University of Global Village (UGV), Barishal

Dept. of Electrical and Electronic Engineering (EEE)

Lab Manual

IOT Project Establishment

Noor Md Shahriar
BSc in EEE, RUET
Senior Lecturer
Co-chairman, Dept. of EEE
University of Global Village (UGV)
874/322, C&B Road, Barishal, Bangladesh.
Contact: +8801743500587
Facebook | LinkedIn | Twitter

INDEX

Contents

Course Title:	IOT Project Establishment	Total Class Hour	37
Course Code:		Total Practice Hour	37
Supervised by	Noor Md Shahriar	Total Hour	85

Course Rationale

The "IoT Project Establishment" course provides students with essential skills and knowledge to navigate the rapidly growing IoT field. It bridges theoretical concepts with hands-on experience, focusing on integrating hardware and software for real-world applications. This course prepares students to design, implement, and manage IoT projects, addressing the increasing demand for IoT solutions across industries. It also fosters innovation and problem-solving, enabling students to contribute to technological advancements in IoT-driven fields.

Course Objective

- To introduce the terminology, technology and its applications.
- To introduce the concept of M2M (machine to machine) with necessary protocols.
- To introduce the Python Scripting Language which is used in many IoT devices.
- To introduce the Raspberry PI platform, that is widely used in IoT applications.
- To introduce the implementation of web-based services on IoT devices.

Course Outcomes

CLO1	Understand the design, characteristics and technologies of Internet of Things
CLO2	Interpret the impact and challenges posed by IoT networks leading to new architectural models.
CLO3	Compare and contrast the deployment of smart objects and the technologies to connect them to network.
CLO4	Appraise the role of IoT protocols for efficient network communication
CLO5	Elaborate the need for Data Analytics and Security in IoT.
CLO6	Illustrate different sensor technologies for sensing real world entities and identify the applications of IoT in Industry

Assessment Pattern

- Continuous Assessment

Bloom's Category	**Tests**
Imitation	12
Manipulation	8
Precision	6
Articulation	2
Naturalization	2

- Semester End Examination: (SEE):

Bloom's Category Marks (out of 30)	Tests (20)	Quiz (10)	External Participation in Curricular/Co-Curricular Activities (20)
Imitation	06	06	Bloom's Affective Domain: (Attitude or will) • **Attendance: 10** • **Viva-Voca: 5** • **Report Submission: 5**
Manipulation	04	04	
Precision	06		
Articulation	02		
Naturalization	02		

Course Outline

Sl. No.	Topic & Details	Class Hours	CLO Mapping
1	**Functional Testing of Devices and Exporting Display:** Flashing OS, ensuring stability, using SSH & X11.	4	CLO 1, CLO 2
2	**GPIO Programming and Basic Interfacing:** Testing LEDs, switches, and Chronos eZ430 functionality.	5	CLO 2, CLO 3
3	**Light-Intensity-Based Control and Voltage Monitoring:** Automating LEDs and battery level indicators.	4	CLO 3, CLO 4
4	**Dice Game Simulation and RSS News Feed Display:** LCD interfacing and dynamic data handling.	4	CLO 4, CLO 5
5	**Porting OpenWRT and Hosting Websites:** Configuring networks and creating accessible websites.	5	CLO 5, CLO 6
6	**Webcam Server and FM Transmission:** Creating IP webcams and FM audio transmitters.	5	CLO 5, CLO 6
7	**Capstone Project and Review:** Integrating multiple IoT concepts into practical applications.	5	CLO 6, CLO 7

Course Schedule

Week	Experiment Title	Teaching-Learning Strategy	Assessment Strategy	Class Hours
1	**Functional Testing of Devices**	Demonstration and hands-on OS flashing and setup	Observation of stable OS functionality	3
2	**Exporting Display onto Other Systems**	Guided tutorial using SSH client and X11 server; hands-on exercise	Submission of working export setup screenshots	3
3	**GPIO Programming**	Hands-on session on GPIO programming; live troubleshooting	Evaluation of LED/Switch interfacing functionality	3
4	**Interfacing Chronos eZ430**	Demonstration of Chronos features; practical interfacing exercises	Practical functionality test of interfaced devices	3
5	**ON/OFF Control Based on Light Intensity**	Problem-solving session using light sensors; peer collaboration	Functionality check of LED control logic	3
6	**Battery Voltage Range Indicator**	Instructor-led coding and circuit building; Q&A session	Submission of working range indicator with voltage levels	3

7	**Dice Game Simulation**	Collaborative programming activity; demonstration of LCD interfacing	Code and functionality evaluation	3
8	**Displaying RSS News Feed on LCD**	Guided programming in Python; live internet data fetching and display	Submission of a functional RSS feed display	3
9	**Porting OpenWrt to the Device**	Demonstration of OpenWrt setup; group activity for Wi-Fi and access point testing	Live testing of Wi-Fi functionality and access point setup	3
10	**Hosting a Website on the Device**	Step-by-step tutorial on server setup and website hosting; peer feedback	Functional website hosting verification	3
11	**Webcam Server**	Hands-on USB webcam interfacing; testing IP webcam setup	Demonstration of IP webcam functionality	3
12	**FM Transmission**	Instructor-led setup of FM transmitter; frequency adjustment exercises	Functional testing of FM transmission	3
13	**Project Proposal Discussion**	Instructor feedback on project ideas; brainstorming and Q&A	Submission of project proposals	3
14	**Project Development Phase 1**	Guided project development; problem-solving support	Weekly progress report submission	3
15	**Project Development Phase 2**	Peer evaluation of intermediate projects; instructor guidance	Intermediate functionality check	3
16	**Final Project Demonstration**	Presentation of completed projects to peers and instructor	Assessment of project functionality and innovation	3
17	**Course Wrap-Up and Feedback**	Review session; instructor-led discussion on challenges and learnings	Submission of final project report	3

References

1. **Books and Manuals:**
   - *Internet of Things: Principles and Paradigms* by Rajkumar Buyya and Amir Vahid Dastjerdi.
   - *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things* by David Hanes et al.

2. **Web Resources:**
   - Raspberry Pi Official Documentation: https://www.raspberrypi.org/documentation/
   - Arduino Reference Page: https://www.arduino.cc/reference/en/

3. **Software Tools:**
   - Python for GPIO Programming: https://www.python.org/
   - OpenWRT Documentation: https://openwrt.org/docs/start

4. **Online Tutorials:**
   - Adafruit Learning System: https://learn.adafruit.com/
   - SparkFun IoT Projects: https://www.sparkfun.com/

Experiments

1. **Functional Testing of Devices**
Flashing the OS on to the device into a stable functional state by porting desktop environment with necessary packages.

2. **Exporting Display on To Other Systems**
Making use of available laptop/desktop displays as a display for the device using SSH client & X11 display server.

3. **GPIO Programming**
Programming of available GPIO pins of the corresponding device using native programming language. Interfacing of I/O devices like LED/Switch etc., and testing the functionality.

4. **Interfacing Chronos eZ430**
Chronos device is a programmable texas instruments watch which can be used for multiple purposes like PPT control, Mouse operations etc., Exploit the features of the device by interfacing with devices.

5. **ON/OFF Control Based on Light Intensity**
Using the light sensors, monitor the surrounding light intensity & automatically turn ON/OFF the high intensity LED's by taking some pre-defined threshold light intensity value.

6. **Battery Voltage Range Indicator**
Monitor the voltage level of the battery and indicating the same using multiple LED's (for ex: for 3V battery and 3 led's, turn on 3 led's for 2-3V, 2 led's for 1-2V, 1 led for 0.1-1V & turn off all for 0V)

7. **Dice Game Simulation**
Instead of using the conventional dice, generate a random value similar to dice value and display the same using a 16X2 LCD. A possible extension could be to provide the user with option of selecting single or double dice game.

8. **Displaying RSS News Feed on Display Interface**
Displaying the RSS news feed headlines on a LCD display connected to device. This can be adapted to other websites like twitter or other information websites. Python can be used to acquire data from the internet.

9. **Porting Openwrt To the Device**
Attempt to use the device while connecting to a wifi network using a USB dongle and at the same time providing a wireless access point to the dongle.

10. **Hosting a website on Board**

Building and hosting a simple website(static/dynamic) on the device and make it accessible online. There is a need to install server (eg: Apache) and thereby host the website.

11. **Webcam Server**

Interfacing the regular usb webcam with the device and turn it into fully functional IP webcam & test the functionality.

