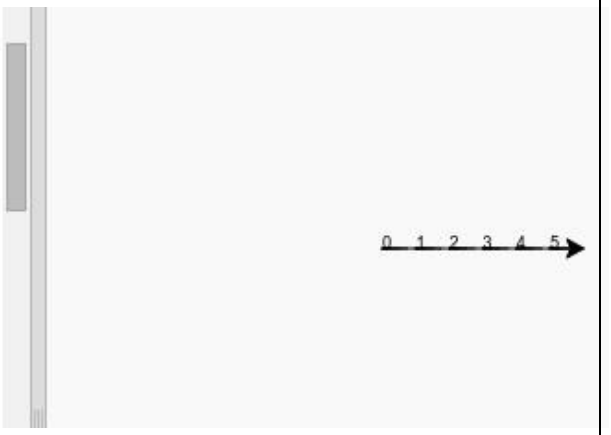
  The turtle `write` function writes text to the screen.
  Try it:

```
from turtle import *

write(0)
forward(100)
write(5)
```

- Now you need to fill in the numbers in between to create markings:

```
write(0)
forward(20)
write(1)
forward(20)
write(2)
forward(20)
write(3)
forward(20)
write(4)
forward(20)
write(5)
forward(20)
```



- Did you notice that your code is very repetitive? The only thing that changes is the number to write.

  There's a better way of doing this in Python. You can use a `for` loop.
  Update your code to use a `for` loop:

```
from turtle import *

for step in range(5):
  write(step)
  forward(20)
```

- Hmm, that only prints numbers up to 4. In Python `range(5)` returns five numbers, from 0 up to 4. To get it to also return 5 you'll need to use `range(6)`:

```
for step in range(6):
  write(step)
  forward(20)
```

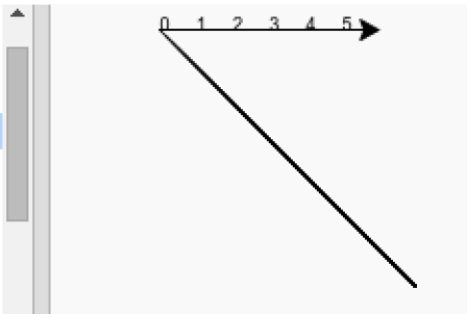
- Now we can draw some track markings. The turtle starts at coordinates (0,0) in the middle of the screen.

  Move the turtle to the top left instead:

```
from turtle import *

goto(-140, 140)

for step in range(6):
  write(step)
  forward(20)
```



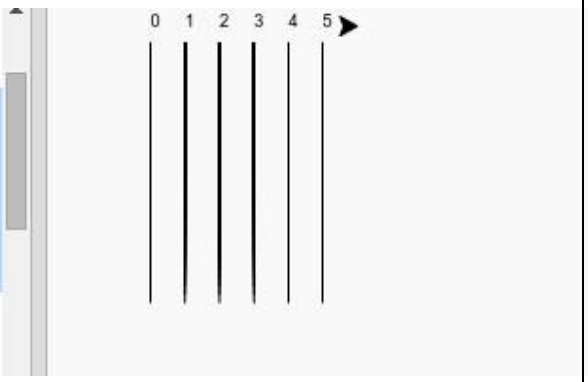
- Ah, you'll want to lift the pen up first!

```
penup()
goto(-140, 140)

for step in range(6):
  write(step)
  forward(20)
```

- Instead of drawing a line horizontally, let's draw vertical lines to create a track:

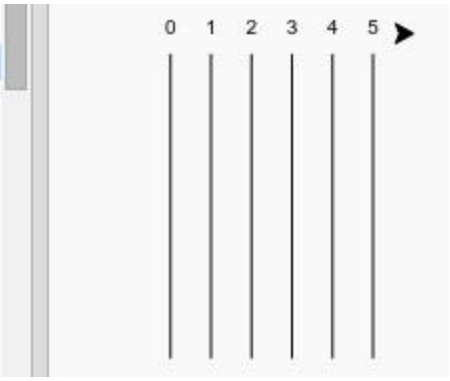
```
for step in range(6):
  write(step)
  right(90)
  forward(10)
  pendown()
  forward(150)
  penup()
  backward(160)
  left(90)
  forward(20)
```



`right(90)` makes the turtle turn right 90 degrees (a right angle.)
Moving `forward(10)` before putting the pen down leaves a small gap between the number and the start of the line. After drawing the line you lift up the pen and go `backward(160)` the length of the line plus the gap.

- It looks neater if you centre the numbers:

```
for step in range(6):
  write(step, align='center')
  right(90)
  forward(10)
  pendown()
  forward(150)
  penup()
  backward(160)
  left(90)
  forward(20)
```



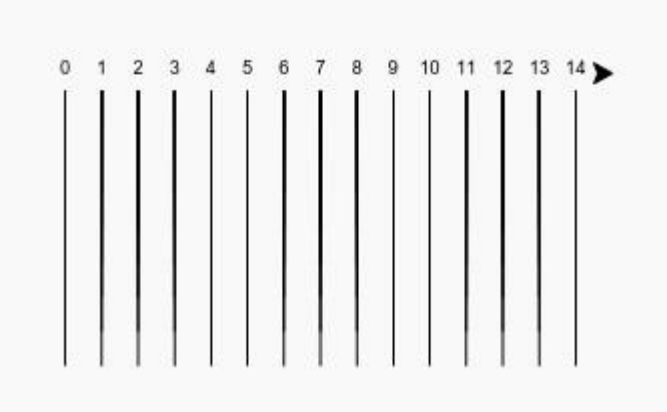
- And you can speed up the turtle so it draws faster:

```
from turtle import *

speed(10)
penup()
goto(-140, 140)
```

Challenge: More lines

Can you change your code so that the track lines go right across the screen?



If you want to make the turtle go even faster you can use `speed(0)`.

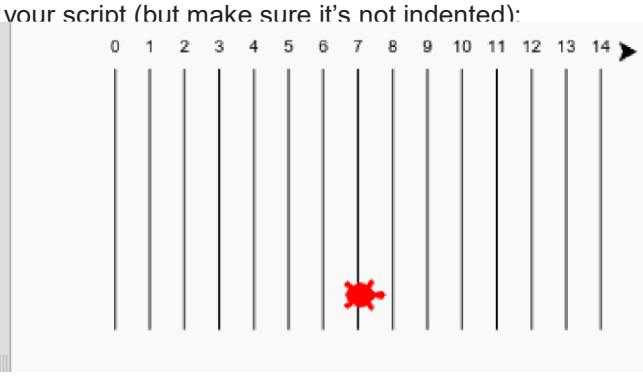
Racing turtles

Now for the fun bit. Let's add some racing turtles. It would be really boring if the turtles did the same thing every time so they will move a random number of steps each turn. The winner is the turtle that gets the furthest in 100 turns.

- When you use commands like `forward(20)` you are using a single turtle. But you can create more turtles. Add the following code to the end of your script (but make sure it's not indented):

```
forward(20)

ada = Turtle()
ada.color('red')
ada.shape('turtle')
```

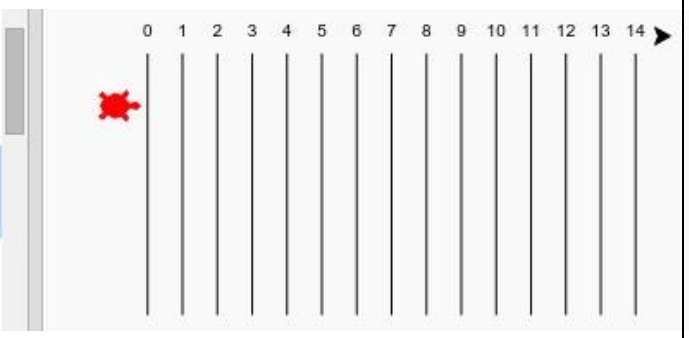


The first line creates a turtle called 'ada'. The next lines set the colour and shape of the turtle. Now it really looks like a turtle!

- Let's send the turtle to the starting line:

```
ada = Turtle()
ada.color('red')
ada.shape('turtle')

ada.penup()
ada.goto(-160, 100)
ada.pendown()
```



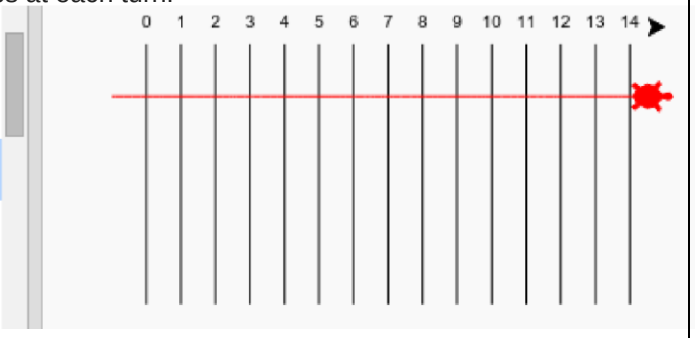
- Now you need to make the turtle race by moving a random number of steps at a time. You'll need the `randint` function from the Python `random` library. Add this `import` line to the top of your script:

```
from turtle import *
from random import randint
```

- The `randint` function returns a random integer (whole number) between the values chosen. The turtle will move forward 1, 2, 3, 4, or 5 steps at each turn.

```
ada.penup()
ada.goto(-160, 100)
ada.pendown()

for turn in range(100):
  ada.forward(randint(1,5))
```

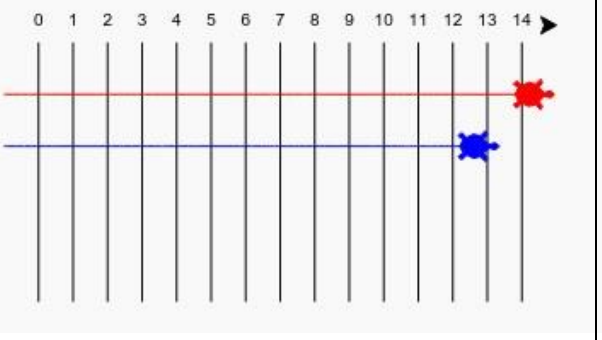


- One turtle isn't much of a race! Let's add another one:

```
bob = Turtle()
bob.color('blue')
bob.shape('turtle')

bob.penup()
bob.goto(-160, 70)
bob.pendown()

for turn in range(100):
  ada.forward(randint(1,5))
  bob.forward(randint(1,5))
```

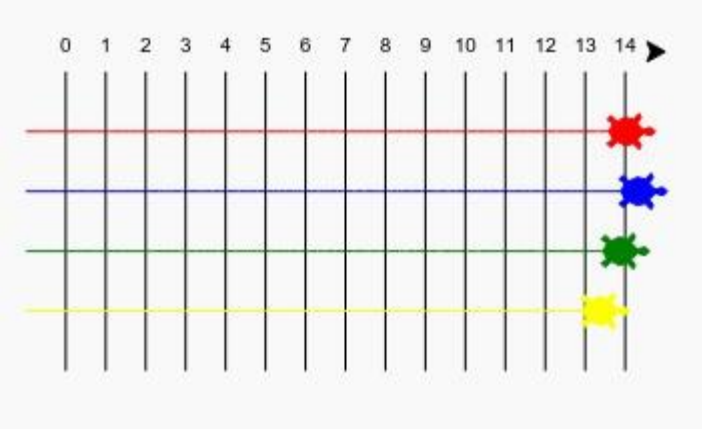


Note that the code for moving the blue turtle needs to be in the same `for` loop as the code for moving the red turtle so that they each make a move every turn.

Challenge: Race time!

Now you're ready to race. Pick a turtle and an opponent and see who wins.

Can you add more turtles so you can race with more friends?



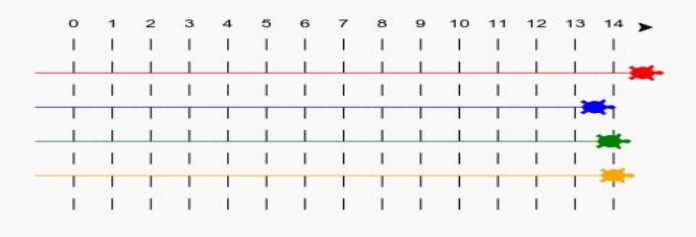
Colours include: orange, purple, violet, tomato, turquoise, magenta and brown - or you can go to jumpto.cc/colours and pick any colour you like!

Challenge: Do a twirl

Can you use a `for turn in range():` loop to make each turtle do a 360 degree twirl after they get to the starting line? You'll need to make sure they are facing in the right direction at the start of the race!
`ada.right(36)` will turn the red turtle right by 36 degrees.
Hint: A full turn is 360 degrees. A turtle could turn right 10 degrees 36 times, or left 5 degrees 72 times, or any other numbers make 360!

Challenge: Dashed lines

Can you use a loop to make the track lines dashed instead of solid?



Hint: Find the code that draws a straight line. Try using:`for forward() penup()` and `pendown()`

Practical No: 06

Input:

```
import random

when = ['A long time ago', 'Yesterday', 'Before you were born', 'In future', 'Before Thanos arrived']

who = ['Abhishek', 'Iron Man', 'Batman', 'Abhimanyu', 'Captain America']

went = ['Goa', 'America', 'Stark Tower', 'Prayagraj', 'Ayodhya']

what = ['to eat a lot of cakes', 'to fight for justice', 'to worship', 'to Wandering']

print(random.choice(when) + ', ' + random.choice(who) + ' went to ' + random.choice(went) + ' ' + random.choice(what) + '.')
```

Output:

Shell

```
In future, Abhishek went to Ayodhya to fight for justice.
>
```

Practical No: 07

Building a lamp server:

What you will make

Learn to set up a LAMP (Linux, Apache, MySQL, PHP) stack on your Raspberry Pi and configure it to work as a web server. You'll download and install WordPress and set up a basic website which you can access on any device on the same network as your Pi.

What you will learn

By following this resource and setting up a web server and WordPress website you will learn how to:

- Install software on your Raspberry Pi
- Install and configure Apache, PHP, and MySQL to create a LAMP web server
- Download WordPress and run it as a local website on your Raspberry Pi
- Configure WordPress and make your website accessible to other devices on your local network

What you will need

Hardware

- A Raspberry Pi computer connected to the internet
- An up to date install of the Raspberry Pi OS

Set up an Apache web server

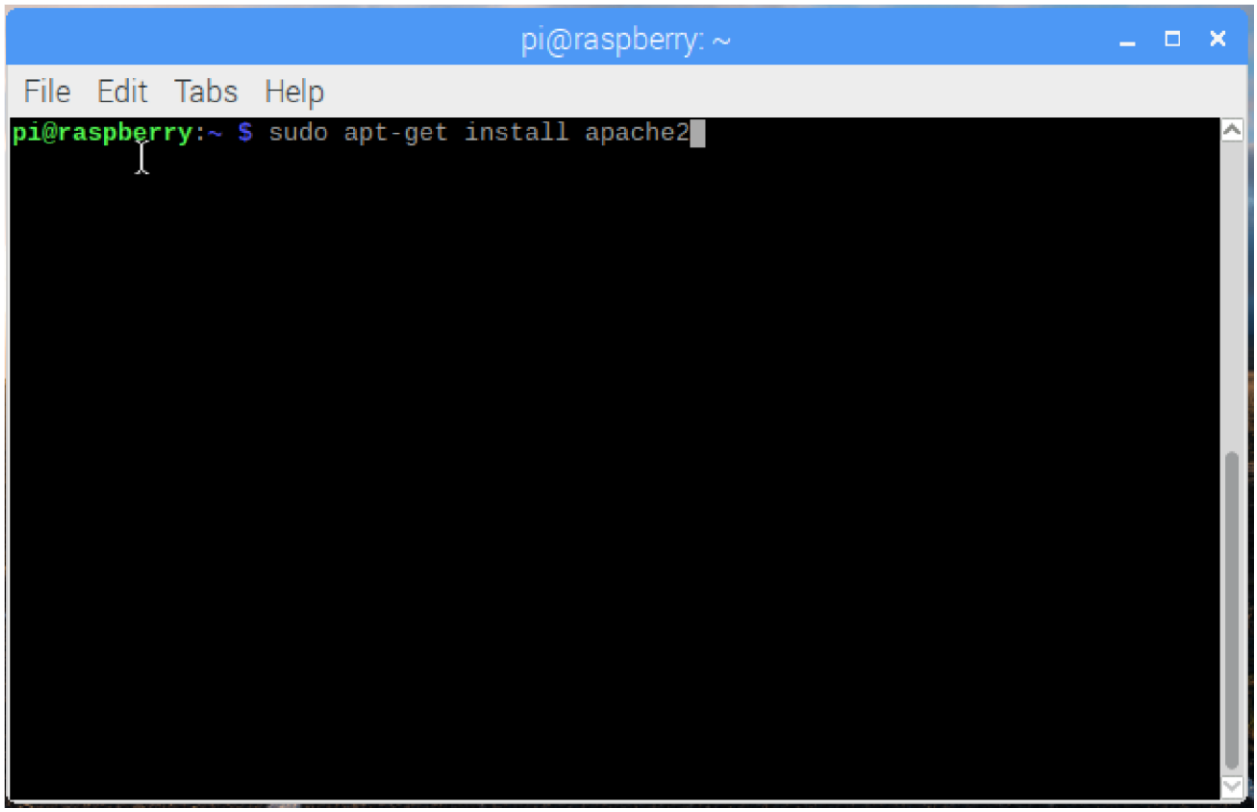
Apache is a popular web server application you can install on the Raspberry Pi to allow it to serve web pages.