12. **FM Transmission**

Transforming the device into a regular fm transmitter capable of transmitting audio at desired frequency (generally 88-108 Mhz)

Experiment - 1

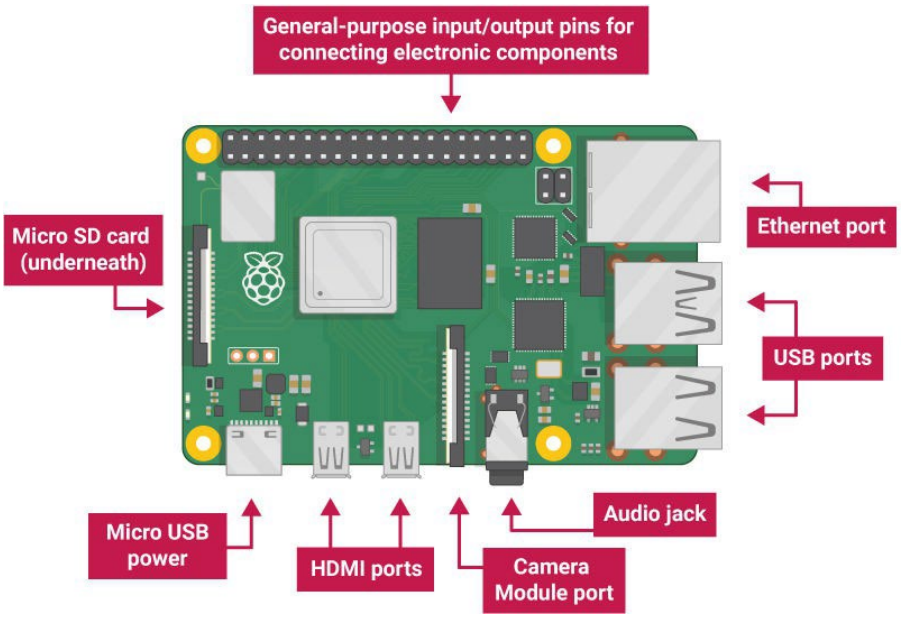
Functional Testing of Devices

Flashing the OS on to the device into a stable functional state by porting desktop environment with necessary packages.

Raspberry Pi

You are going to take a first look at Raspberry Pi! You should have a Raspberry Pi computer in front of you for this. The computer shouldn't be connected to anything yet.

- Look at your Raspberry Pi. Can you find all the things labelled on the diagram?



- USB ports — these are used to connect a mouse and keyboard. You can also connect other components, such as a USB drive.
- SD card slot — you can slot the SD card in here. This is where the operating system software and your files are stored.
- Ethernet port — this is used to connect Raspberry Pi to a network with a cable. Raspberry Pi can also connect to a network via wireless LAN.
- Audio jack — you can connect headphones or speakers here.

- HDMI port — this is where you connect the monitor (or projector) that you are using to display the output from the Raspberry Pi. If your monitor has speakers, you can also use them to hear sound.
- Micro USB power connector — this is where you connect a power supply. You should always do this last, after you have connected all your other components.
- GPIO ports — these allow you to connect electronic components such as LEDs and buttons to Raspberry Pi.

Set up your SD card

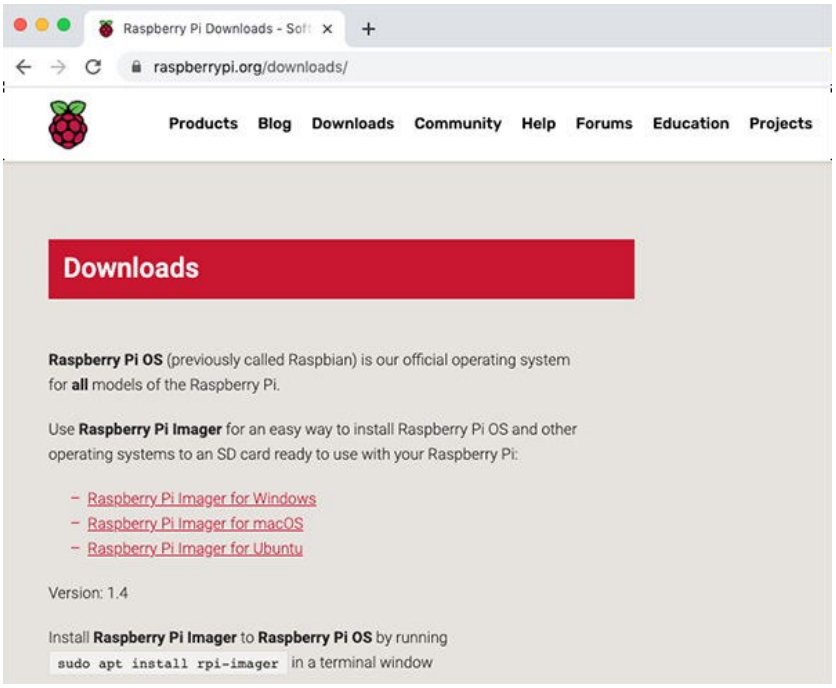
If you have an SD card that doesn't have the Raspberry Pi OS operating system on it yet, or if you want to reset your Raspberry Pi, you can easily install Raspberry Pi OS yourself. To do so, you need a computer that has an SD card port — most laptop and desktop computers have one.

The Raspberry Pi OS operating system via the Raspberry Pi Imager

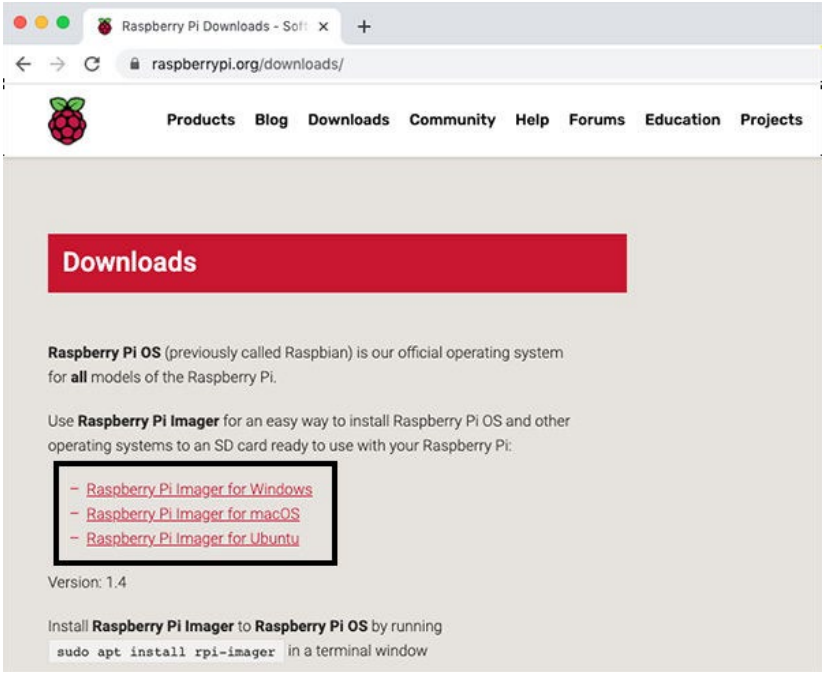
Using the Raspberry Pi Imager is the easiest way to install Raspberry Pi OS on your SD card. Note: More advanced users looking to install a particular operating system should use this guide to installing operating system images.

Download and launch the Raspberry Pi Imager

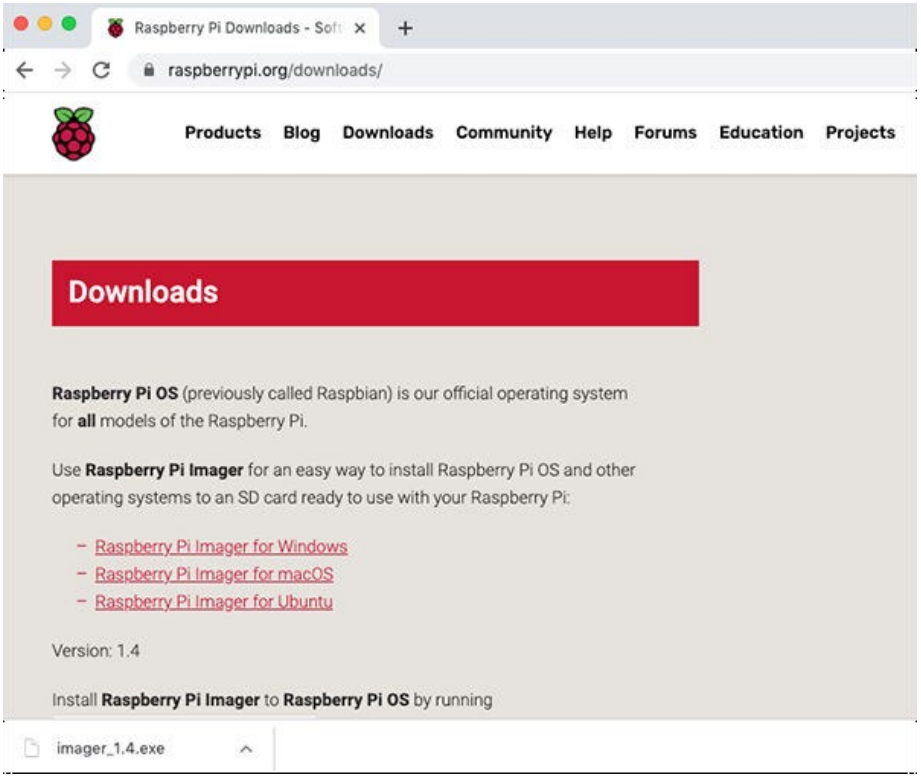
- Visit the Raspberry Pi downloads page



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



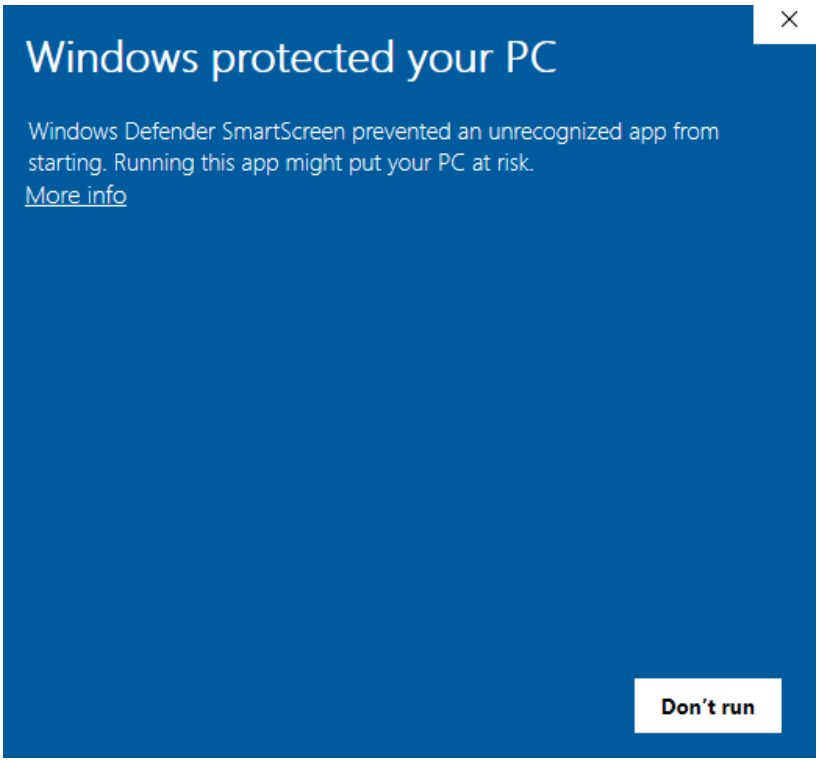
- When the download finishes, click it to launch the installer



Using the Raspberry Pi Imager

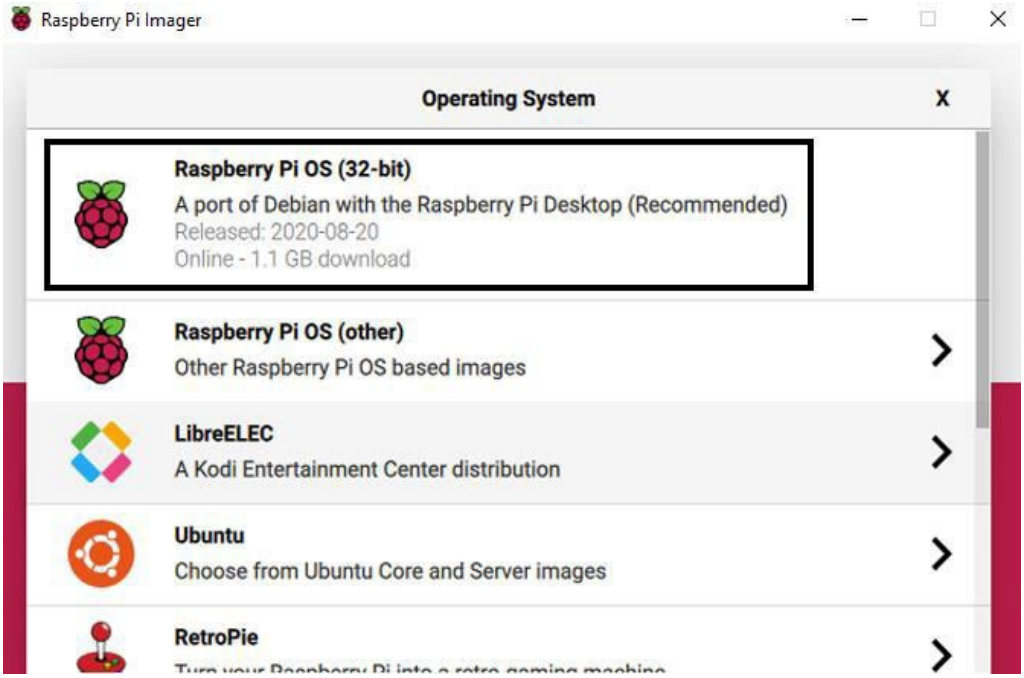
Anything that's stored on the SD card will be overwritten during formatting. If your SD card currently has any files on it, e.g. from an older version of Raspberry Pi OS, you may wish to back up these files first to prevent you from permanently losing them.

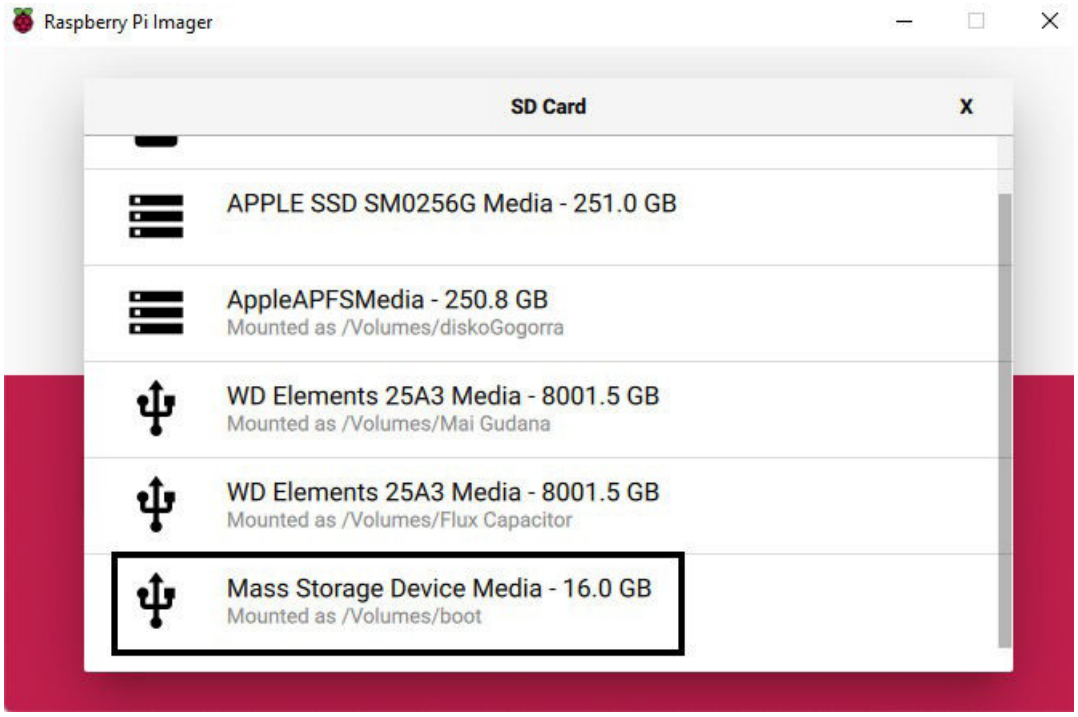
When you launch the installer, your operating system may try to block you from running it. For example, on Windows I receive the following message:



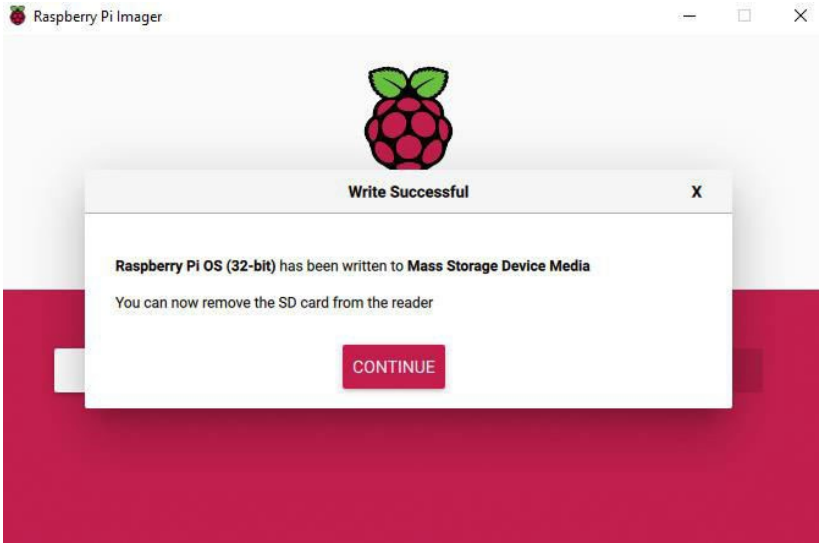
- If this pops up, click on More info and then Run anyway
- Follow the instructions to install and run the Raspberry Pi Imager
- Insert your SD card into the computer or laptop SD card slot
- In the Raspberry Pi Imager, select the OS that you want to install and the SD card you would like to install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