On its own, Apache can serve HTML files over HTTP. With additional modules it can serve dynamic web pages using scripting languages such as PHP.

Install Apache ○ Open a terminal window by selecting Accessories > Terminal from the menu.

- Install the package by typing the following command into the terminal and pressing Enter:

```
sudo apt-get install apache2 -y
```



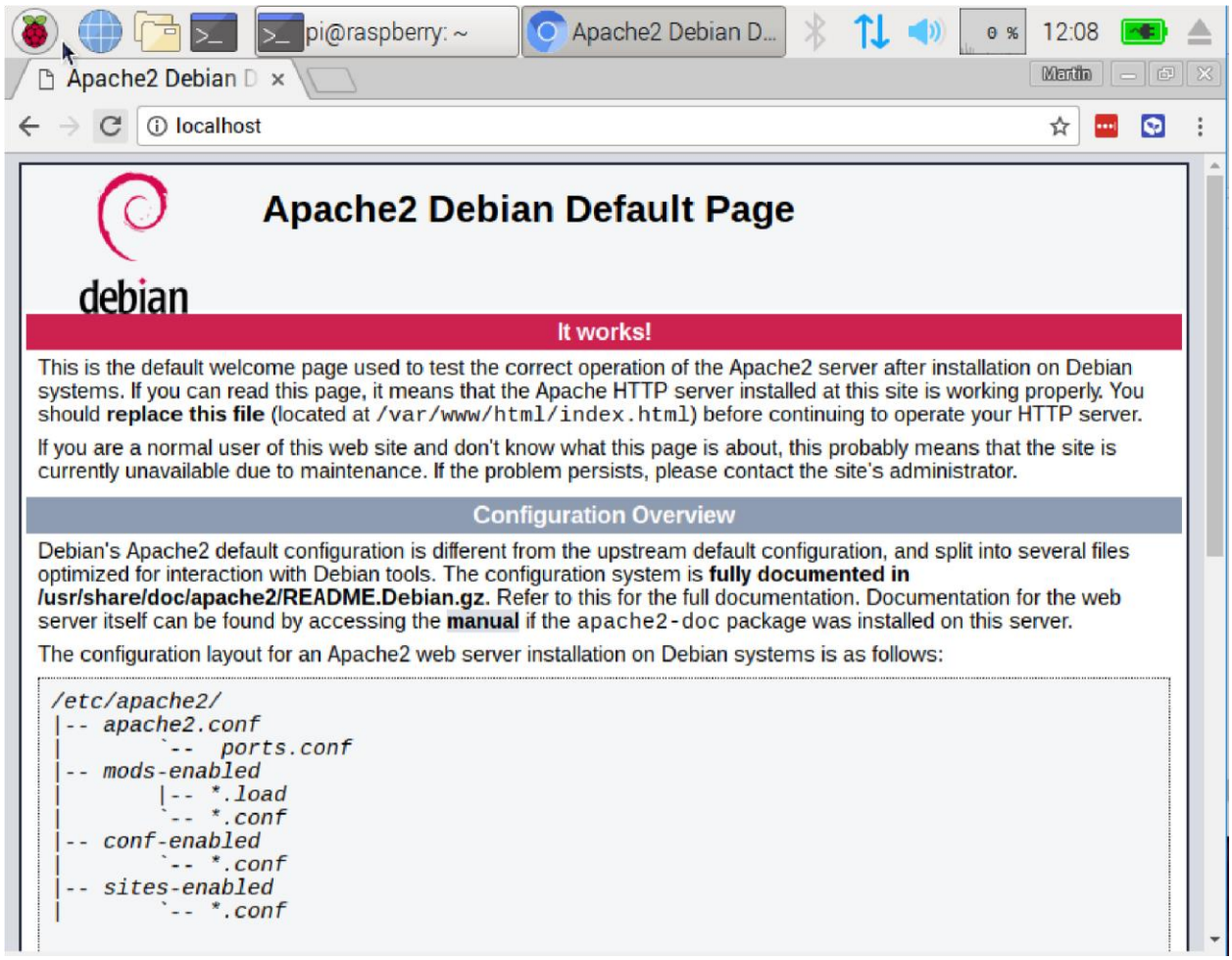
Test the web server

By default, Apache puts a test HTML file in the web folder that you will be able to view from your Pi or another computer on your network.

Open the Apache default web page on your Raspberry Pi:

- Open Chromium by selecting Internet > Chromium Web Browser from the menu.
- Enter the address `http://localhost`.

You should see this in your browser window:



This means you have Apache working!

You will also be able to open this web page from any other computer on your network using the IP address of your Raspberry Pi, e.g. `/192.168.1.10`
To find out your Raspberry Pi's IP address, type `hostname -I` into the terminal window. Your Raspberry Pi's IP address is a really useful and will allow you to remotely access it. Changing the default web page

This default web page is just a HTML file on the file system. It is located at `/var/www/html/index.html`

- Navigate to this directory in the terminal and have a look at what's inside:

```
cd /var/www/html
ls -al
```

You should see this in the window:

```
total 12 drwxr-xr-x  2 root root 4096 Jan  8
01:29 . drwxr-xr-x  3 root root 4096 Jan  8
01:28 ..
-rw-r--r--  1 root root  177 Jan  8 01:29 index.html
```

This shows that there is one file in called . refers to the directory

itself , refers to the parent directory .
and

What the columns mean

1. The permissions of the file or directory
2. The number of files in the directory (or `1` if it's a file).
3. The user that owns the file or directory
4. The group that owns the file or directory
5. The size of the file or directory
6. The date and time of the last modification

As you can see, the `html` directory and `index.html` file are both owned by the `root` user, so you'll need to use `sudo` to edit them. You can edit this file using mousepad:

```
sudo mousepad index.html
```

If you make a change to the file, save it, and refresh the browser, you will see your change appear.

Install PHP

PHP is a preprocessor: it's code that runs when the server receives a request for a web page via a web browser. It works out what needs to be shown on the page, and then sends that page to the browser. Unlike static HTML, PHP can show different content under different circumstances. Other languages are also capable of doing this, but since WordPress is written in PHP, that's what we need to use this time. PHP is a very popular language on the web: huge projects like Facebook and Wikipedia are written in PHP.

- Install the PHP package with the following command:

```
sudo apt-get install php -y
```

Test PHP

- Create the file `index.php`

```
sudo mousepad index.php
```

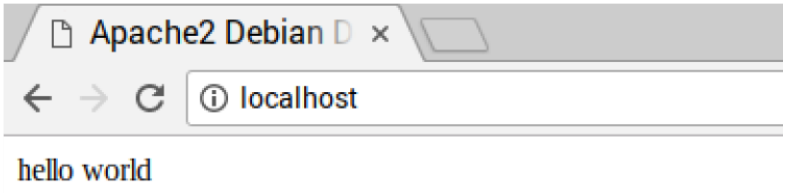
- Put some PHP content in it:

```
<?php echo "hello world"; ?>
```

- Save the file.
- Delete `index.html`, because it takes precedence over `index.php`:

```
sudo rm index.html
```

Refresh your browser. You should see "hello world". This page is not dynamic, but it is still served by PHP.



If you see the raw PHP above instead of “hello world”, reload and restart Apache like so:

```
sudo service apache2 restart
```

- Edit `index.php` to include some dynamic content, for example:

```
<?php echo date('Y-m-d H:i:s'); ?>
```

Or show your PHP info:

```
<?php phpinfo(); ?>
```

Install MariaDB

MariaDB is a popular database engine. Like PHP, it’s widely used on web servers, which is why projects like WordPress use it, and why those projects are so popular.

Install the MariaDB Server and PHP-MySQL packages by entering the following command into the terminal window:

```
sudo apt-get install mariadb-server php-mysql -y
```

Now restart Apache:

```
sudo service apache2 restart
```

Download WordPress

You can download WordPress from wordpress.org using the `wget` command. Helpfully, a copy of the latest version of WordPress is always available at wordpress.org/latest.tar.gz, so you can grab the latest version without having to look it up on the website. At the time of writing, this is version 4.5.

What is a .tar.gz file?

- Change directory to `/var/www/html/` and delete all the files in the folder.

```
cd /var/www/html/ sudo
rm *
```

- Download WordPress using `wget`

```
sudo wget http://wordpress.org/latest.tar.gz
```

-
  Extract the WordPress tarball to get at the WordPress files.

```
sudo tar xzf latest.tar.gz
```

- Move the contents of the extracted `wordpress` directory to the current directory.

```
sudo mv wordpress/* .
```

- Tidy up by removing the tarball and the now empty `wordpress` directory.

```
sudo rm -rf wordpress latest.tar.gz
```

- Running the `ls` or `tree -L 1` command now will show you the contents of a WordPress project:

```
.
├── index.php
├── license.txt
├── readme.html
├── wp-activate.php
├── wp-admin
├── wp-blog-header.php
├── wp-comments-post.php
├── wp-config-sample.php
├── wp-content
├── wp-cron.php
├── wp-includes
├── wp-links-opml.php
├── wp-load.php
├── wp-login.php
├── wp-mail.php
├── wp-settings.php
├── wp-signup.php
├── wp-trackback.php
└── xmlrpc.php

3 directories, 16 files
```

This is the source of a default WordPress installation. The files you edit to customise your installation belong in the `wp-content` folder.

- You should now change the ownership of all these files to the Apache user:

```
sudo chown -R www-data: .
```

Set up your WordPress Database

Set up MySQL/MariaDB

To get your WordPress site set up, you need a database. This is where MySQL and MariaDB come in!

-

Run the MySQL secure installation command in the terminal window.

```
sudo mysql_secure_installation
```

- You will be asked — press Enter.
- Type in Y and press Enter to `Set root password?`.
- Type in a password at the prompt, and press Enter. Important: remember this root password, as you will need it later to set up WordPress.
- Type in Y to `Remove anonymous users`.
- Type in Y to .
- Type in Y to .
- Type in Y to `Reload privilege tables now`.

When complete, you will see the message and .

Create the WordPress database

- Run `mysql` in the terminal window:

```
sudo mysql -uroot -p
```

- Enter the root password you created.

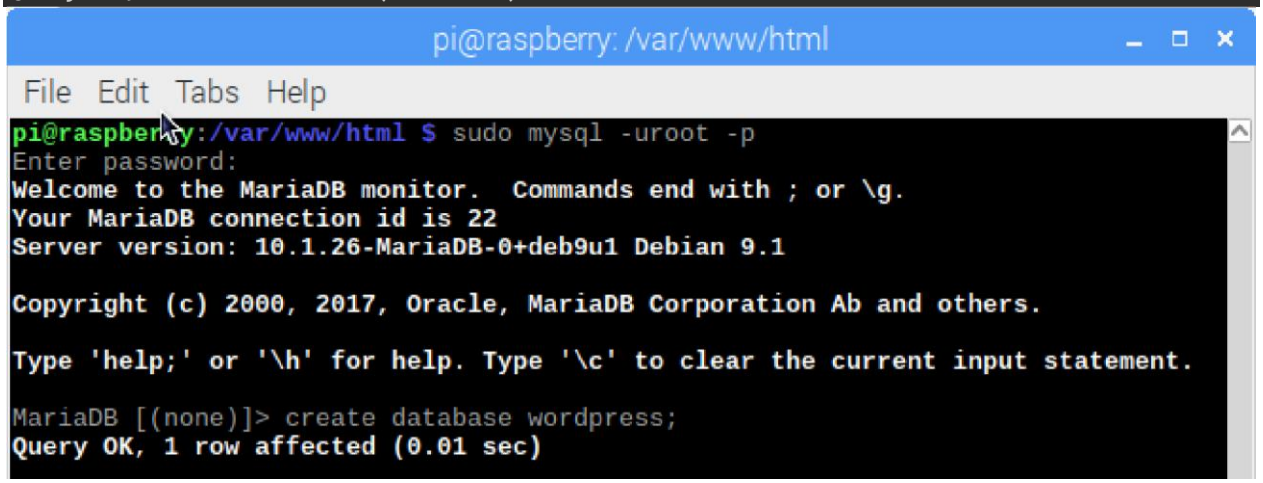
You will be greeted by the message `Welcome to the MariaDB monitor`

- Create the database for your WordPress installation at the `MariaDB [(none)]>` prompt using:

```
create database wordpress;
```

Note the semi-colon ending the statement.
If this has been successful, you should see this:

```
Query OK, 1 row affected (0.00 sec)
```



- Now grant database privileges to the root user. Note: you will need to enter your own password after `IDENTIFIED BY`

```
GRANT ○
ALL
PRIVILEGES ON wordpress.* TO 'root'@'localhost' IDENTIFIED BY 'YOURPASSWORD';
```

For the changes to take effect, you will need to flush the database privileges:

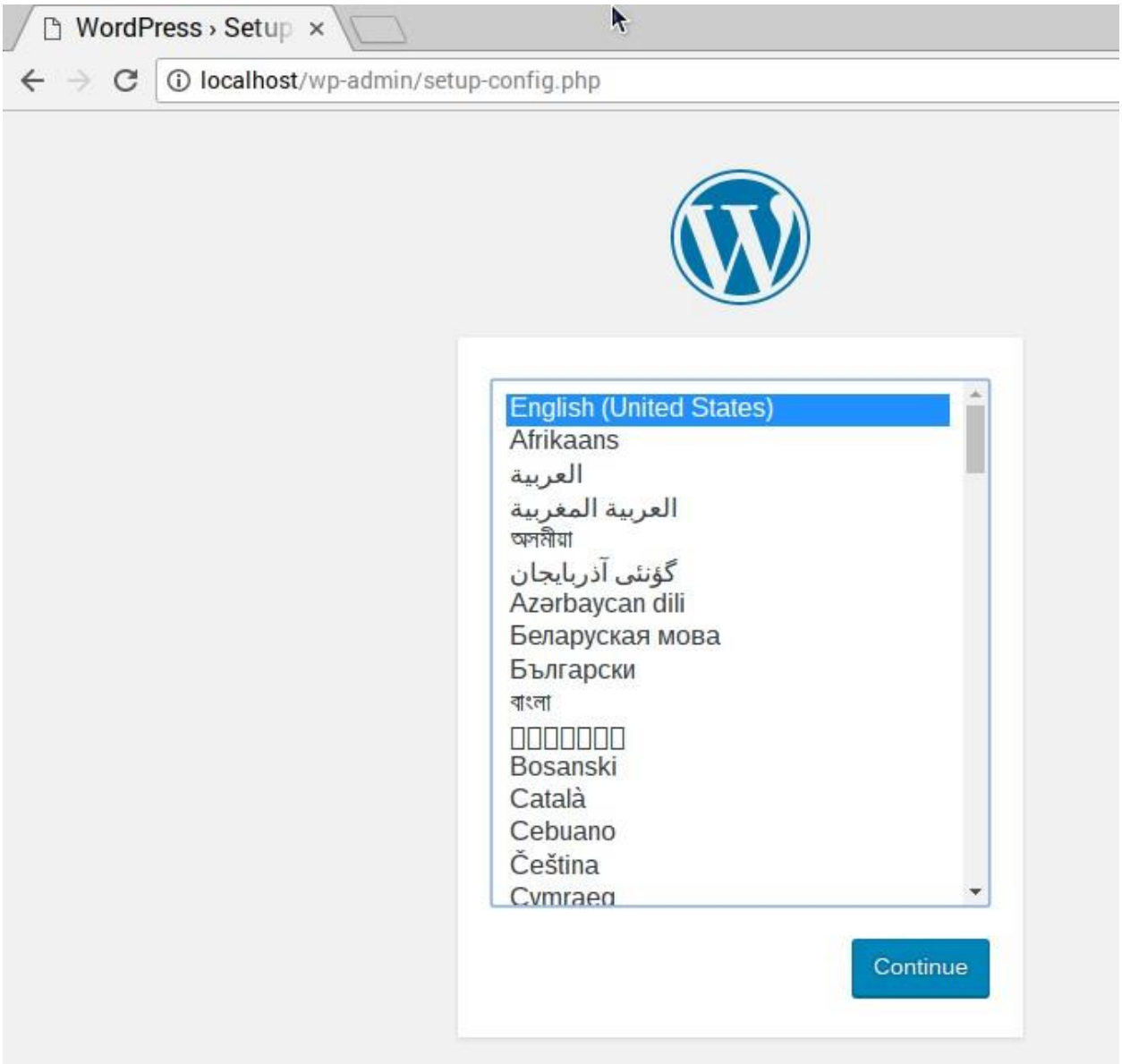
```
FLUSH PRIVILEGES;
```

- Exit the MariaDB prompt with `Ctrl` + `D`.
- Restart your Raspberry Pi:

```
sudo reboot
```