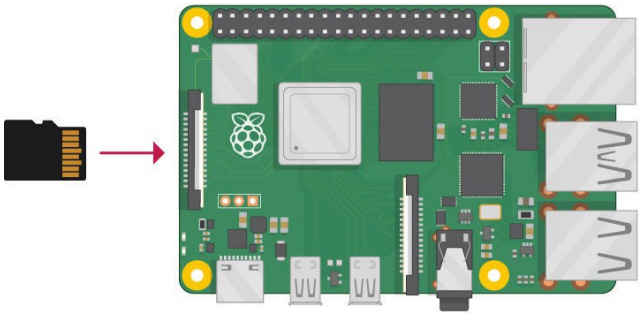
- Then simply click the WRITE button
- Wait for the Raspberry Pi Imager to finish writing
- Once you get the following message, you can eject your SD card



Connect your Raspberry Pi

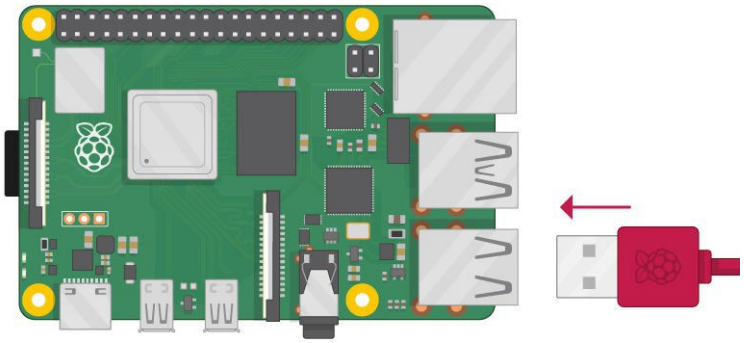
Let's connect up your Raspberry Pi and get it running.

- Check the slot on the underside of your Raspberry Pi to see whether an SD card is inside. If no SD card is there, then insert an SD card with Raspbian installed (via NOOBS).

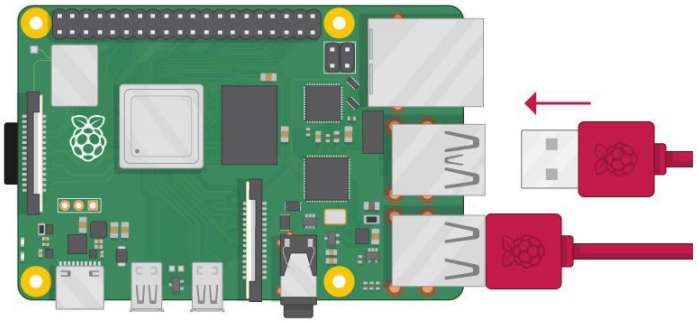


Note: Many microSD cards come inside a larger adapter — you can slide the smaller card out using the lip at the bottom.

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



- Connect the keyboard in the same way.

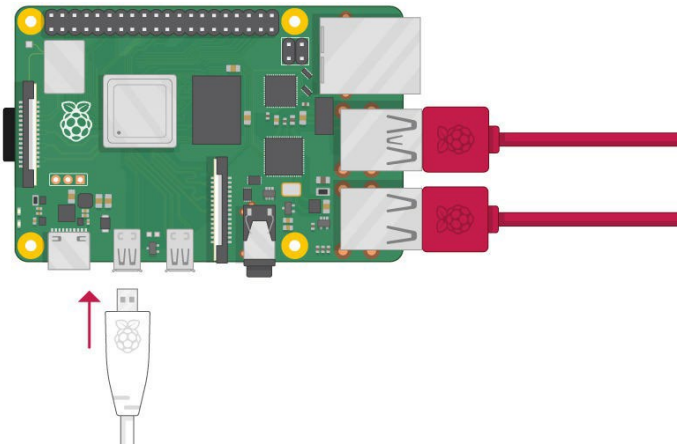


- Make sure your screen is plugged into a wall socket and switched on.
- Look at the HDMI port(s) on your Raspberry Pi — notice that they have a flat side on top.

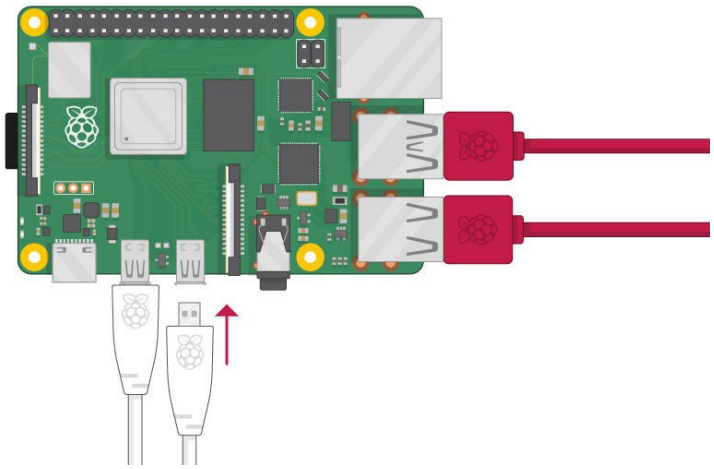
- Use a cable to connect the screen to the Raspberry Pi's HDMI port — use an adapter if necessary.

  Raspberry Pi 4

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

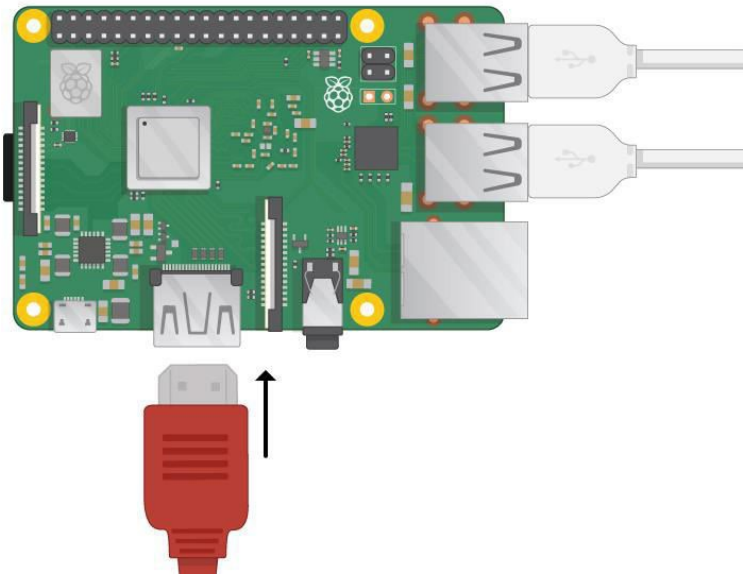


You could connect an optional second screen in the same way.



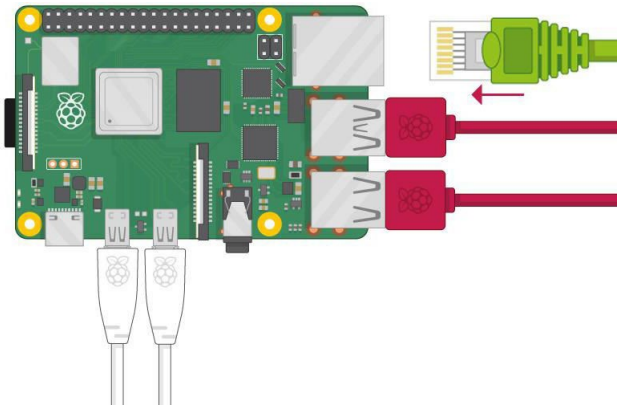
Raspberry Pi 1, 2, 3

Connect your screen to the single HDMI port.

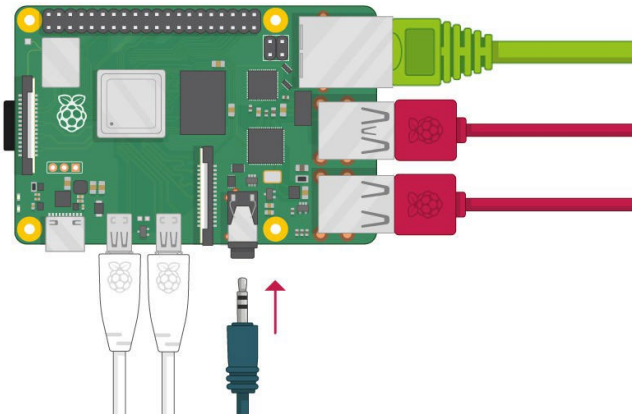


Note: nothing will display on the screen, because the Raspberry Pi is not running yet.

- If you want to connect the Pi to the internet via Ethernet, use an Ethernet cable to connect the Ethernet port on the 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.



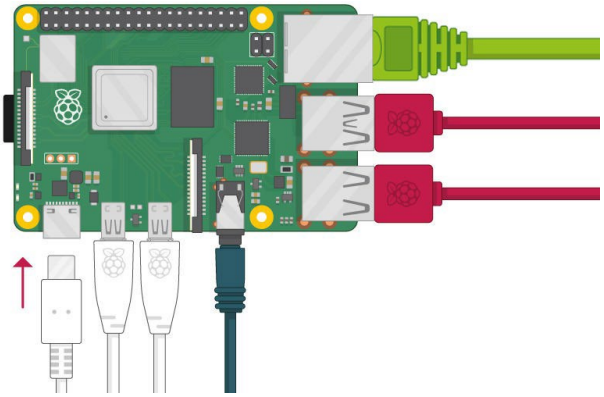
- If your screen has speakers, your Raspberry Pi can play sound through these. Or you could connect headphones or speakers to the audio port.



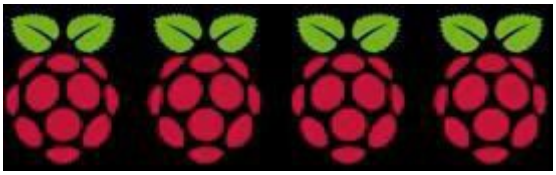
Start up your Raspberry Pi

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.

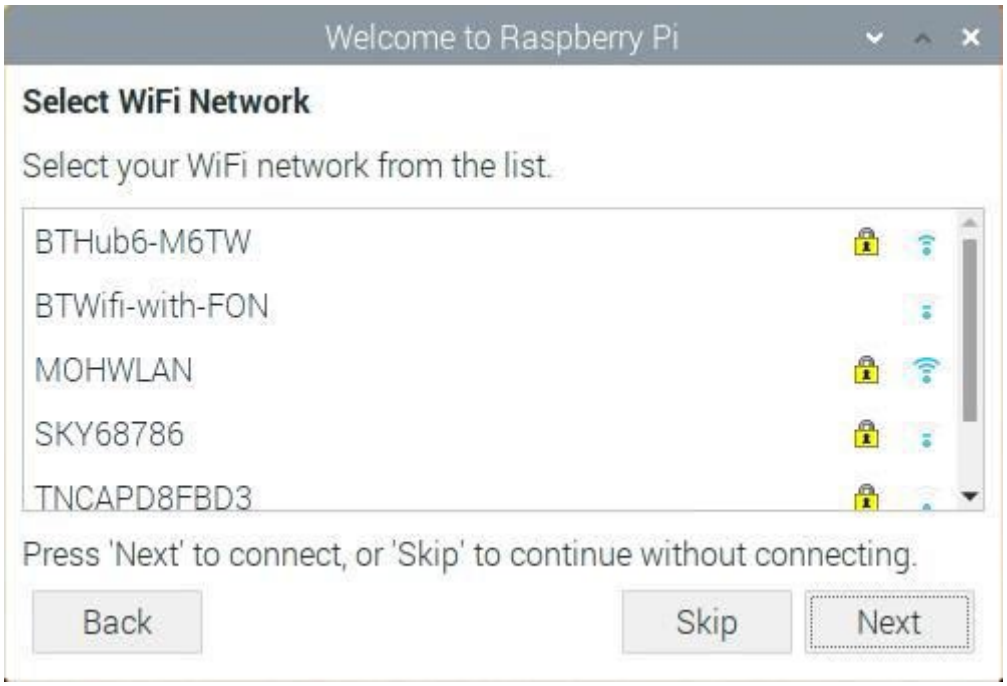
Finish the setup

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 Next to start the setup.
- Set your Country, Language, and Timezone, then click Next again.

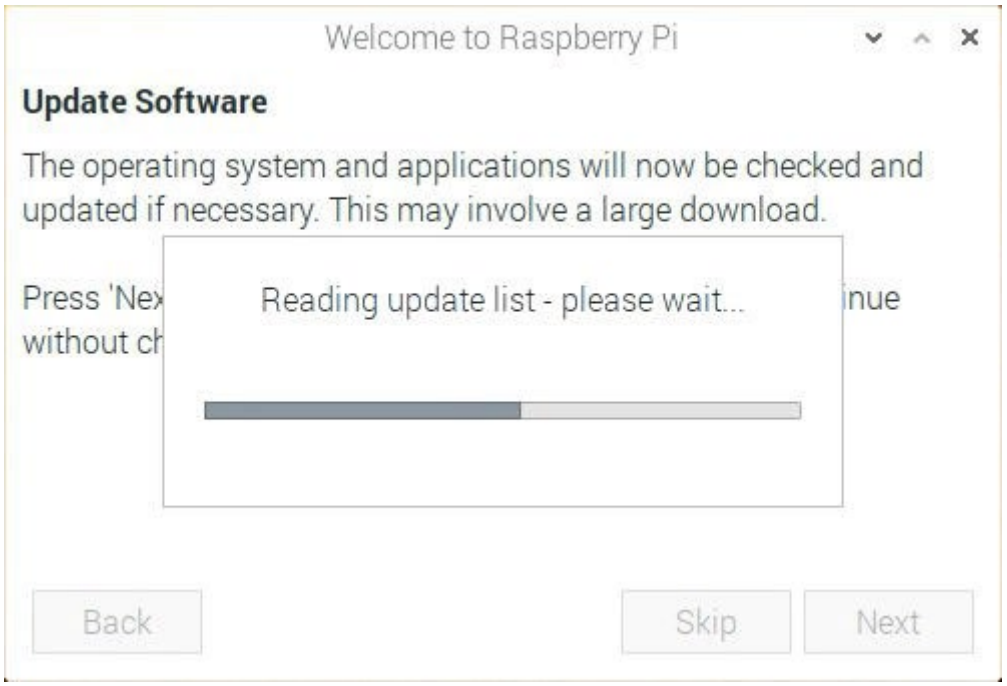
- Enter a new password for your Raspberry Pi and click Next.

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



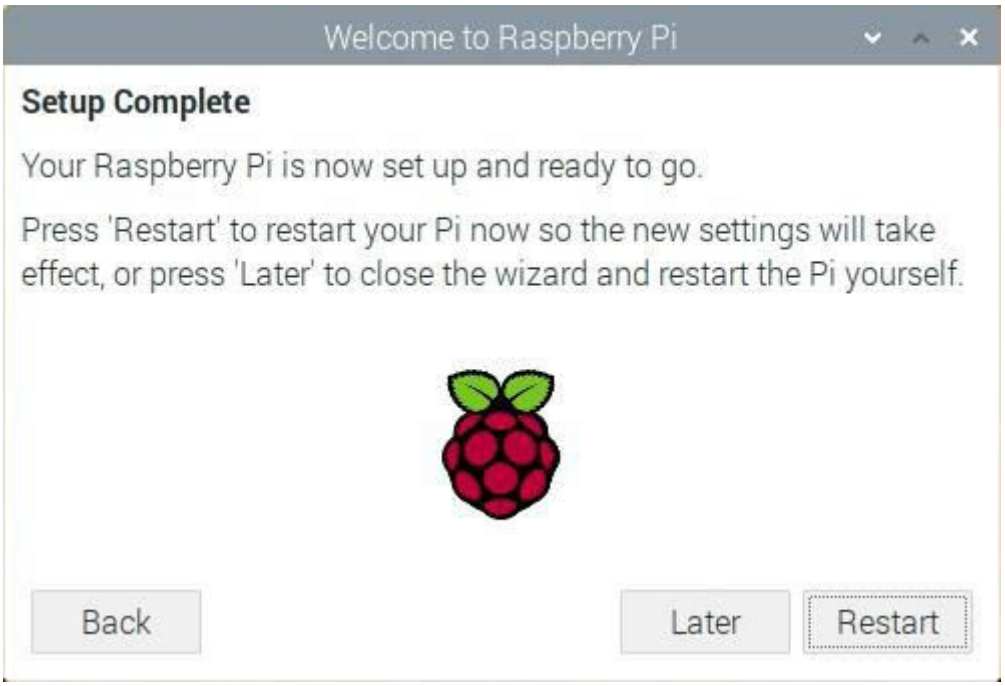
Note: if your Raspberry Pi model doesn't have wireless connectivity, you won't see this screen.

- Click Next let the wizard check for updates to Raspbian and install them (this might take a little while).



- Click Done or Reboot to finish the setup.

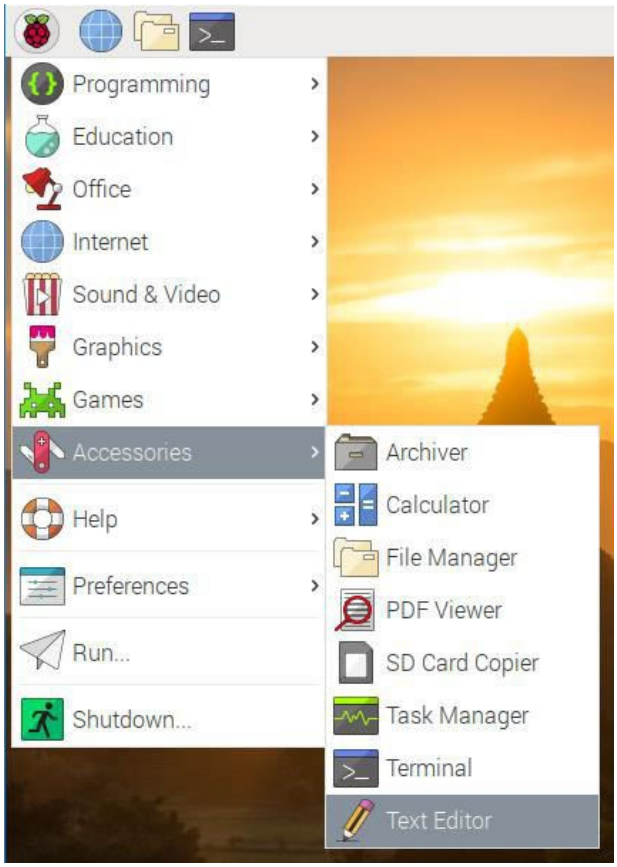
Note: you will only need to reboot if that's necessary to complete an update.