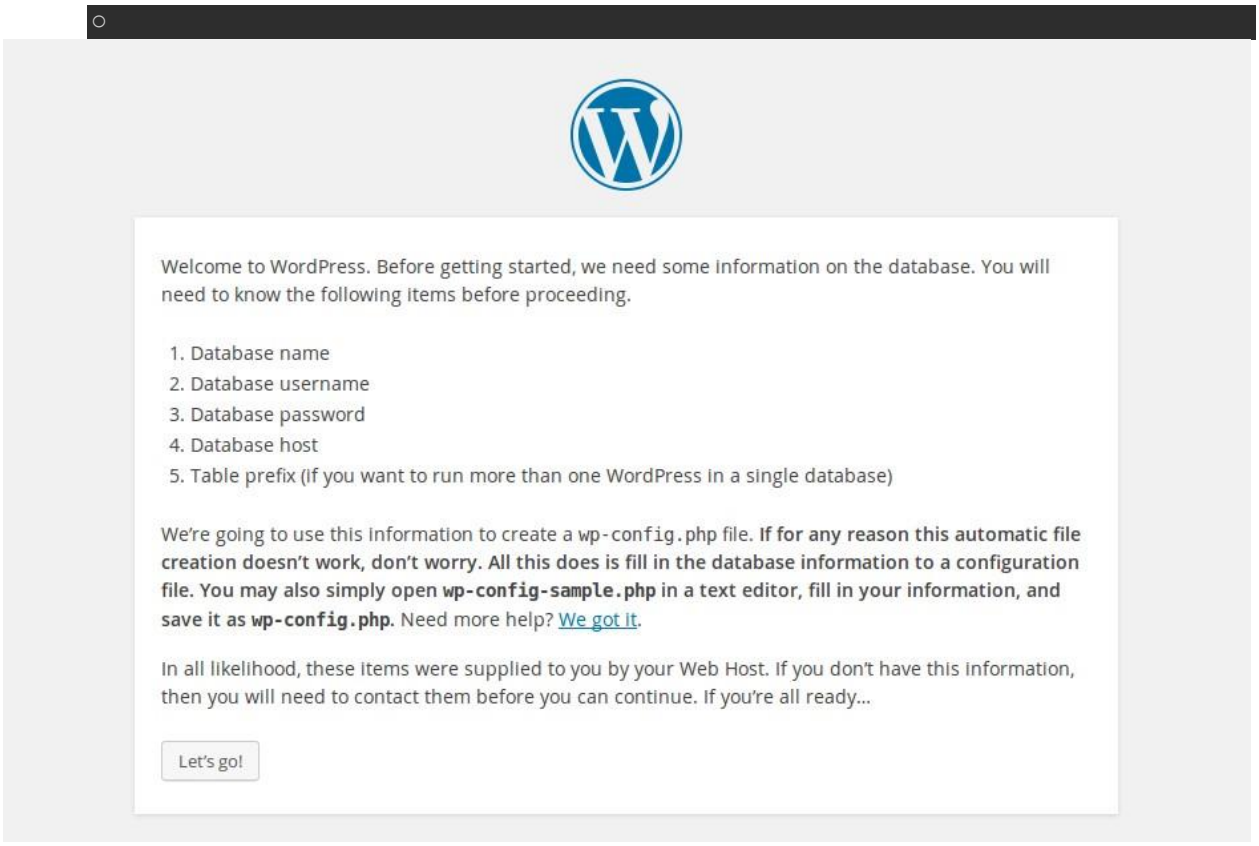
WordPress configuration

- Open the web browser on your Pi and goto `http://localhost`, you should see a WordPress page asking to pick your language.



- Select your language and click Continue.

You will be presented with the WordPress welcome screen.

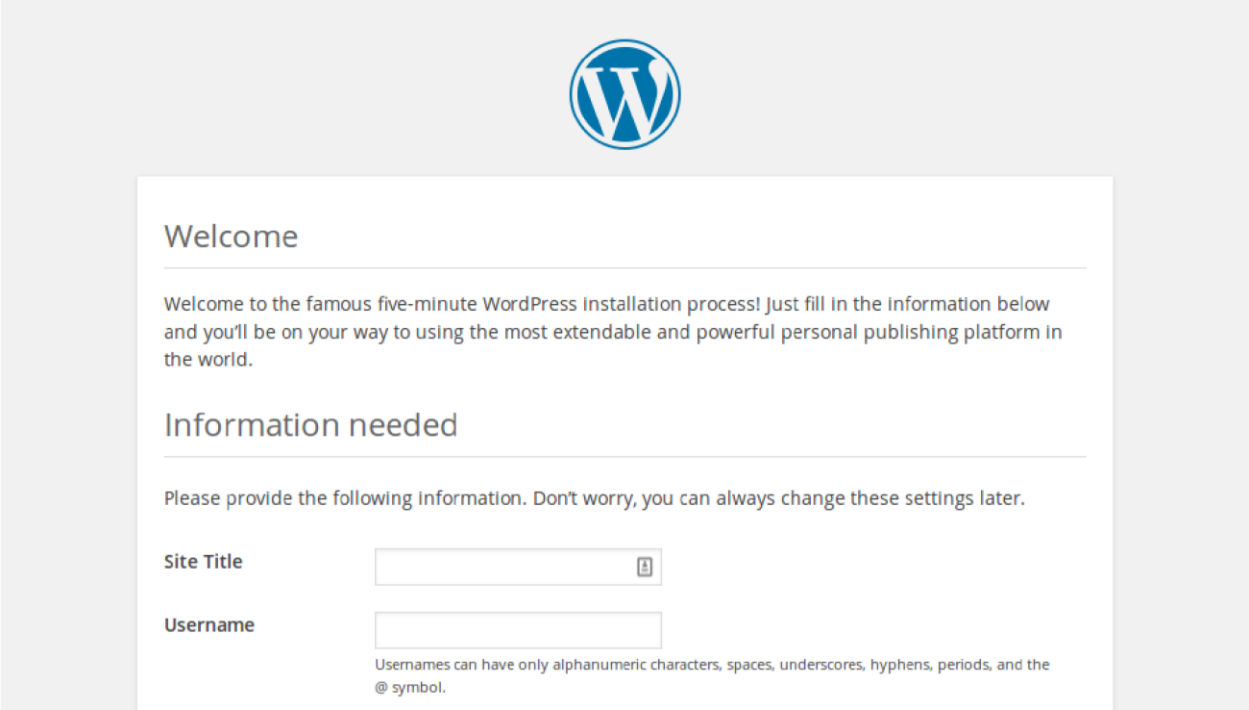


- Click the Let's go! button.
- Now fill out the basic site information as follows:

```
Database Name:      wordpress
User Name:          root
Password:           <YOUR PASSWORD>
Database Host:      localhost
Table Prefix:       wp
```

- Click Submit to proceed.
- Click the Run the install button.

Now you're getting close!



Fill out the information: give your site a title, create a username and password, and enter your email address. Hit the `Install WordPress` button, then log in using the account you just created.
Now you're logged in and have your site set up, you can see the website by visiting your `http://localhost/wp-admin`.

Log in to WordPress from another computer

Friendly permalinks

It's recommended that you change your permalink settings to make your URLs more friendly.

To do this, log in to WordPress and go to the dashboard.

- Go to Setting, then Permalinks.
- Select the Post name option and click Save Changes.

You'll need to enable Apache's `rewrite` mod:

```
sudo a2enmod rewrite
```

You'll also need to tell the virtual host serving the site to allow requests to be overwritten.

- Edit the Apache configuration file for your virtual host:

```
sudo mousepad /etc/apache2/sites-available/000-default.conf
```

- Add the following lines after line 1.

```
<Directory "/var/www/html">
    AllowOverride All </Directory>
```

- Ensure it's within the `<VirtualHost *:80>` like so:

```
<VirtualHost *:80>
    <Directory "/var/www/html">
        AllowOverride All
    </Directory>
    ...
```

- Save the file and exit.
- Restart Apache.

```
sudo service apache2 restart
```

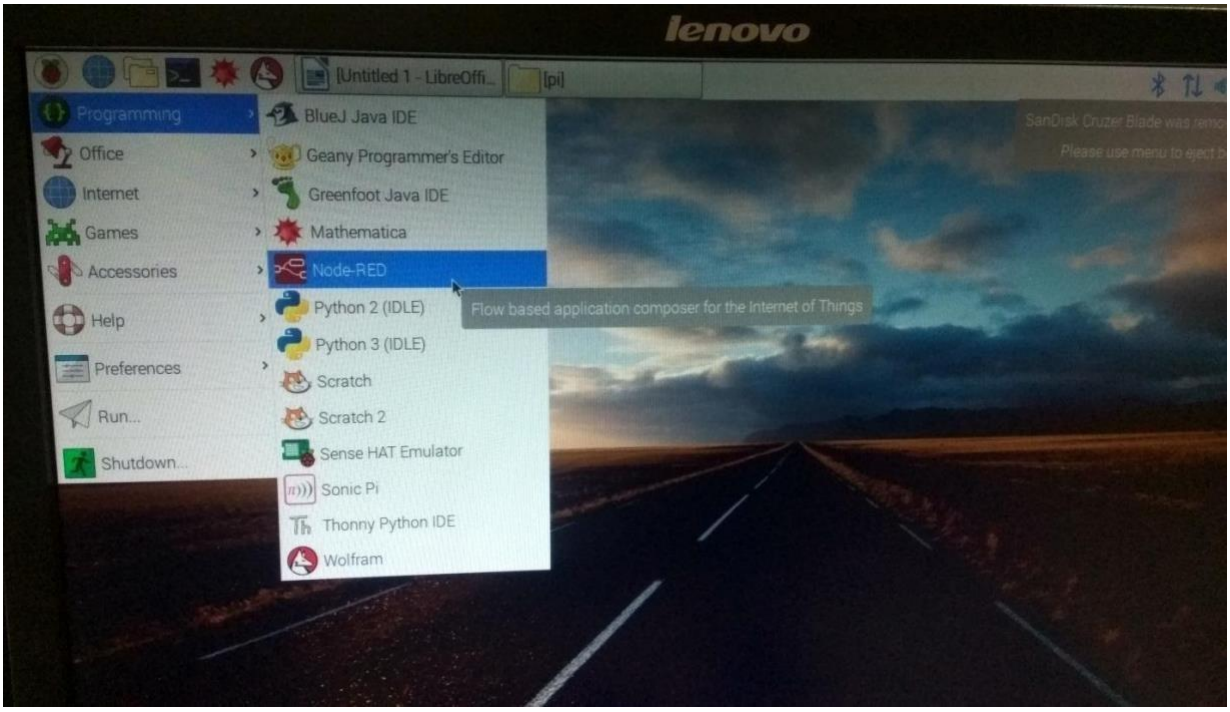
Customisation

WordPress is very customisable. By clicking your site name in the WordPress banner at the top of the page (when logged you're in), you'll be taken to the Dashboard. From there, you can change the theme, add pages and posts, edit the menu, add plugins, and lots more. This is just a taster for getting something interesting set up on the Raspberry Pi's web server.

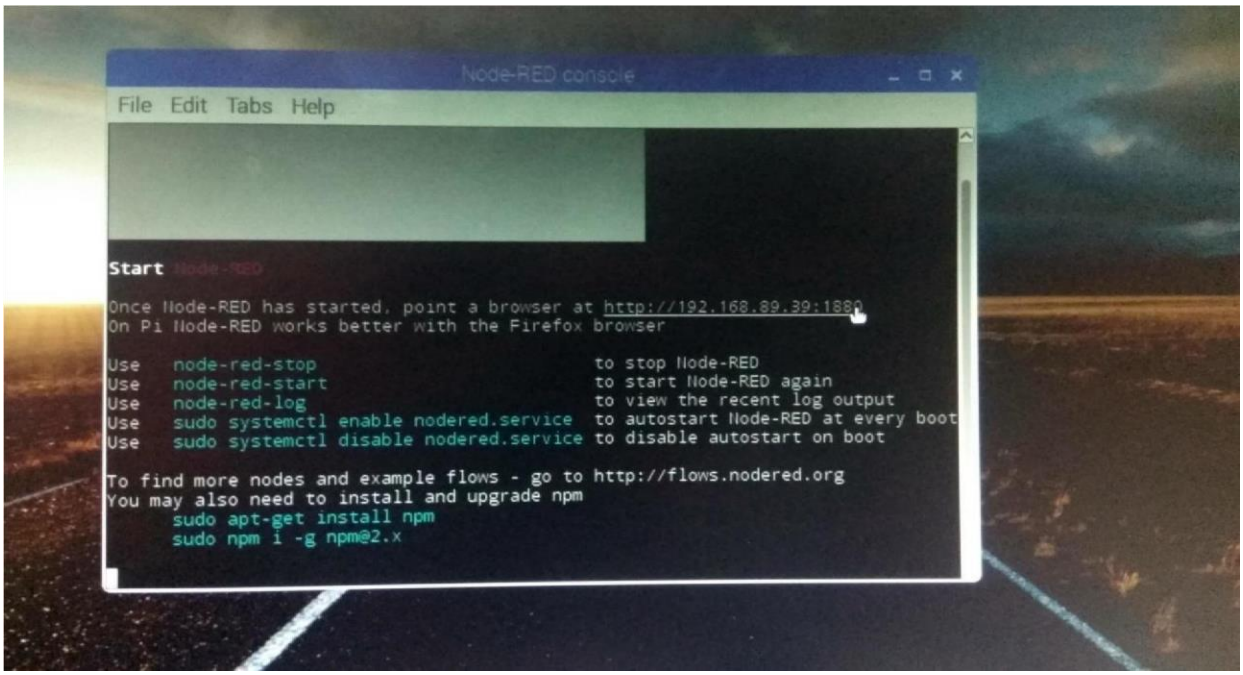
Practical No: 08

NODE Red: Connect LED to Internet of Things.

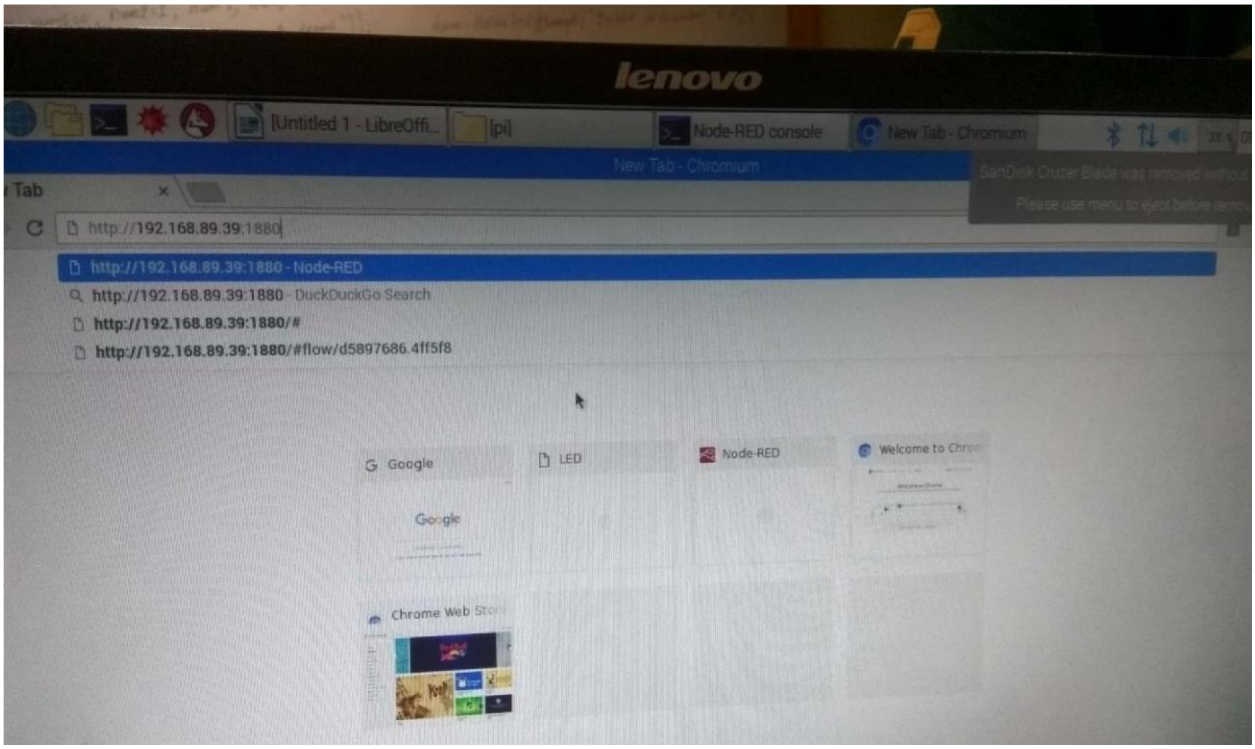
STEP 1:



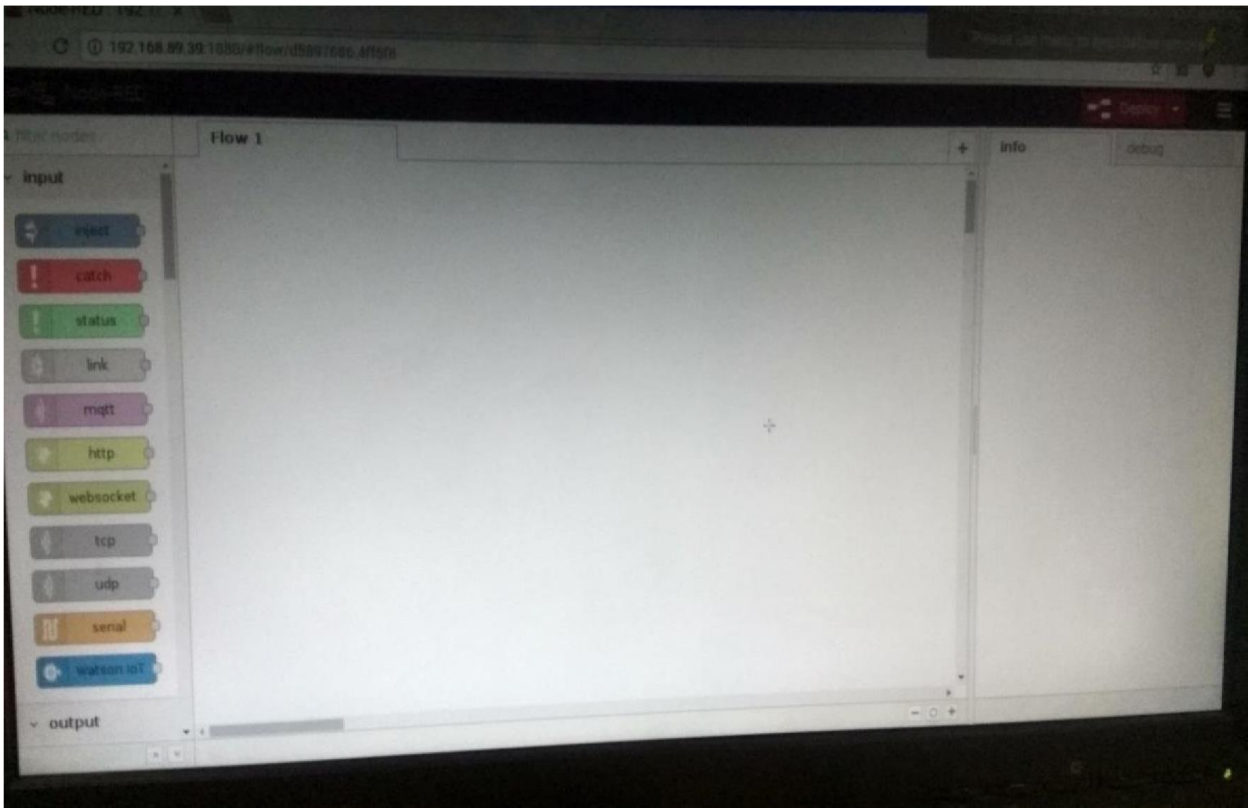
STEP 2:


Node-RED console
File Edit Tabs Help
Start Node-RED
Once Node-RED has started, point a browser at http://192.168.89.39:1880
On Pi Node-RED works better with the Firefox browser
Use node-red-stop to stop Node-RED
Use node-red-start to start Node-RED again
Use node-red-log to view the recent log output
Use sudo systemctl enable nodered.service to autostart Node-RED at every boot
Use sudo systemctl disable nodered.service to disable autostart on boot
To find more nodes and example flows - go to http://flows.nodered.org
You may also need to install and upgrade npm
sudo apt-get install npm
sudo npm i -g npm@2.x

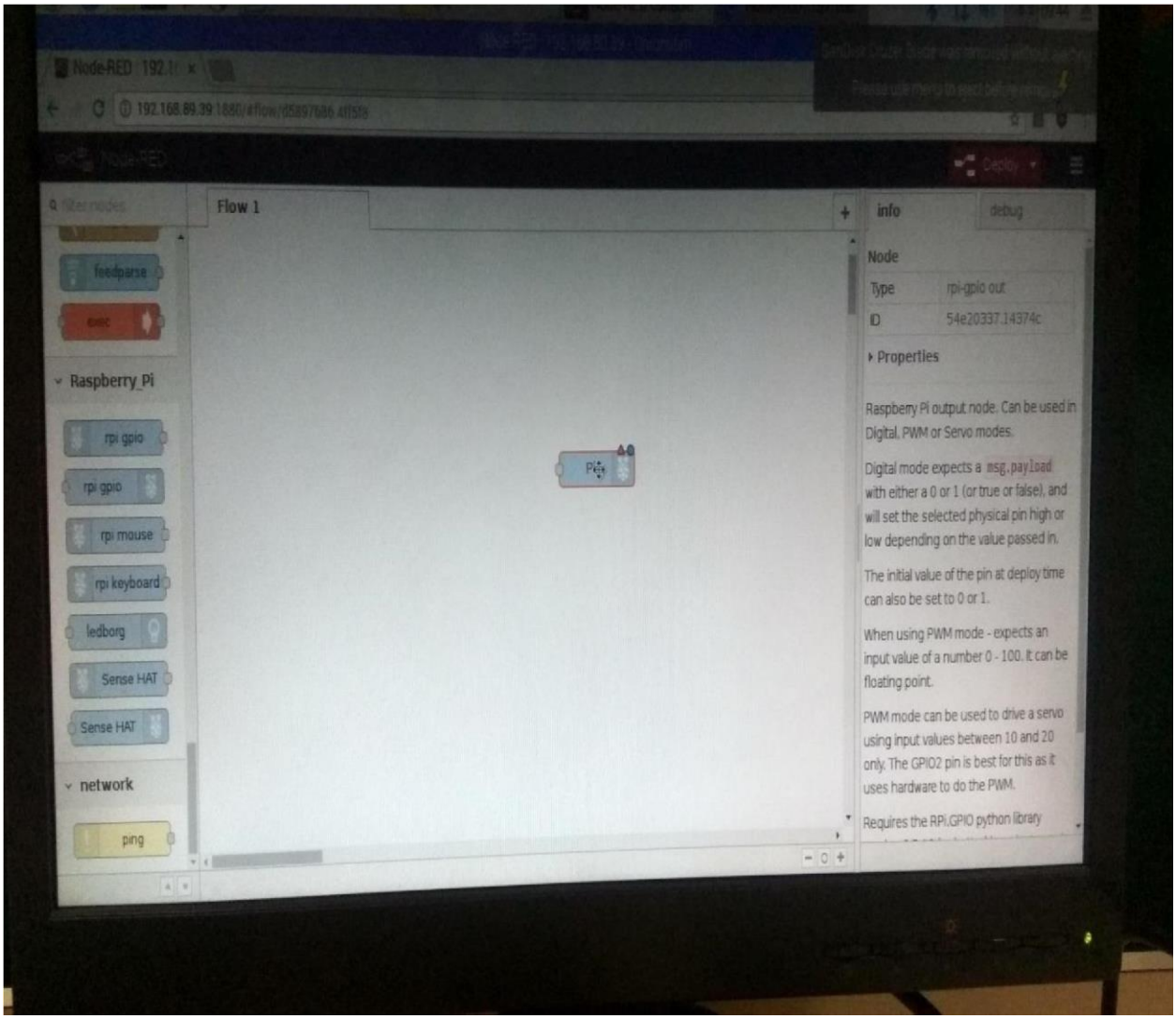
STEP 3:


lenovo
[Untitled 1 - LibreOffi...
[pi]
Node-RED console
New Tab - Chromium
http://192.168.89.39:1880
http://192.168.89.39:1880 - Node-RED
http://192.168.89.39:1880 - DuckDuckGo Search
http://192.168.89.39:1880/#
http://192.168.89.39:1880/#flow/d5897686.4ff5f8
Google
LED
Node-RED
Chrome Web Store

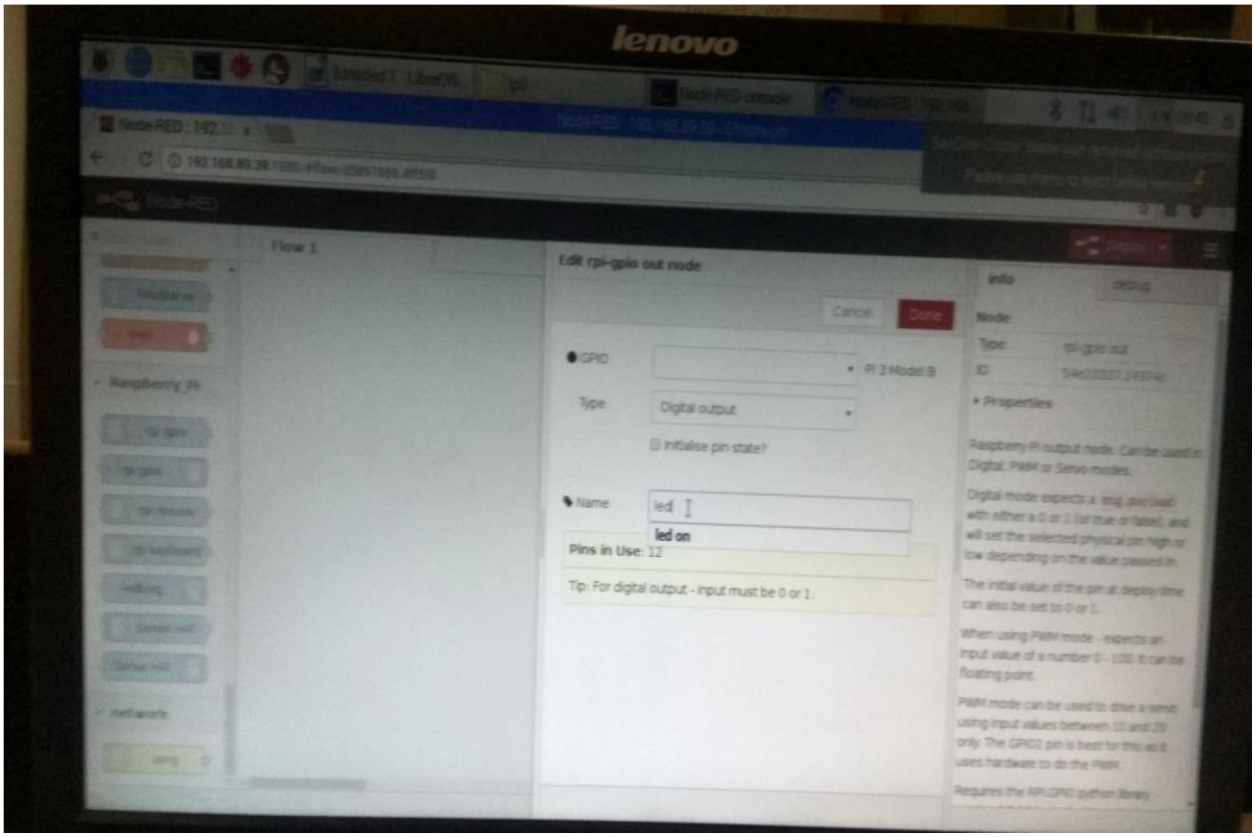
STEP 4:


Flow 1
info
debug
input
inject
catch
status
link
mqtt
http
websocket
tcp
udp
serial
watson iot
output

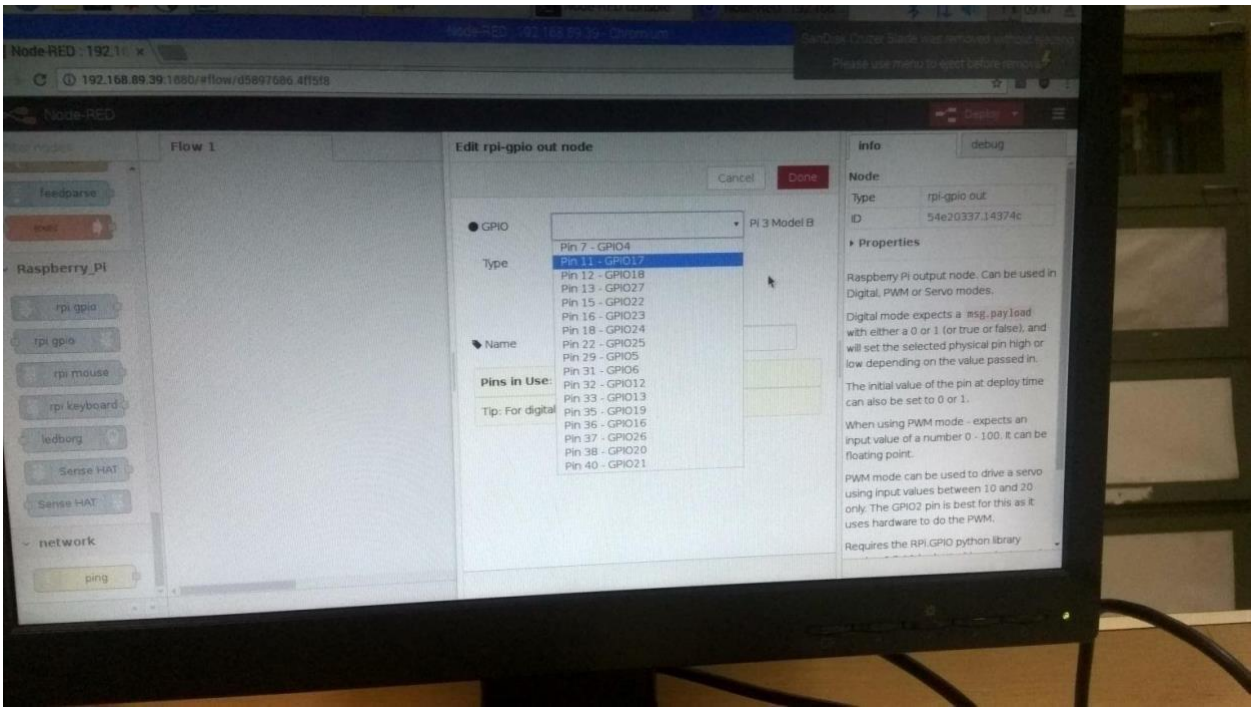
STEP 5:


Node-RED
Flow 1
feedparse
exec
Raspberry_Pi
rpi gpio
rpi gpio
rpi mouse
rpi keyboard
ledborg
Sense HAT
Sense HAT
network
ping
info
debug
Node
Type
rpi-gpio out
ID
54e20337.14374c
Properties
Raspberry Pi output node. Can be used in Digital, PWM or Servo modes.
Digital mode expects a msg.payload with either a 0 or 1 (or true or false), and will set the selected physical pin high or low depending on the value passed in.
The initial value of the pin at deploy time can also be set to 0 or 1.
When using PWM mode - expects an input value of a number 0 - 100. It can be floating point.
PWM mode can be used to drive a servo using input values between 10 and 20 only. The GPIO2 pin is best for this as it uses hardware to do the PWM.
Requires the RPi.GPIO python library
Deploy

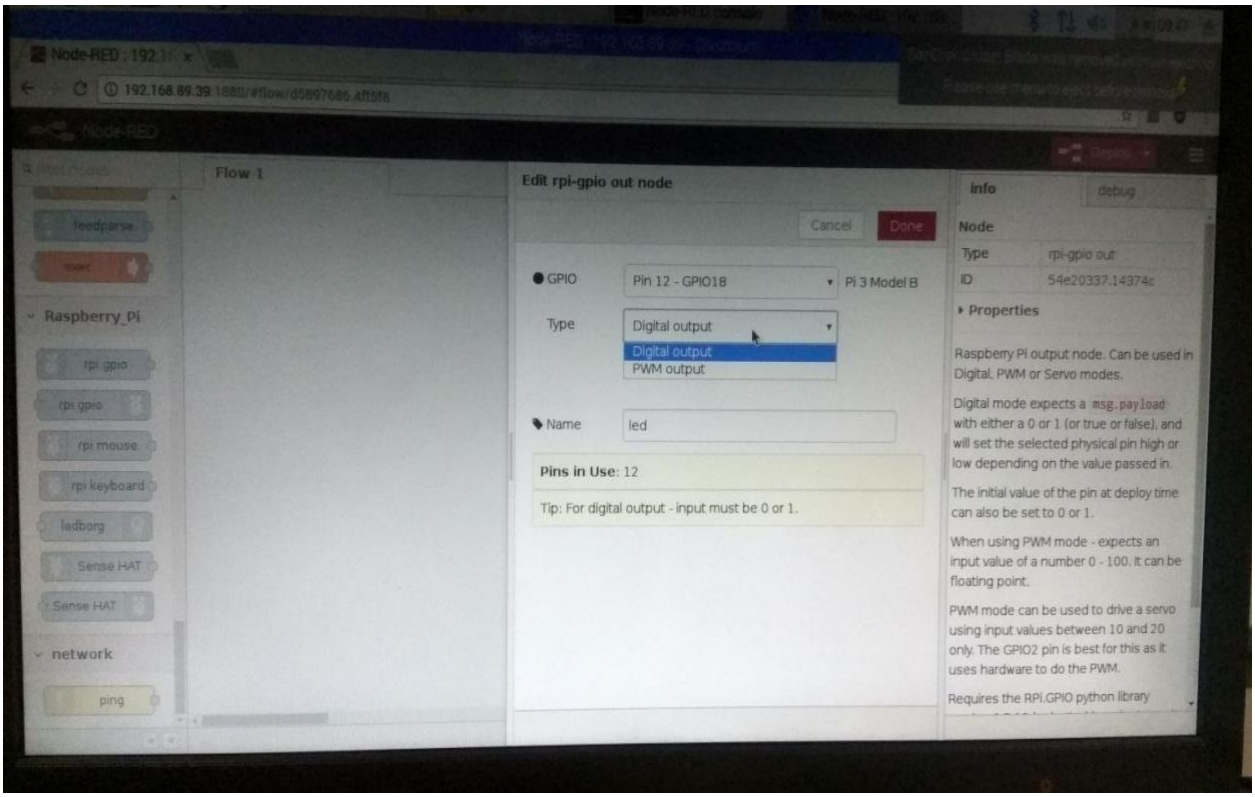
STEP 6:



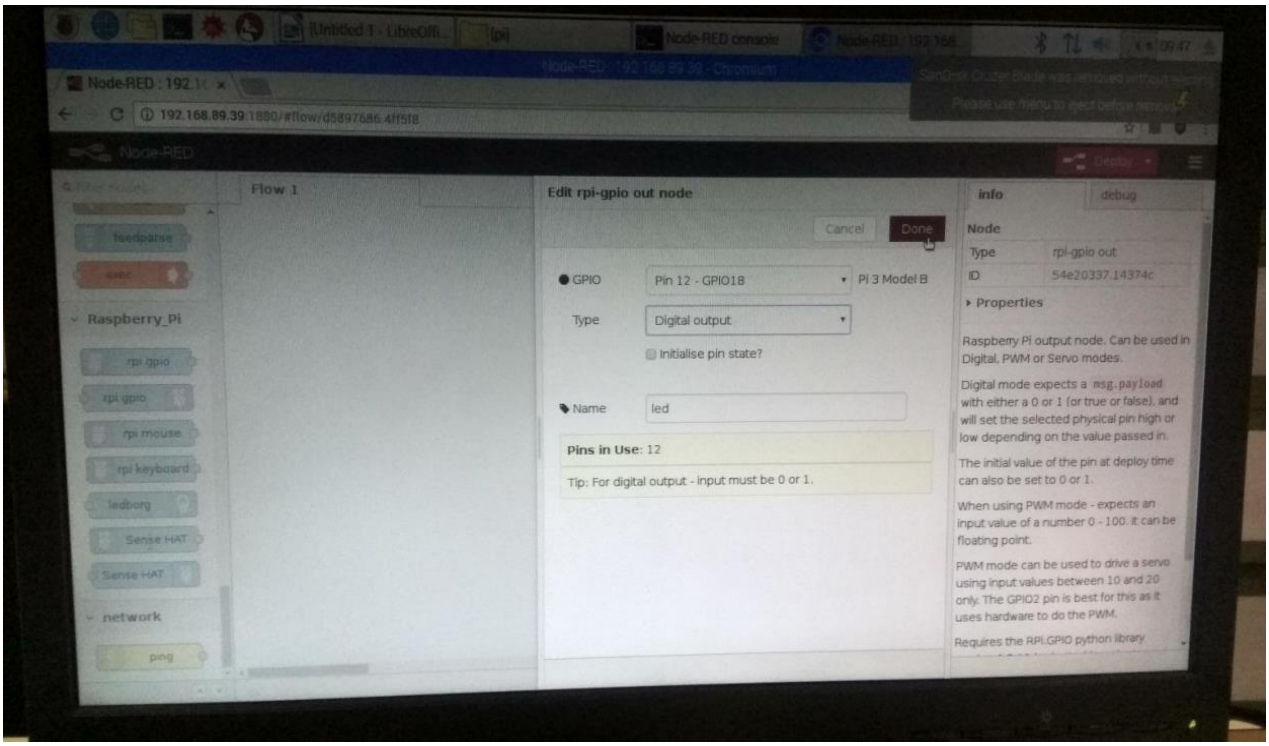
STEP 7:



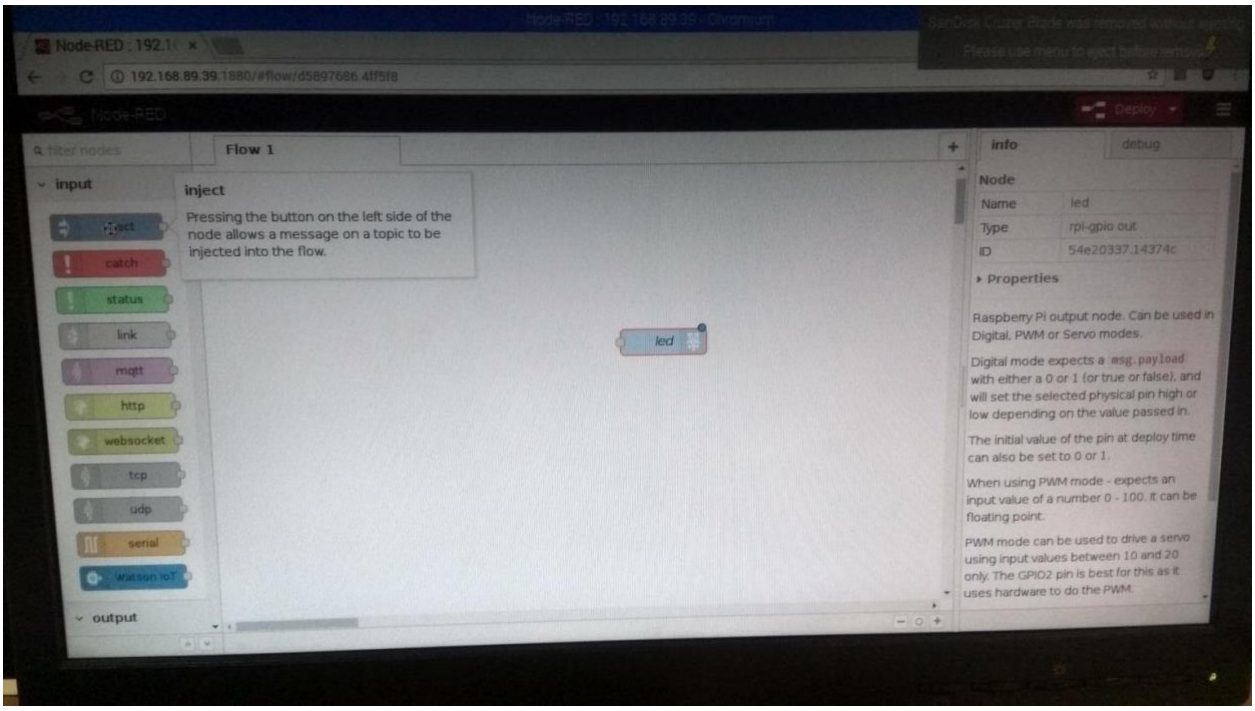
STEP 104:



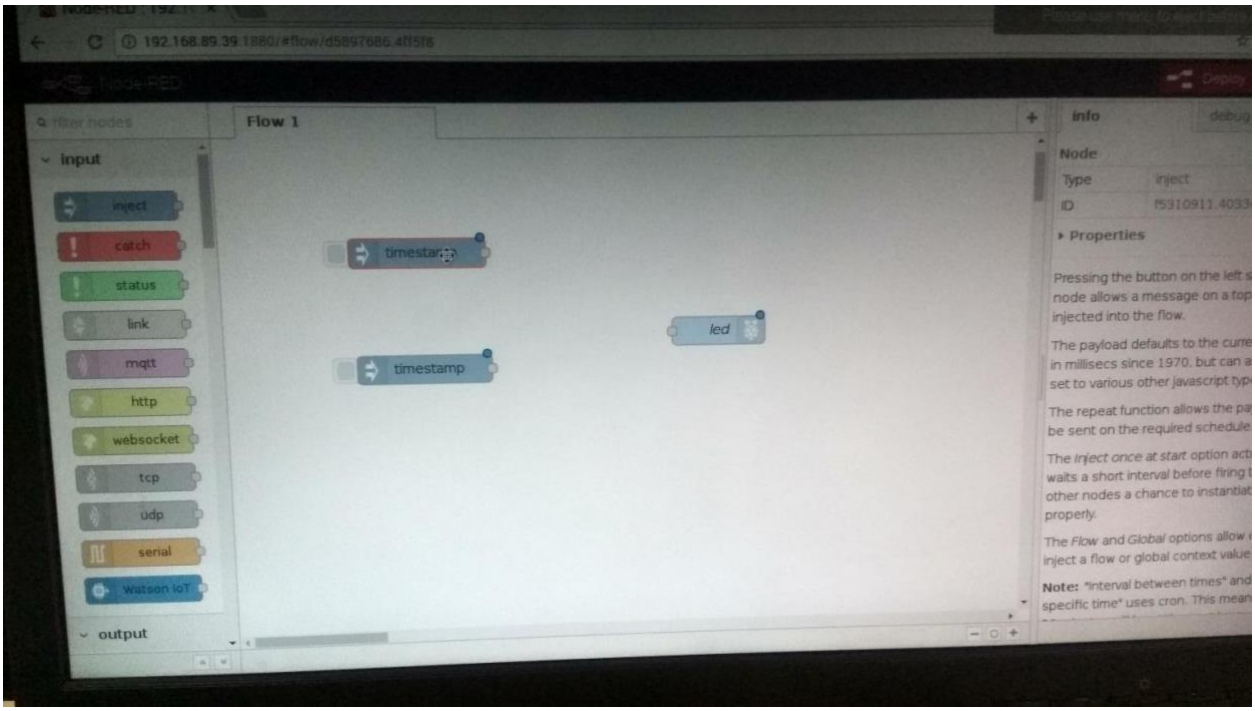
STEP 105:



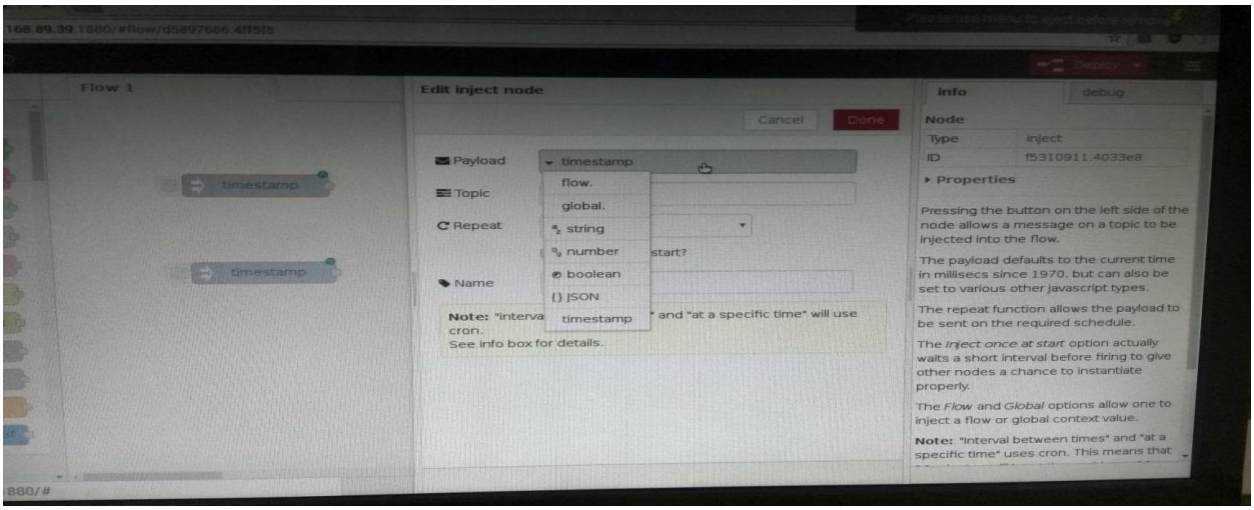
STEP 105:


Node-RED : 192.1
192.168.89.39:1880/#flow/d5897686.4ff5f8
Node-RED
Deploy
filter nodes
Flow 1
info
debug
input
inject
Pressing the button on the left side of the node allows a message on a topic to be injected into the flow.
catch
status
link
mqtt
http
websocket
tcp
udp
serial
watson IoT
output
led
Node
Name
led
Type
rpi-gpio out
ID
54e20337.14374c
Properties
Raspberry Pi output node. Can be used in Digital, PWM or Servo modes.
Digital mode expects a msg.payload with either a 0 or 1 (or true or false), and will set the selected physical pin high or low depending on the value passed in.
The initial value of the pin at deploy time can also be set to 0 or 1.
When using PWM mode - expects an input value of a number 0 - 100. It can be floating point.
PWM mode can be used to drive a servo using input values between 10 and 20 only. The GPIO2 pin is best for this as it uses hardware to do the PWM.

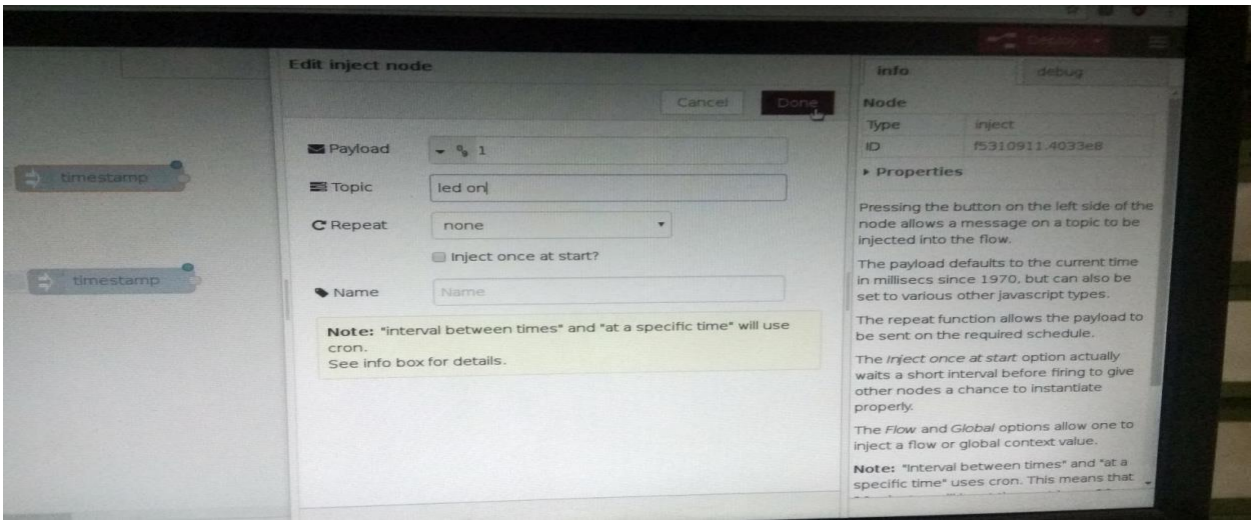
STEP 11:


192.168.89.39:1880/#flow/d5897686.4ff5f8
Node-RED
Deploy
filter nodes
Flow 1
info
input
inject
catch
status
link
mqtt
http
websocket
tcp
udp
serial
watson IoT
output
timestamp
led
timestamp
Node
Type
inject
Properties
Pressing the button on the left s
node allows a message on a top
injected into the flow.
The payload defaults to the curre
in millisecs since 1970. but can a
set to various other javascript typ
The repeat function allows the pa
be sent on the required schedule
The inject once at start option act
waits a short interval before firing t
other nodes a chance to instantiat
properly.
The Flow and Global options allow
inject a flow or global context value
Note: "interval between times" and
specific time" uses cron. This mean

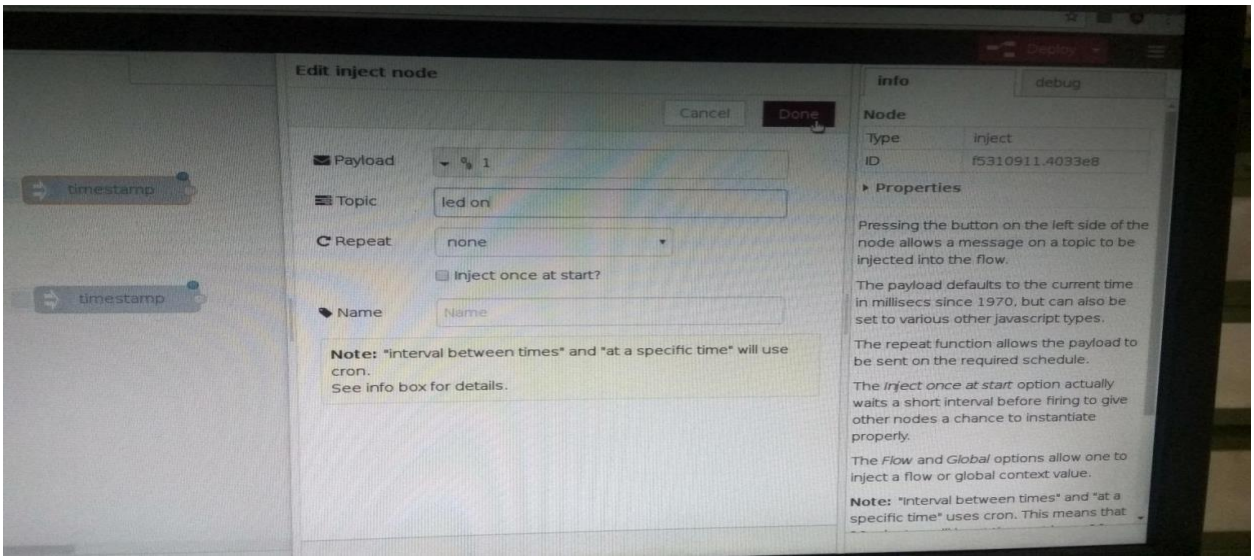
STEP 12:



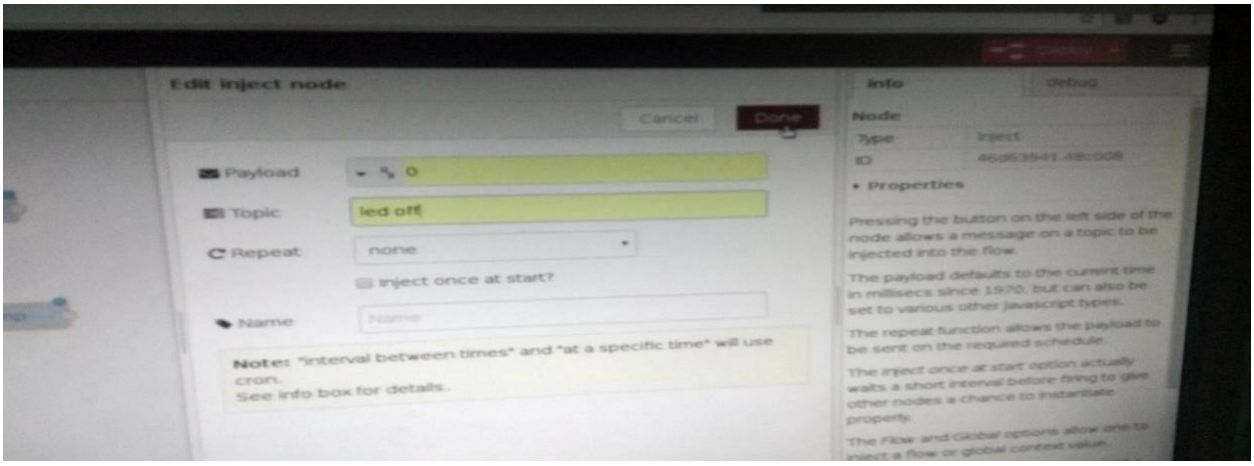
STEP 13:



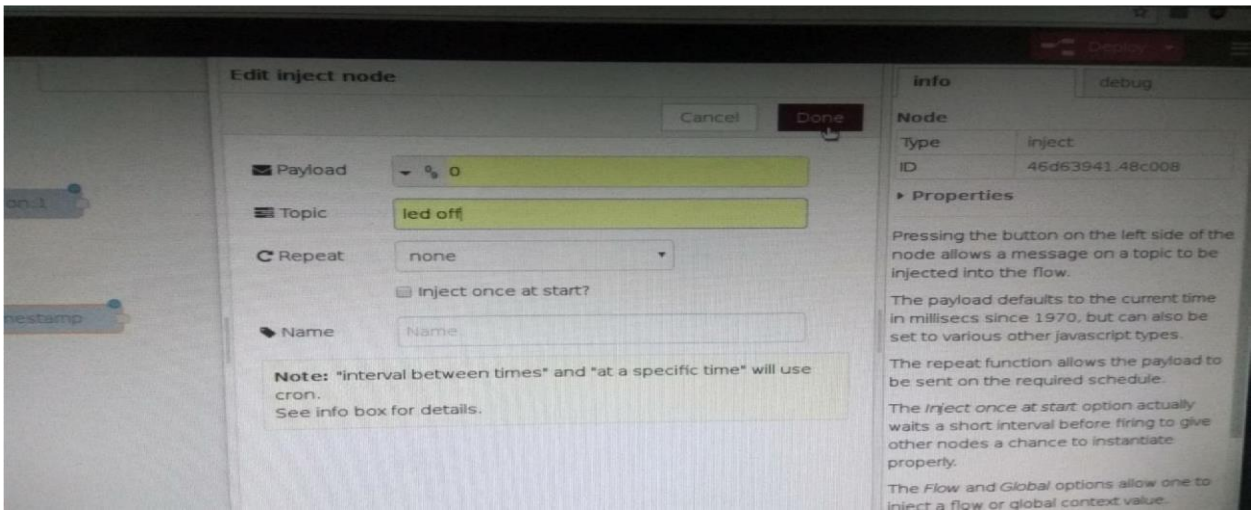
STEP 14:



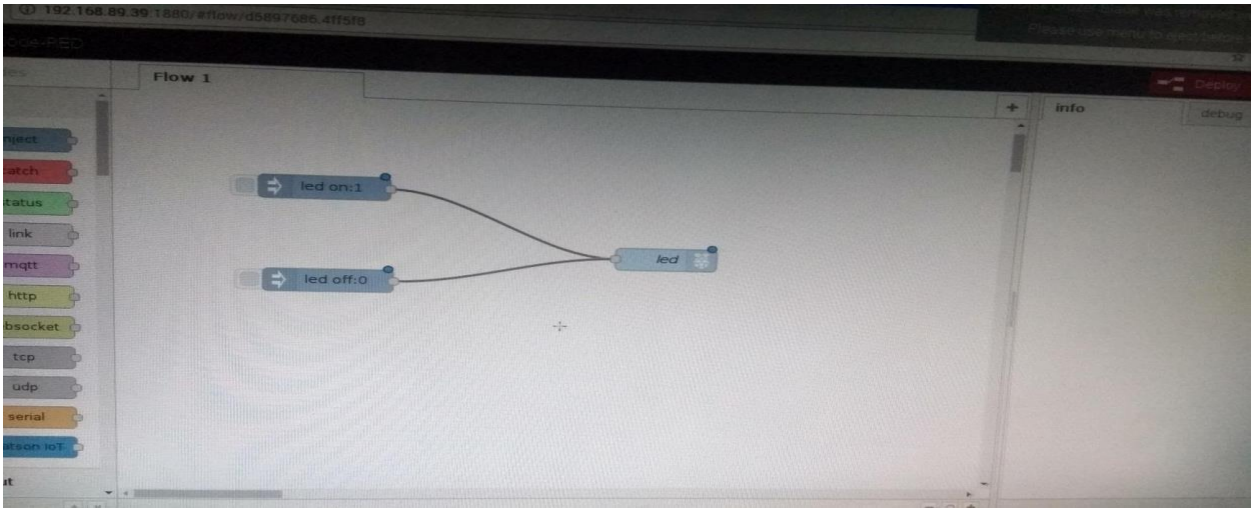
STEP 15:



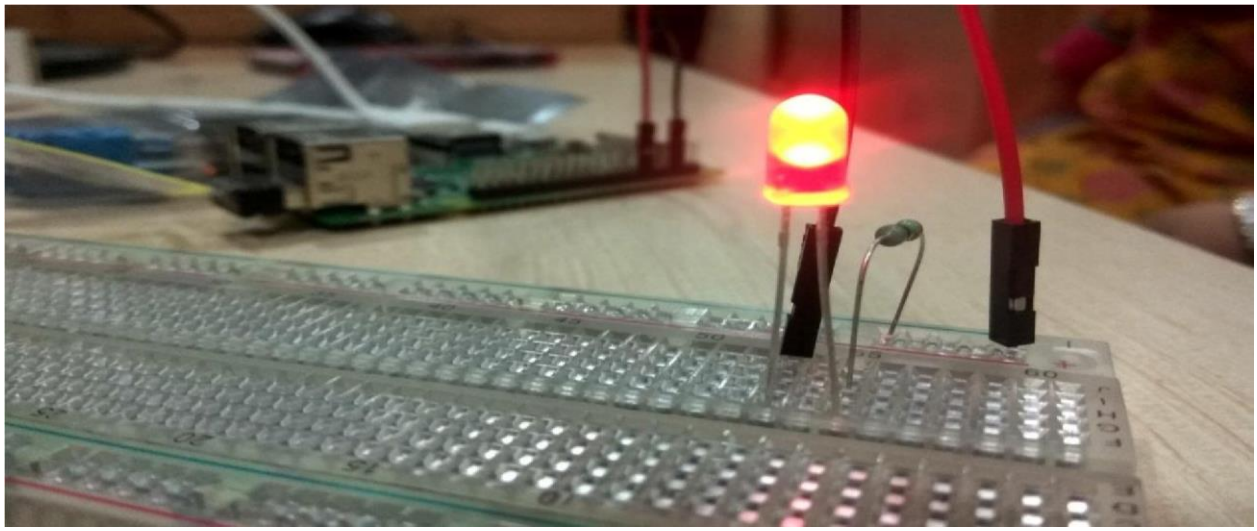
STEP 16:

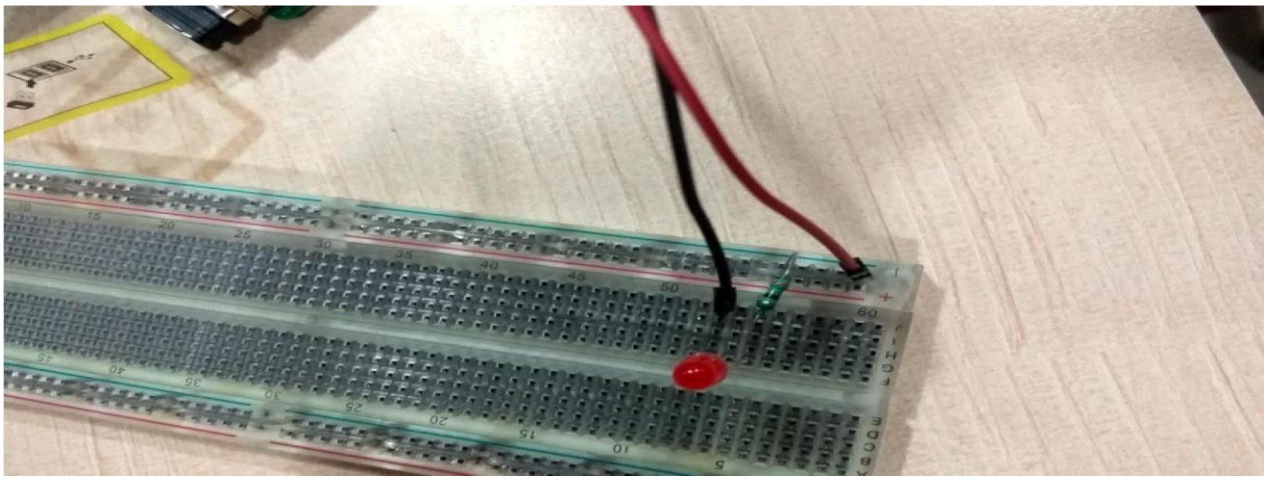


STEP 17:



HARDWARE CONNECTIONS:





Practical No: 09

GPIO: LED Grid Module:

We have created a series of Raspberry Pi Tutorials, in which we have covered Interfacing of Raspberry Pi with all the basic components like LED, LCD, button, DC motor, Servo Motor, Stepper Motor, ADC, shift Register, etc. We have also published some **simple Raspberry Pi projects** for beginners, along with some good IoT projects. Today, in continuation of these tutorials, we are going to **Control 8x8 LED Matrix Module by Raspberry Pi**. We will write a python program to show characters on the matrix module.

Also check Interfacing 8x8 LED Matrix with Arduino and LED Matrix with AVR Microcontorller.

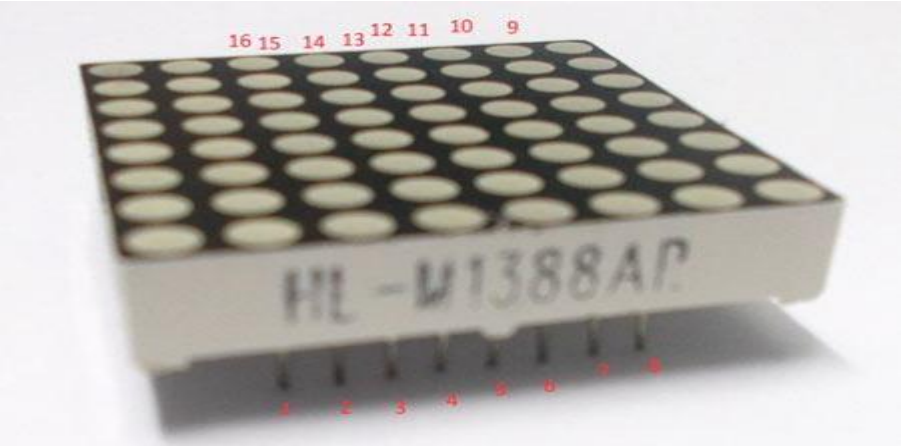
Components Required:

Here we are using **Raspberry Pi 2 Model B with Raspbian Jessie OS**. All the basic Hardware and Software requirements are previously discussed, you can look it up in the Raspberry Pi Introduction and Raspberry PI LED Blinking for getting started, other than that we need:

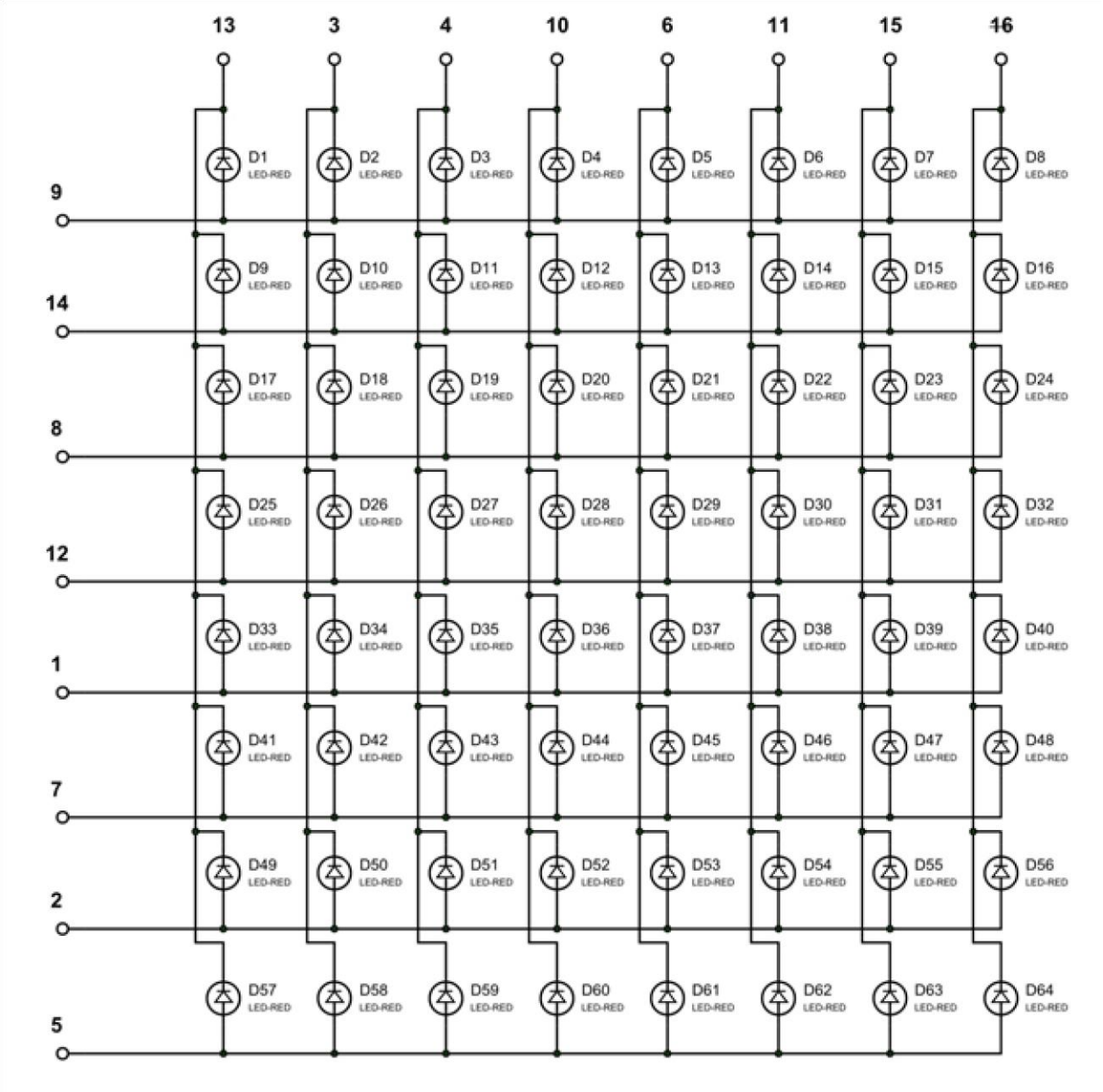
- Raspberry Pi Board
- Power supply (5v)
- 1000uF capacitor (connected across power supply)
- 1KΩ resistor (8 pieces)

8x8 LED Matrix Module:

An 8*8 LED matrix module contains 64 LED (Light Emitting Diodes) which are arranged in the form of a matrix, hence the name is LED matrix. These compact modules are available in different sizes and many colors. One can choose them on convenience. The PIN configuration of the module is as shown in picture. Keep in mind that, the pinouts of module are not in order so the PINs should be numbered exactly as shown in picture for avoiding errors.



There are 8+8=16 common terminals in the LED Matrix module. Over them, we have 8 common positive terminals and 8 common negative terminals, in the form of 8 rows and 8 columns, for connecting 64 LED in matrix form. If the module were to be drawn in the form of circuit diagram we will have a picture as shown below:



So for 8 rows, we have **8 Common Positive Terminals** (9, 14, 8, 12, 17, 2, 5). Consider the first row, the LEDs from D1 to D8 have a common positive terminal and the pin is brought out at PIN9 of LED Matrix module. When we want one or all LEDs in a ROW to be ON, The corresponding pin of LED MODULE should be powered with +3.3v.