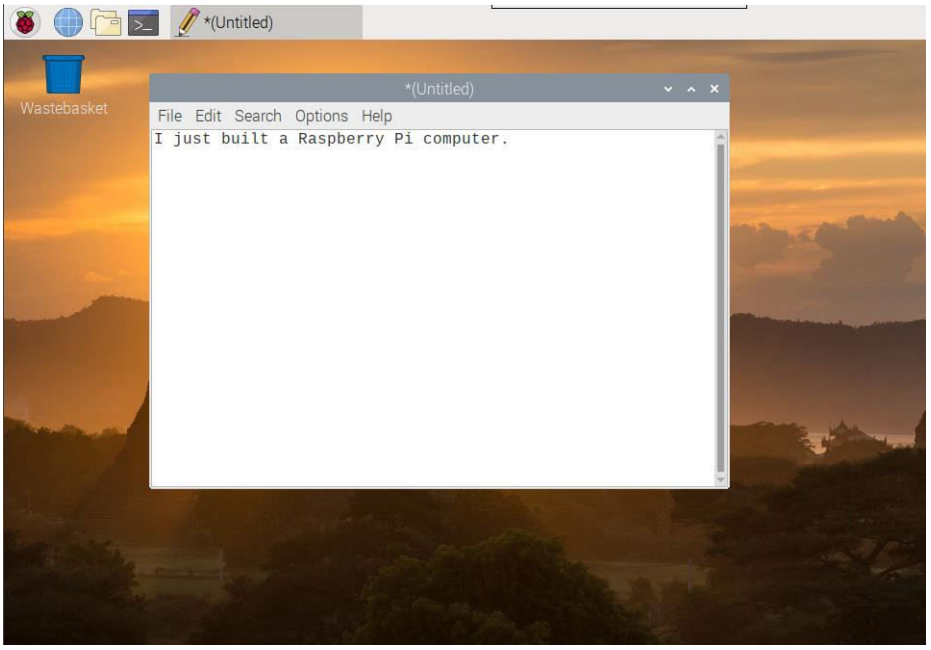
A tour of Raspberry Pi

Now it's time to take a tour of your Raspberry Pi.

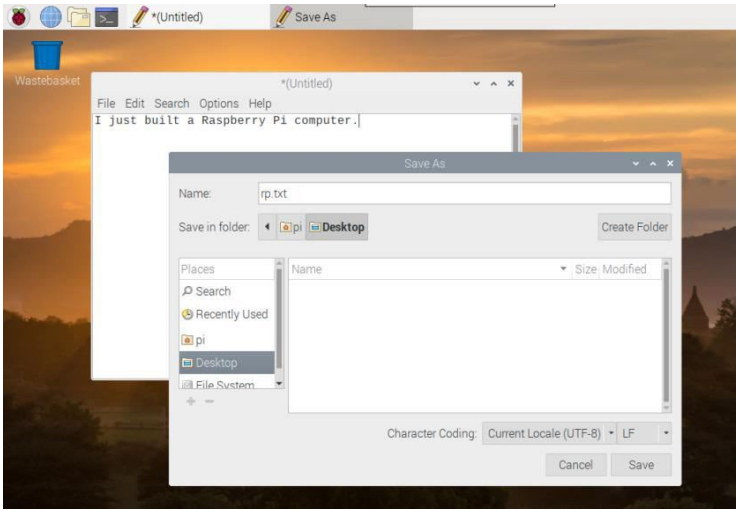
- Do you see the raspberry symbol in the top left-hand corner? That's where you access the menu: click on it to find lots of applications.
- Click on Accessories, and then click on Text Editor.



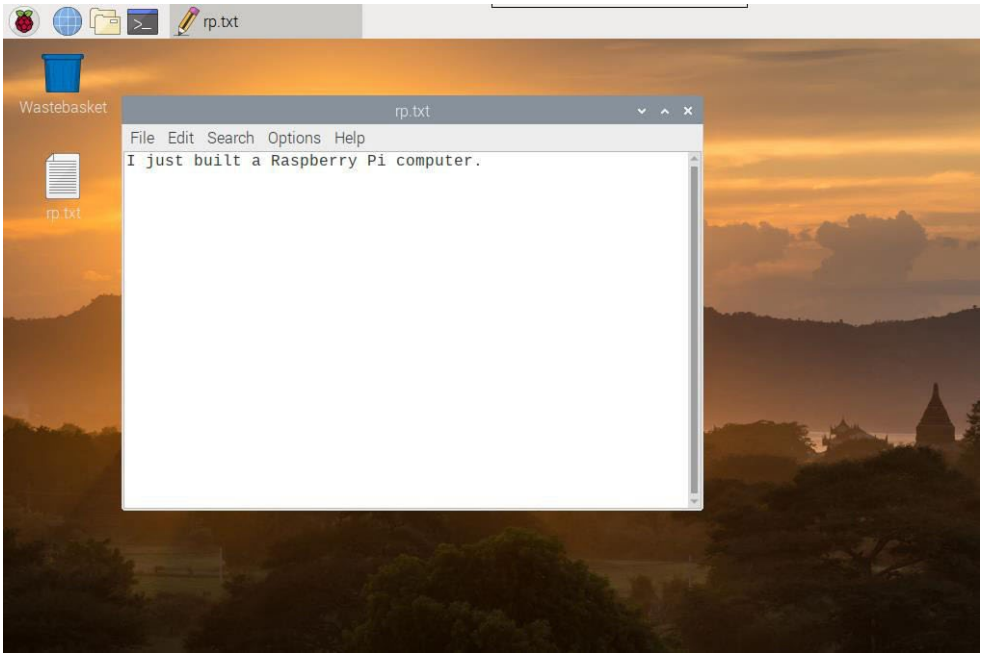
- Type I just built a Raspberry Pi computer in the window that appears.



- Click on File, then choose Save, and then click on Desktop and save the file as .

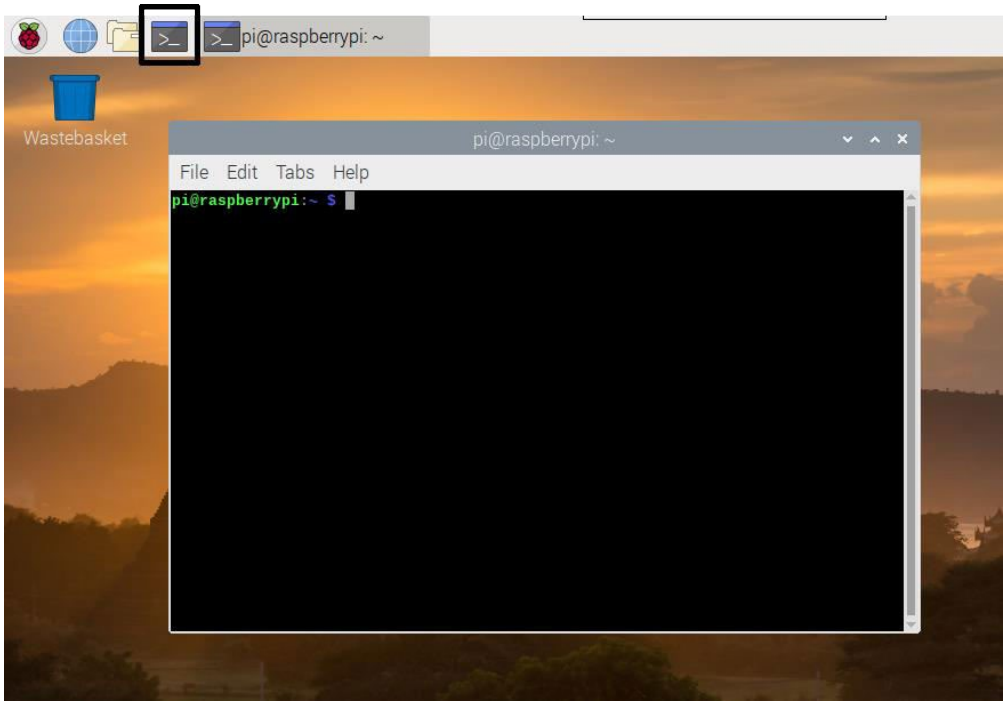


- You should see an icon named rp.txt appear on the desktop.



Your file has been saved to your Raspberry Pi's SD card.

- Close the text editor by clicking the X in the top right-hand corner of the window.
- Return to the menu, click on Shutdown, and then click on Reboot.
- When Raspberry Pi has rebooted, your text file should still be there on the desktop.
- Raspberry Pi runs a version of an operating system called Linux (Windows and macOS are other operating systems). This operating system allows you to make things happen by typing in commands instead of clicking on menu options. To try this out, click on the Terminal symbol at the top of the screen:



- In the window that appears, type:

```
ls
```

and then press Enter on the keyboard.

You can now see a list of the files and folders in your directory.

- Now type this command to change directory to the Desktop:

```
cd Desktop
```

You have to press the Enter key after every command.

Then type:

```
ls
```

Can you see the text file you created?