Similar to common positive terminals, we have **8 Common Negative Terminals** as columns (13, 3, 4, 10, 6, 11, 15, 16). For grounding any LED in any column the respective common negative terminal to be grounded.

Circuit Explanation:

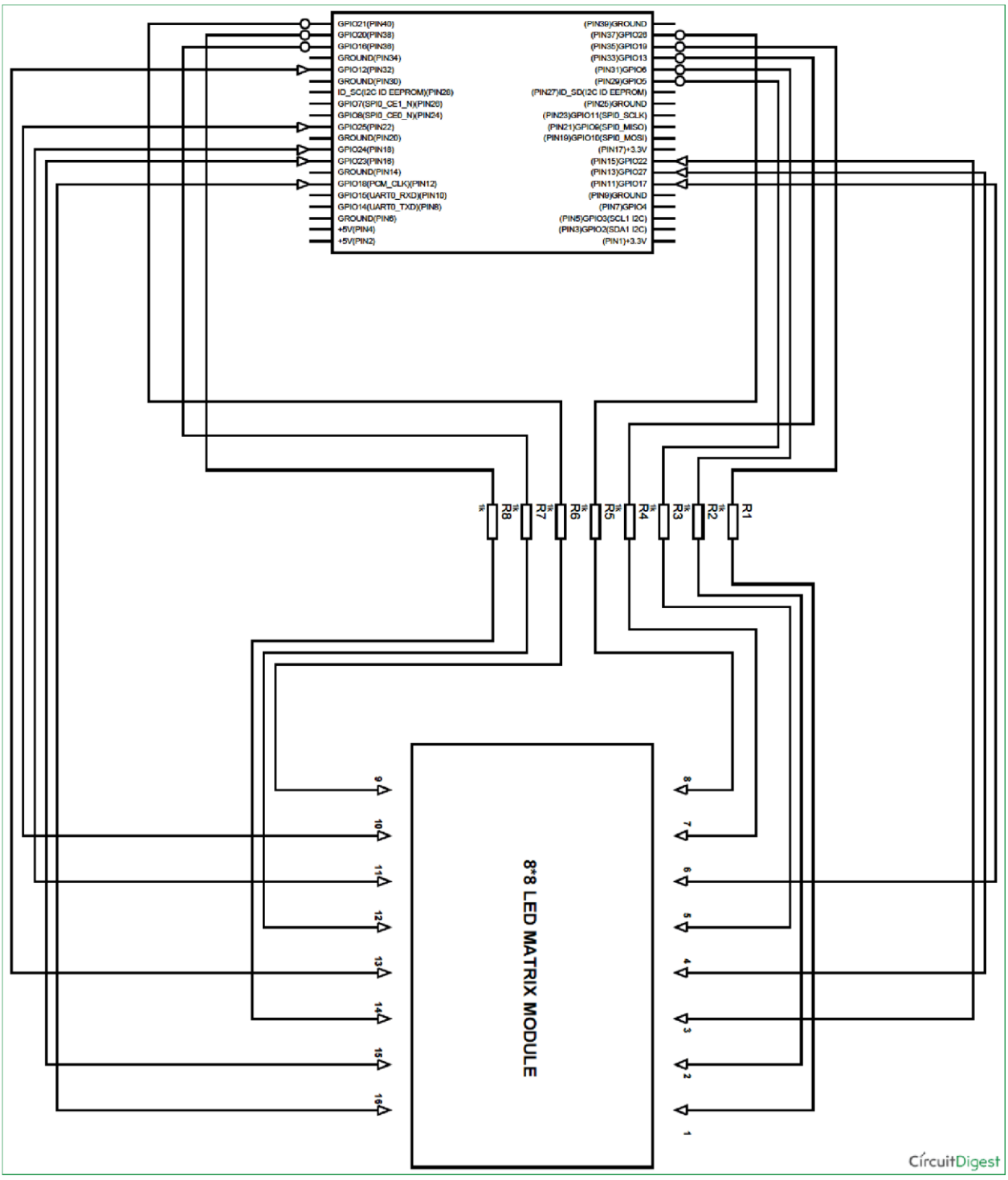
The connections which are done between **Raspberry Pi and LED matrix** module are shown in below table.

LED Matrix Module Pin no.	**Function**	**Raspberry Pi GPIO Pin No.**
13	POSITIVE0	GPIO12
3	POSITIVE1	GPIO22
4	POSITIVE2	GPIO27
10	POSITIVE3	GPIO25
6	POSITIVE4	GPIO17
11	POSITIVE5	GPIO24
15	POSITIVE6	GPIO23
16	POSITIVE7	GPIO18

1	NEGATIVE4	GPIO19
7	NEGATIVE5	GPIO13
2	NEGATIVE6	GPIO6
5	NEGATIVE7	GPIO5

Here is the Final circuit diagram for **Interfacing 8x8 LED Matrix with Raspberry Pi** :

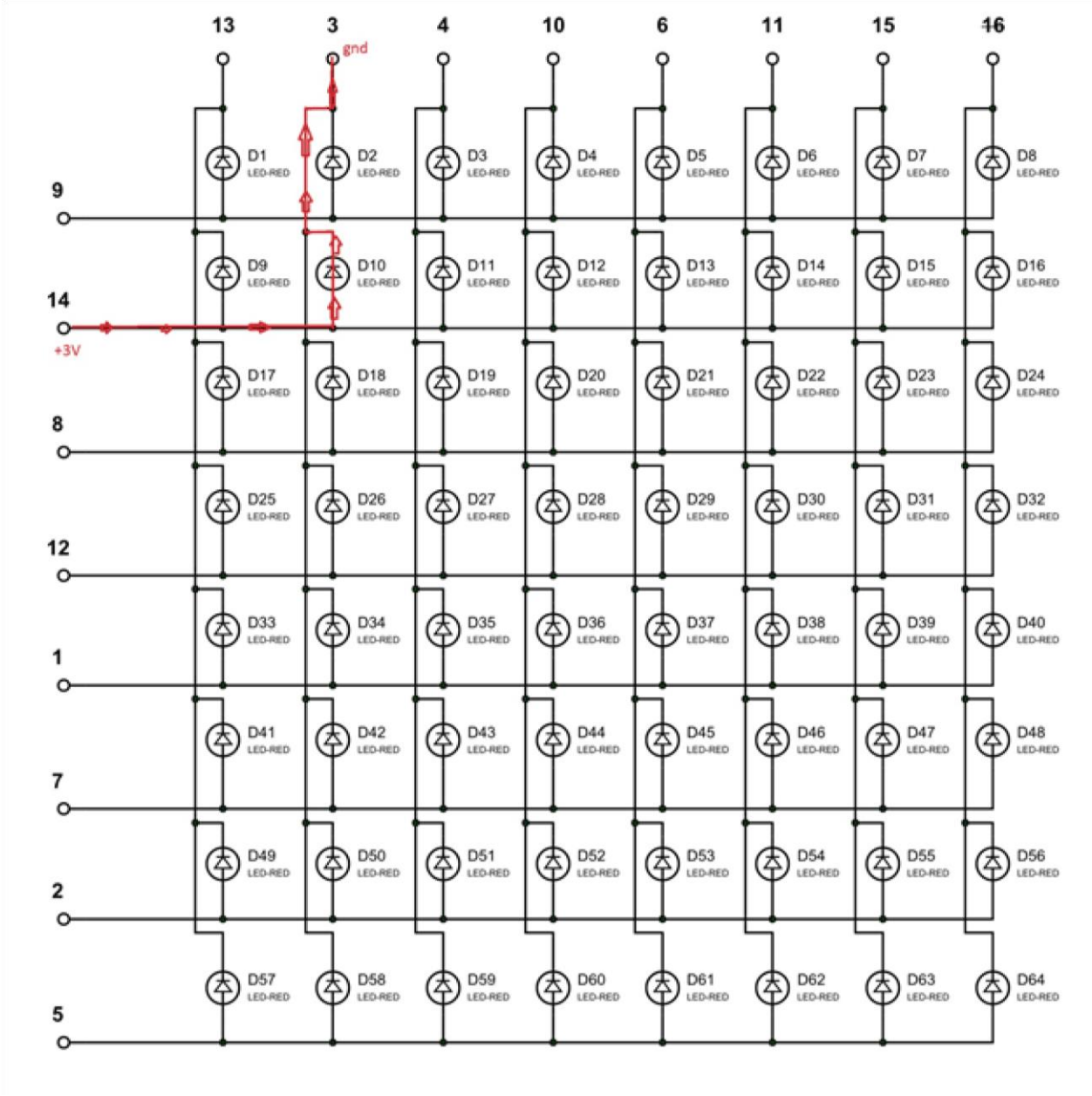
9	NEGATIVE0	GPIO21
14	NEGATIVE1	GPIO20
8	NEGATIVE2	GPIO26
12	NEGATIVE3	GPIO16



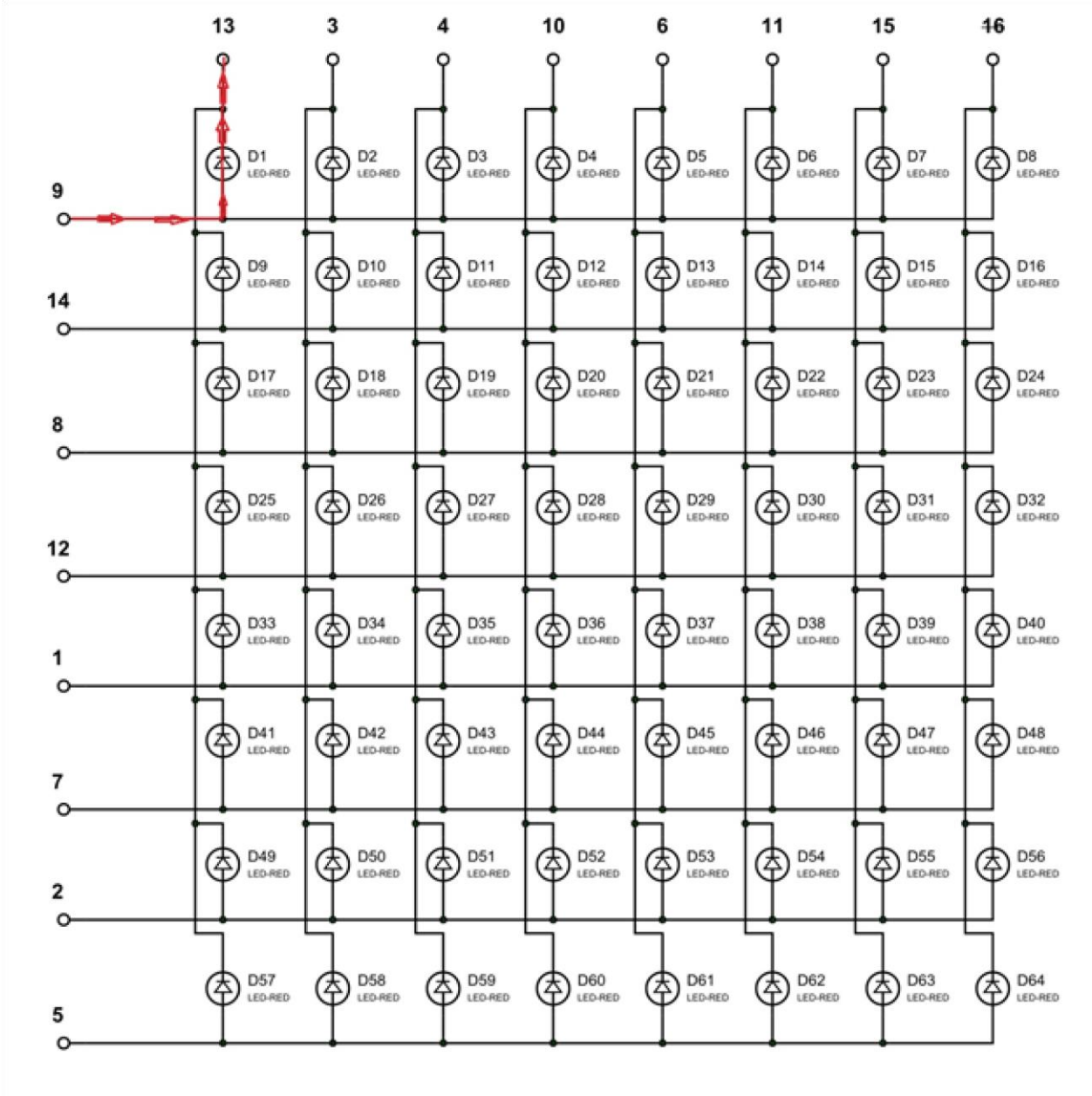
Working Explanation:

Here we will use **Multiplexing Technique** to show characters on the 8x8 LED Matrix Module. So let's discuss about this multiplexing in detail. Say if we want to turn on LED D10 in the matrix, we need to

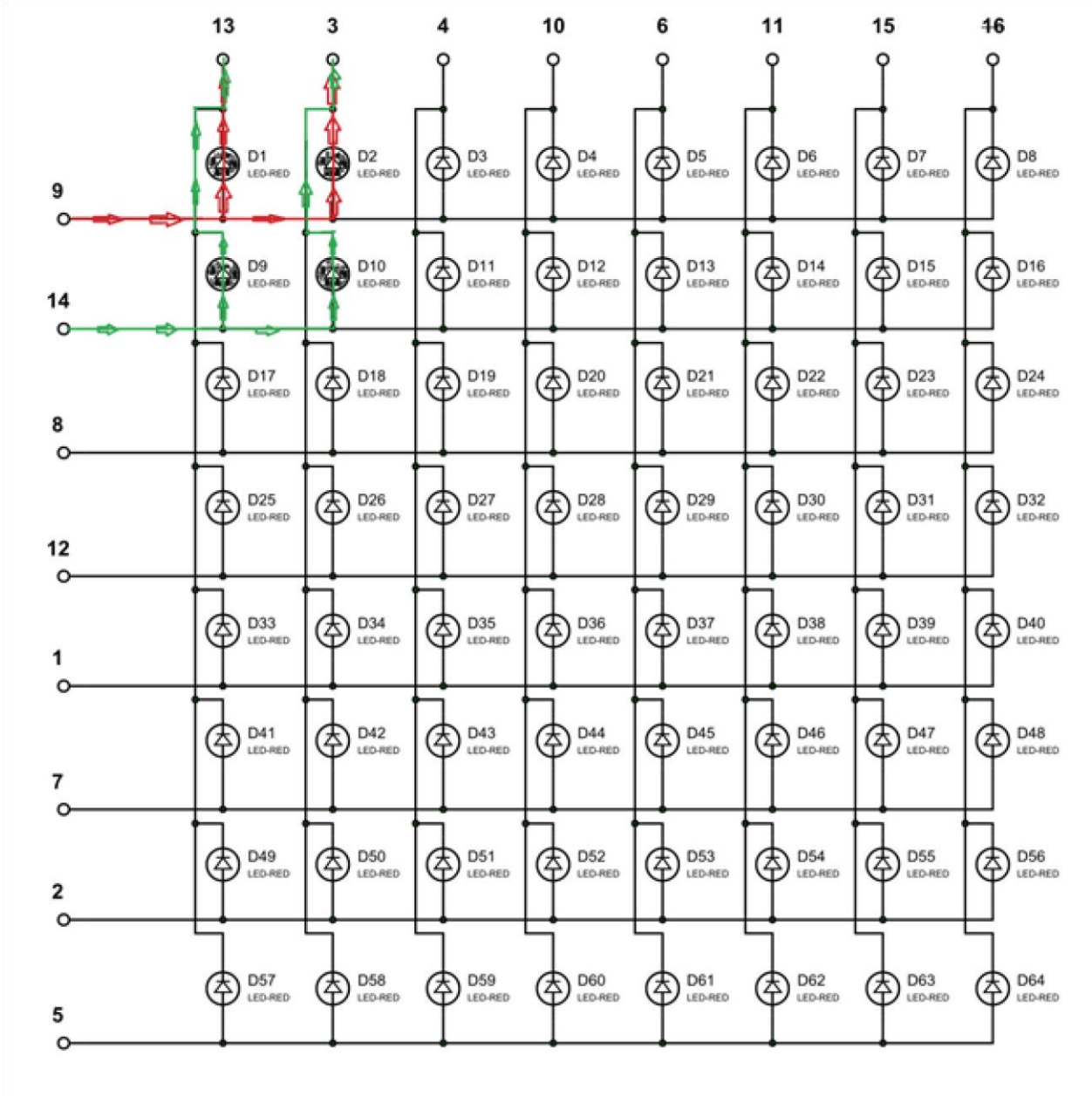
power the PIN14 of module and ground the PIN3 of module. With this LED D10 will turn ON as shown in below figure. This should also be checked first for MATRIX to know everything is in order.



Now, say if we want to turn on D1, we need to power PIN9 of matrix and ground the PIN13. With that LED D1 will glow. The current direction in this case is shown in below figure.