- Close the terminal window by clicking on the X.
- Now drag to the Wastebasket on the desktop so the Raspberry Pi will be tidy for the next person using it.

Browsing the web

You might want to connect your Raspberry Pi to the internet. If you didn't plug in an ethernet cable or connect to a WiFi network during the setup, then you can connect now.

- Click the icon with red crosses in the top right-hand corner of the screen, and select your network from the drop-down menu. You may need to ask an adult which network you should choose.

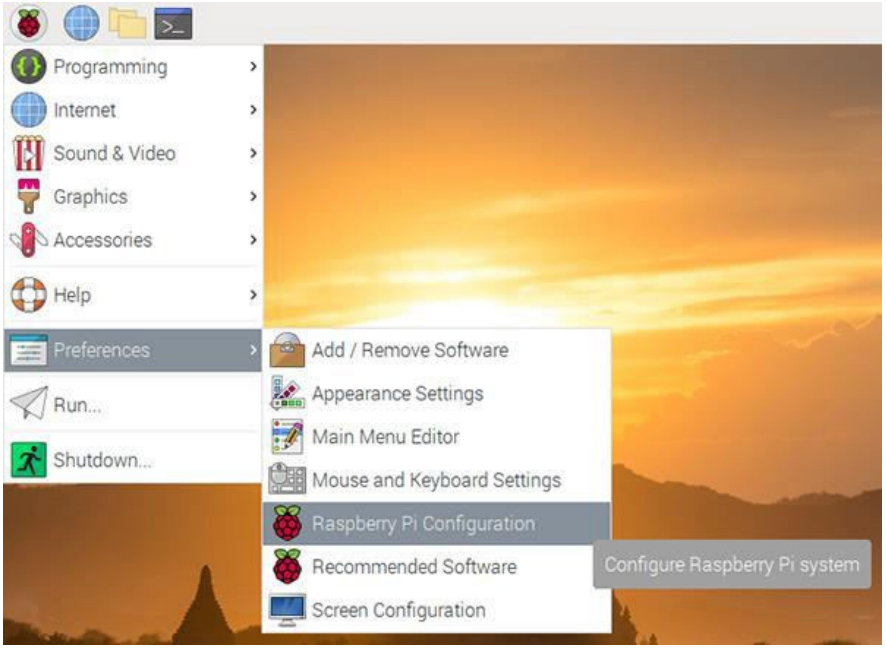
- Type in the password for your wireless network, or ask an adult to type it for you, then click OK.

- When your Pi is connected to the internet, you will see a wireless LAN symbol instead of the red crosses.

- Click the web browser icon and search for .

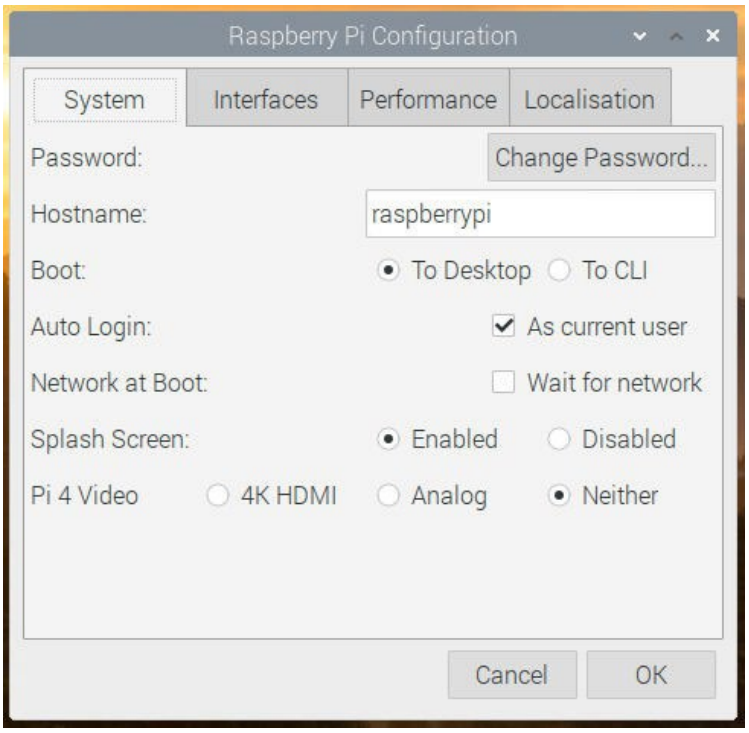
Configuring your Raspberry Pi

You can control most of your Raspberry Pi's settings, such as the password, through the Raspberry Pi Configuration application found in Preferences on the menu.



- System

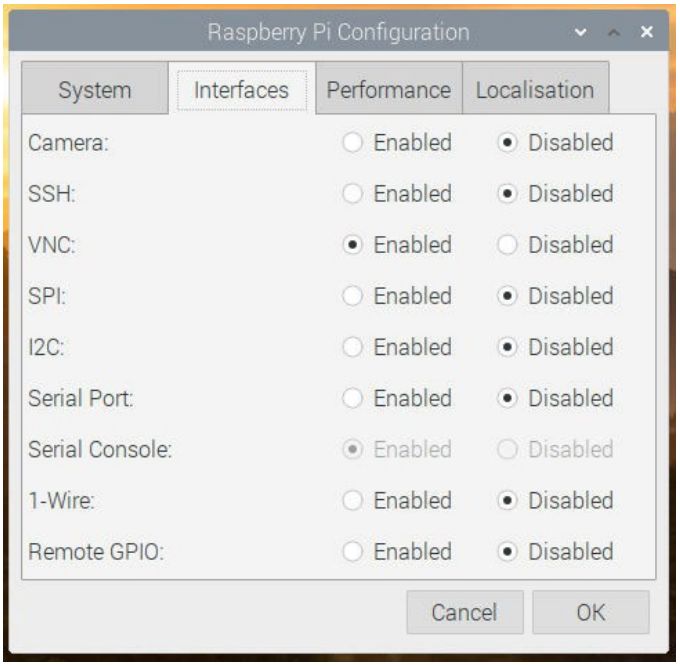
In this tab you can change basic system settings of your Raspberry Pi.



- Password — set the password of the pi user (it is a good idea to change the password from the factory default 'raspberry')
- Boot — select to show the Desktop or CLI (command line interface) when your Raspberry Pi starts
- Auto Login — enabling this option will make the Raspberry Pi automatically log in whenever it starts

- Network at Boot — selecting this option will cause your Raspberry Pi to wait until a network connection is available before starting
- Splash Screen — choose whether or not to show the splash (startup) screen when your Raspberry Pi boots
- Interfaces

You can link devices and components to your Raspberry Pi using a lot of different types of connections. The Interfaces tab is where you turn these different connections on or off, so that your Raspberry Pi recognizes that you've linked something to it via a particular type of connection.

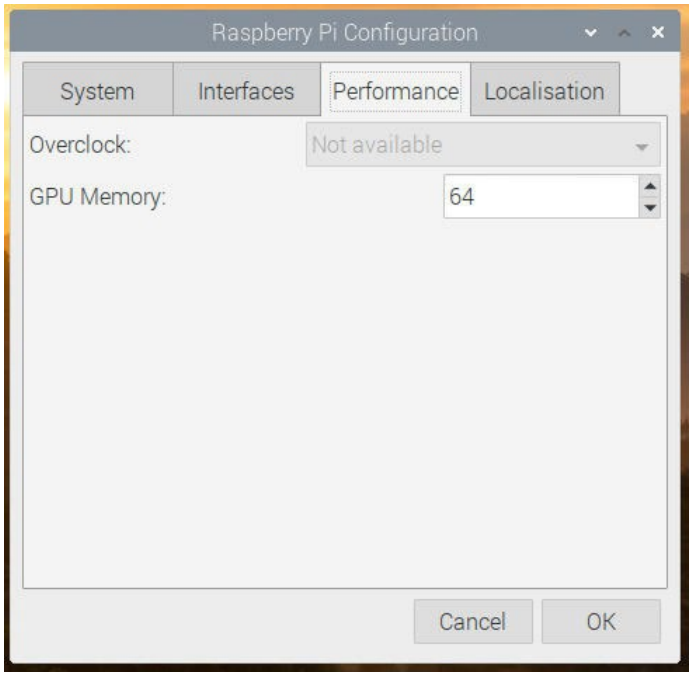


- Camera — enable the Raspberry Pi Camera Module
- SSH — allow remote access to your Raspberry Pi from another computer using SSH
- VNC — allow remote access to the Raspberry Pi Desktop from another computer using VNC
- SPI — enable the SPI GPIO pins
- I2C — enable the I2C GPIO pins
- Serial — enable the Serial (Rx, Tx) GPIO pins
- 1-Wire — enable the 1-Wire GPIO pin
- Remote GPIO — allow access to your Raspberry Pi's GPIO pins from another computer

- Performance

If you need to do so for a particular project you want to work on, you can change the performance settings of your Raspberry Pi in this tab.

Warning: Changing your Raspberry Pi's performance settings may result in it behaving erratically or not working.



- Overclock — change the CPU speed and voltage to increase performance

- **GPU Memory** — change the allocation of memory given to the GPU
- Localization

This tab allows you to change your Raspberry Pi settings to be specific to a country or location.