Now for the tricky part, consider we want to turn on both D1 and D10 at the same time. So we should power both PIN9, PIN14 and ground both PIN13, PIN3. This will turn on the LED D1 and D10, but along with that it will also turn on LED D2 and D9. It's because they share common terminals. So if we want to turn on the LEDs along the diagonal, we will be forced to turn ON all the LEDs along the way. This is shown in below figure:



To avoid this problem, we use a technique called **Multiplexing**. We have also discussed this Multiplexing Technique while interfacing 8x8 LED Matrix with AVR, here we are explaining again. This same multiplexing technique is also used in Scrolling Text on 8x8 LED matrix with Arduino and with AVR microcontroller.

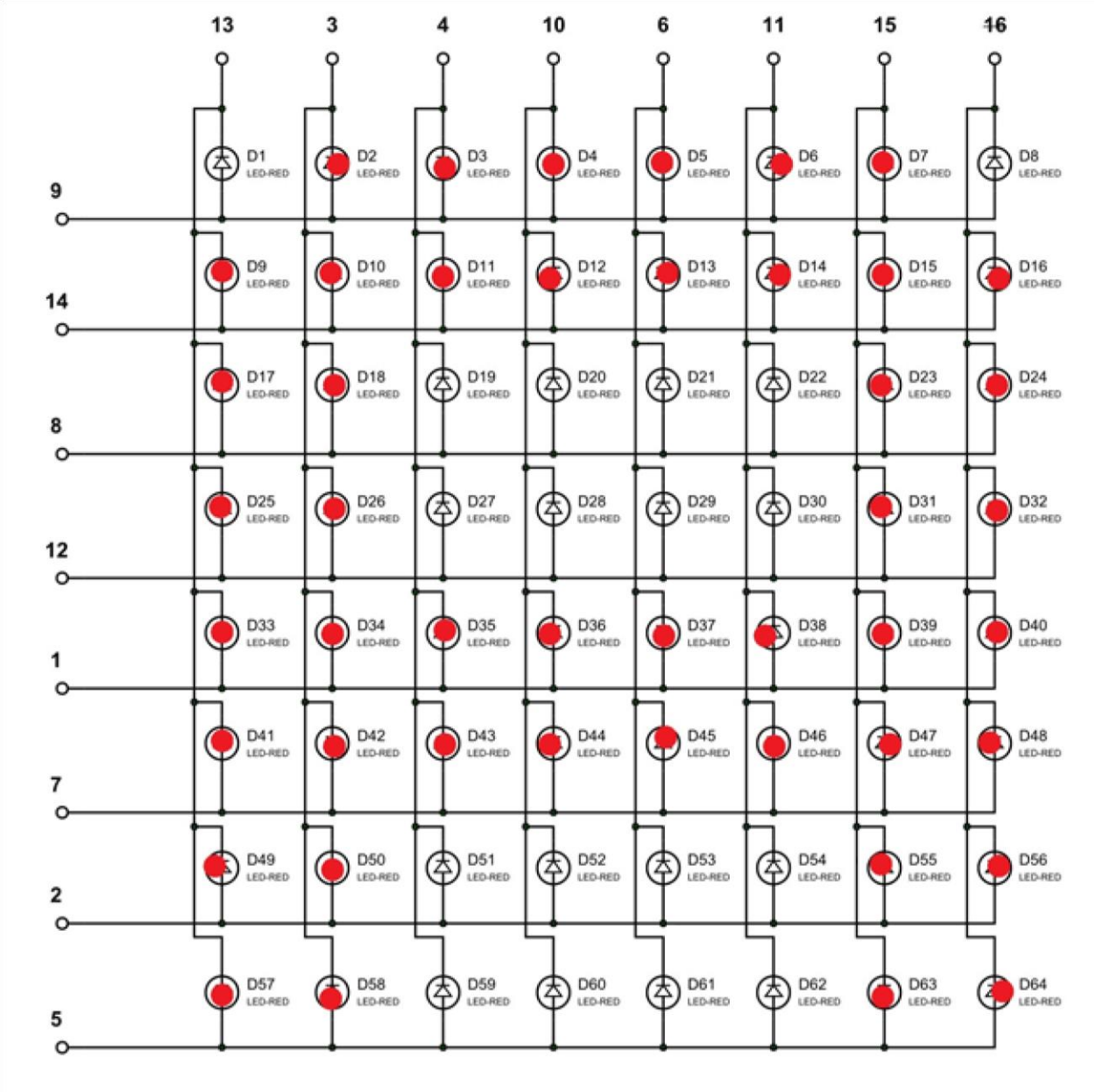
The human eye cannot capture a frequency more than 30 HZ. That is if a LED goes ON and OFF continuously at the rate of 30HZ or more. The eye sees the LED as continuously ON. However this is not the case and LED will be actually turning ON and OFF constantly. This technique is called **Multiplexing**.

Let's say for example, we want to only turn on LED D1 and LED D10 without turning on D2 and

D9. Trick is, we will first provide power to only LED D1 using PIN 9 & 13 and wait for 1mSEC,

and then we will turn it OFF. Then we will provide power to LED D10 using PIN 14 & 3 and wait for 1mSEC then will turn it OFF. The cycle goes continuously with high frequency and D1 & D10 will getting On and Off rapidly and both the LEDs will appear to be continuously ON to our eye. Means we are only providing power to the one row (LED) at a time, eliminating the chances of turning on other LEDs in other rows. We will use this technique to show all characters.

We can further understand it by one example, like if we want to display “A” on the matrix, like shown below:



As told we will turn ON one row in an instant,
At t=0m SEC, PIN09 is set HIGH (other ROW pins are LOW at this time) at this time,PIN3,PIN4,PIN10,PIN6,PIN11,PIN15 are grounded(other COLUMN pins are HIGH at this time)

At t=1m SEC, PIN14 is set HIGH (other ROW pins are LOW at this time)at this time, PIN13,PIN3,PIN4,PIN10,PIN6,PIN11,PIN15,PIN16 are grounded(other COLUMN pins are HIGH at this time)

At t=2m SEC, PIN08 is set HIGH (other ROW pins are LOW at this time)at this time, PIN13,PIN3,PIN15,PIN16 are grounded(other COLUMN pins are HIGH at this time)

At t=3m SEC, PIN12 is set HIGH (other ROW pins are LOW at this time)at this time, PIN13,PIN3,PIN15,PIN16 are grounded(other COLUMN pins are HIGH at this time)

At t=4m SEC, PIN01 is set HIGH (other ROW pins are LOW at this time)at this time, PIN13,PIN3,PIN4,PIN10,PIN6,PIN11,PIN15,PIN16 are grounded(other COLUMN pins are HIGH at this time)

At t=5m SEC, PIN07 is set HIGH (other ROW pins are LOW at this time)at this time, PIN13,PIN3,PIN4,PIN10,PIN6,PIN11,PIN15,PIN16 are grounded(other COLUMN pins are HIGH at this time)

At t=6m SEC, PIN02 is set HIGH (other ROW pins are LOW at this time)at this time, PIN13,PIN3,PIN15,PIN16 are grounded(other COLUMN pins are HIGH at this time)

At t=7m SEC, PIN05 is set HIGH (other ROW pins are LOW at this time)at this time, PIN13,PIN3,PIN15,PIN16 are grounded(other COLUMN pins are HIGH at this time)

At this speed, the display will be seen as continuously showing “A” character as shown in figure.

The **Python Program for showing Characters on LED Matrix using Raspberry Pi** is given below. Program is well explained by comments. Port Values for each character are given in the program. You can show whatever characters you want by just changing the *‘pinp’* values in the ‘for loops’ in the given program. Also check the Demo Video below.

Code

```
#working
import RPi.GPIO as IO  #calling for header file which helps in using GPIO’s of PI
import time          #calling for time to provide delays in program
IO.setwarnings(False)  #do not show any warnings x=1
y=1
```

```
IO.setmode (IO.BCM)  #programming the GPIO by BCM pin numbers. (like PIN29 as'GPIO5')
IO.setup(12,IO.OUT)  #initialize GPIO12 as an output.
IO.setup(22,IO.OUT)  #initialize GPIO22 as an output.
IO.setup(27,IO.OUT)
IO.setup(25,IO.OUT)
IO.setup(17,IO.OUT)
IO.setup(24,IO.OUT)
IO.setup(23,IO.OUT)
IO.setup(18,IO.OUT)
IO.setup(21,IO.OUT)
IO.setup(20,IO.OUT)
IO.setup(26,IO.OUT)
IO.setup(16,IO.OUT)
IO.setup(19,IO.OUT)
IO.setup(13,IO.OUT)
IO.setup(6,IO.OUT)
IO.setup(5,IO.OUT)

PORTVALUE = [128,64,32,16,8,4,2,1]
#value of pin in each port
A=[0,0b01111111,0b11111111,0b11001100,0b11001100,0b11001100,0b11111111,0b01111111]
B =[0,0b00111100,0b01111110,0b11011011,0b11011011,0b11011011,0b11111111,0b11111111]
C= [0,0b11000011,0b11000011,0b11000011,0b11000011,0b11100111,0b01111110,0b00111100]
D=[0,0b01111110,0b10111101,0b11000011,0b11000011,0b11000011,0b11111111,0b11111111]
E=[0,0b11011011,0b11011011,0b11011011,0b11011011,0b11011011,0b11111111,0b11111111]
F=[0,0b11011000,0b11011000,0b11011000,0b11011000,0b11011000,0b11111111,0b11111111]
G=[0b00011111,0b11011111,0b11011000,0b11011011,0b11011011,0b11011011,0b11111111,0b111111
11]
H=[0,0b11111111,0b11111111,0b00011000,0b00011000,0b00011000,0b11111111,0b11111111]
I=[0b11000011,0b11000011,0b11000011,0b11111111,0b11111111,0b11000011,0b11000011,0b110000
11]
```

```
11]
J=[0b11000000,0b11000000,0b11000000,0b11111111,0b11111111,0b11000011,0b11001111,0b110011
11]
K=[0,0b11000011,0b11100111,0b01111110,0b00111100,0b00011000,0b11111111,0b11111111]
L=[0b00000011,0b00000011,0b00000011,0b00000011,0b00000011,0b00000011,0b11111111,0b111111
```

```
11]
M=[0b11111111,0b11111111,0b01100000,0b01110000,0b01110000,0b01100000,0b11111111,0b11111
111]
N=[0b11111111,0b11111111,0b00011100,0b00111000,0b01110000,0b11100000,0b11111111,0b111111
```

10]
O=[0b01111110,0b11111111,0b11000011,0b11000011,0b11000011,0b11000011,0b11111111,0b011111
10]
P=[0,0b01110000,0b11111000,0b11001100,0b11001100,0b11001100,0b11111111,0b11111111]
Q=[0b01111110,0b11111111,0b11001111,0b11011111,0b11011011,0b11000011,0b11111111,0b011111

11]
R=[0b01111001,0b11111011,0b11011111,0b11011110,0b11011100,0b11011000,0b11111111,0b111111
11]
S=[0b11001110,0b11011111,0b11011011,0b11011011,0b11011011,0b11011011,0b11111011,0b011100

10]
T=[0b11000000,0b11000000,0b11000000,0b11111111,0b11111111,0b11000000,0b11000000,0b110000
00]
U=[0b11111110,0b11111111,0b00000011,0b00000011,0b00000011,0b00000011,0b11111111,0b111111

110]
V=[0b11100000,0b11111100,0b00011110,0b00000011,0b00000011,0b00011110,0b11111100,0b111000
00]
W=[0b11111110,0b11111111,0b00000011,0b11111111,0b11111111,0b00000011,0b11111111,0b11111

X=[0b01000010,0b11100111,0b01111110,0b00111100,0b00111100,0b01111110,0b11100111,0b010000
10]
Y=[0b01000000,0b11100000, 0b01110000,0b00111111,0b00111111,0b01110000,0b11100000,0b010000

```
00]
Z=[0b11000011,0b11100011,0b11110011,0b11111011,0b11011111,0b11001111,0b11000111,0b110000
11]

def PORT(pin):  #assigning GPIO state by taking 'pin' value
if(pin&0x01 == 0x01):
      IO.output(21,0)   #if bit0 of 8bit 'pin' is true pull PIN21 low     else:
      IO.output(21,1)   #if bit0 of 8bit 'pin' is false pull PIN21 high
if(pin&0x02 == 0x02):
      IO.output(20,0)   #if bit1 of 8bit 'pin' is true pull PIN20 low     else:
      IO.output(20,1)   #if bit1 of 8bit 'pin' is false pull PIN20 high
if(pin&0x04 == 0x04):
      IO.output(26,0)   #if bit2 of 8bit 'pin' is true pull PIN26 low     else:
      IO.output(26,1)   #if bit2 of 8bit 'pin' is false pull PIN26 high
if(pin&0x08 == 0x08):       IO.output(16,0)     else:
      IO.output(16,1)

if(pin&0x10 == 0x10):
IO.output(19,0)     else:
      IO.output(19,1)
if(pin&0x20 == 0x20):
IO.output(13,0)     else:
      IO.output(13,1)
if(pin&0x40 == 0x40):
IO.output(6,0)     else:
      IO.output(6,1)
if(pin&0x80 == 0x80):
IO.output(5,0)     else:
      IO.output(5,1)

def PORTP(pinp):    #assigning GPIO logic for positive terminals by
taking 'pinp' value     if(pinp&0x01 == 0x01):
      IO.output(12,1)     #if bit0 of 8bit 'pinp' is true pull PIN12 high
else:
      IO.output(12,0)       #if bit0 of 8bit 'pinp' is false pull PIN12 low
if(pinp&0x02 == 0x02):
      IO.output(22,1)     #if bit1 of 8bit 'pinp' is true pull PIN22 high
else:
      IO.output(22,0)       #if bit1 of 8bit 'pinp' is false pull PIN22 low
if(pinp&0x04 == 0x04):
      IO.output(27,1)     #if bit2 of 8bit 'pinp' is true pull PIN27 high
else:
      IO.output(27,0)     #if bit2 of 8bit 'pinp' is false pull PIN27 low
   if(pinp&0x08 == 0x08):       IO.output(25,1)
```

```
  else:       IO.output(25,0)
if(pinp&0x10 == 0x10):
for y in range (100):      for
x in range (8):
      IO.output(17,1)
else:
      IO.output(17,0)
if(pinp&0x20 == 0x20):
IO.output(24,1)    else:
      IO.output(24,0)
if(pinp&0x40 == 0x40):
IO.output(23,1)    else:
      IO.output(23,0)    if(pinp&0x80
== 0x80):
      IO.output(18,1) #if bit7 of 8bit 'pinp' is true pull PIN18 high
else:
      IO.output(18,0) #if bit7 of 8bit 'pinp' is false pull PIN18 low

while 1:    for y in range (100):   #execute loop 100
times
      for x in range (8): #execute the loop 8 times incrementing x
value from zero to seven           pin  = PORTVALUE[x]  #assigning
value to 'pin' for each digit           PORT(pin);  #mapping
appropriate GPIO
         pinp=  C[x]    #assigning  character  'C'  value  to  'pinp'
PORTP(pinp);  #turning  the  GPIO  to  show  character  'C'
time.sleep(0.0005) #wait for 0.5msec

   for y in range (100):
for x in range (8):
         pin  = PORTVALUE[x]
         PORT(pin);
pinp= I[x]
         PORTP(pinp);
         time.sleep(0.0005)

for y in range (100):
for x in range (8):
         pin  = PORTVALUE[x]
         PORT(pin);
pinp= R[x]
PORTP(pinp);
time.sleep(0.0005)

   for y in range (100):
for x in range (8):
```

```
        pin  = PORTVALUE[x]
        PORT(pin);
pinp= C[x]
PORTP(pinp);
time.sleep(0.0005)
       pin     =   PORTVALUE[x]
PORT(pin);          pinp= U[x]
        PORTP(pinp);
time.sleep(0.0005)

   for y in range (100):
for x in range (8):
        pin  = PORTVALUE[x]
        PORT(pin);
pinp= I[x]
        PORTP(pinp);
time.sleep(0.0005)

   for y in range (100):
for x in range (8):
        pin  = PORTVALUE[x]
        PORT(pin);
pinp= T[x]
PORTP(pinp);
time.sleep(0.0005)

   for y in range (100):
for x in range (8):
        pin  = PORTVALUE[x]
        PORT(pin);
pinp= D[x]
PORTP(pinp);
        time.sleep(0.0005)

   for y in range (100):
for x in range (8):
        pin  = PORTVALUE[x]
        PORT(pin);
pinp= I[x]
        PORTP(pinp);
time.sleep(0.0005)

   for y in range (100):
for x in range (8):
        pin  = PORTVALUE[x]
```

```
        PORT(pin);
pinp= G[x]
PORTP(pinp);
        time.sleep(0.0005)

   for y in range (100):
for x in range (8):
        pin  = PORTVALUE[x]
        PORT(pin);
pinp= E[x]
        PORTP(pinp);
        time.sleep(0.0005)
```

```
  for y in range (100):        for x
in  range  (8):                  pin   =
PORTVALUE[x]
         PORT(pin);           pinp=
S[x]           PORTP(pinp);
         time.sleep(0.0005)


   for y in range (100):
for x in range (8):
         pin  = PORTVALUE[x]
         PORT(pin);
pinp= T[x]
PORTP(pinp);
time.sleep(0.0005)

   pinp= 0     PORTP(pinp);
time.sleep(1)
```