- Locale — set the language, country, and character set used by your Raspberry Pi

- Timezone — set the time zone
- Keyboard — change your keyboard layout
- WiFi Country — set the WiFi country code

Experiment - 2

Exporting Display On To Other Systems

Making use of available laptop/desktop displays as a display for the device using SSH client & X11 display server.

How Does it Work?

To connect a Raspberry Pi to a laptop display, you can simply use an Ethernet cable. The Raspberry Pi's desktop GUI (Graphical User Interface) can be viewed through the laptop display using a 100 Mbps Ethernet connection between the two. There are many software programs available that can establish a connection between a Raspberry Pi and your laptop. We used VNC server software to connect the Pi to our laptop. Installing the VNC server on your Pi allows you to see the Raspberry Pi's desktop remotely, using the mouse and keyboard as if you were sitting right in front of your Pi. It also means that you can put your Pi anywhere else in your home and still control it. Also, the internet can be shared from your laptop's WiFi over Ethernet. This also lets you access the internet on the Pi and connect it to your laptop display.

Setting up your Raspberry Pi

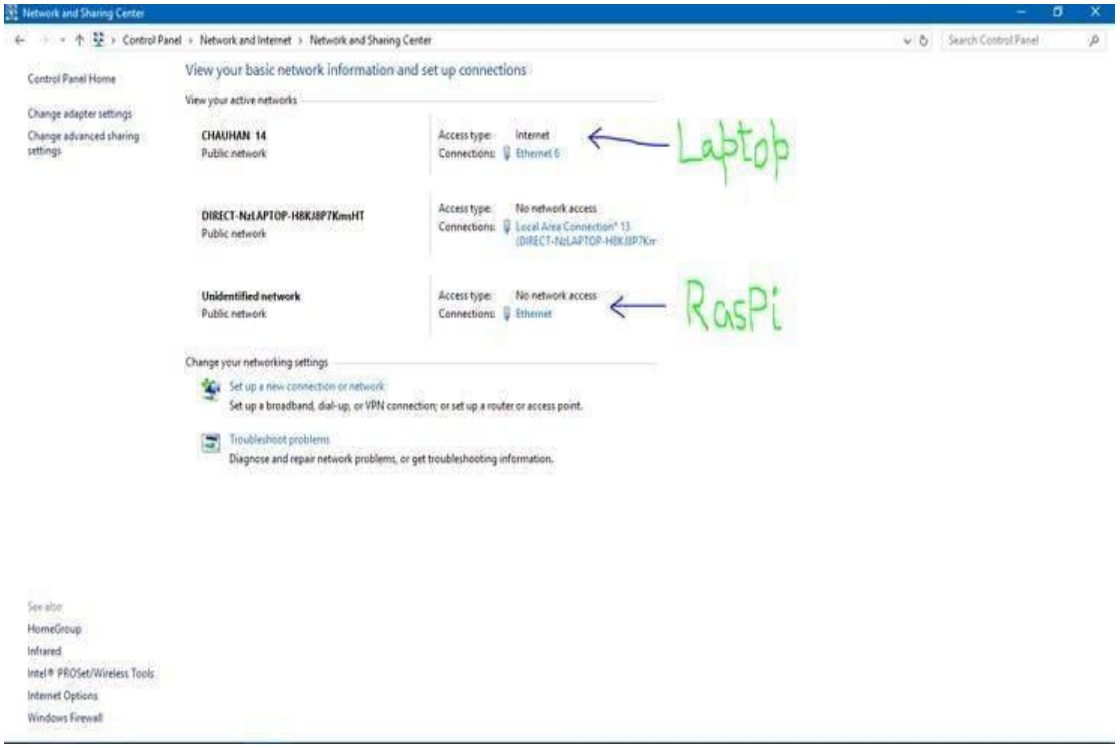
Before moving to connect your Raspberry Pi to your laptop display, you need an SD card with the OS preinstalled, or install **Raspbian** on a blank SD card. You will find lots of blogs and tutorials on preparing an SD card for the Raspberry Pi. If you are a beginner, you can simply click **here**and know more about this. This will show how to install the OS for the Raspberry Pi. You can also buy SD cards with the **Raspbian** and **NOOBs** operating systems preinstalled. *I would suggest you install* the *latest* ***full Raspbian OS image*** *from the official Raspberry Pi website as it is having VNC Server in the OS package.*

After setting up your SD Card, insert it into the Raspberry Pi. Next, connect your power adapter to the Raspberry Pi to power it. Also, connect your Raspberry Pi to the laptop via an Ethernet cable and connect a keyboard and mouse to it.

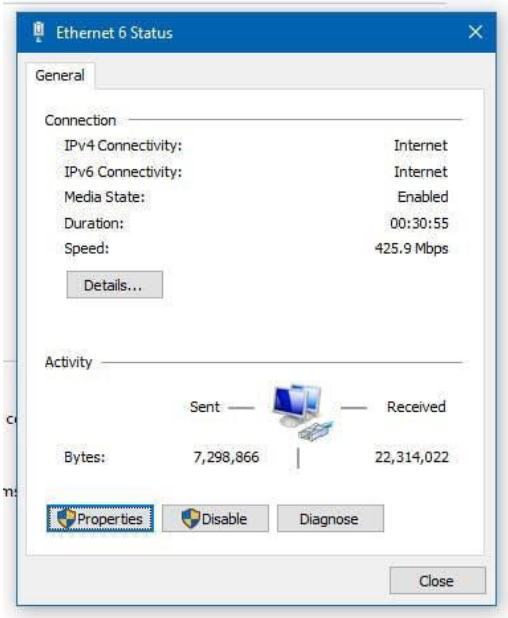
Note: *You need screen and a mouse after booting a new OS into Pi for the first time as by default, the SSH and VNC are disabled in Pi. Without SSH disabled, we cannot enable the PuTTY Configuration.*

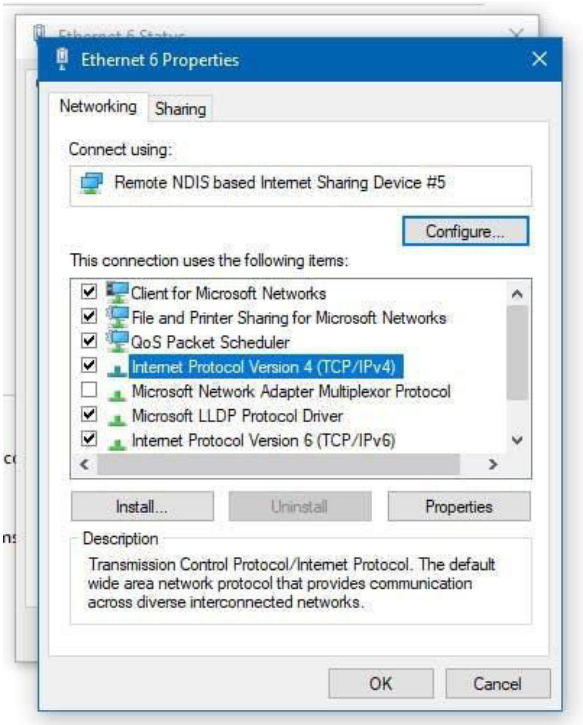
Sharing Internet Over Ethernet

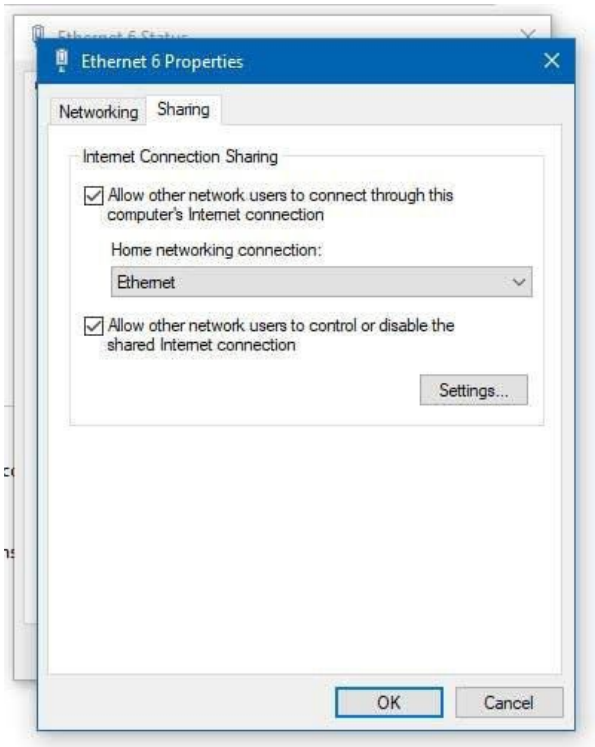
This step explains how you can share your laptop internet with the Raspberry Pi via Ethernet cable. In Windows: To share the internet with multiple users over Ethernet, go to **Network and Sharing Center.** Then click on the WiFi network:



Click on Properties (shown below), then go to **Sharing** and click on **"Allow other network users to connect".** Make sure that the networking connection is changed to the connection of the Raspberry Pi. In my case, it is **Ethernet**:







Finding IP for PuTTY Configuration

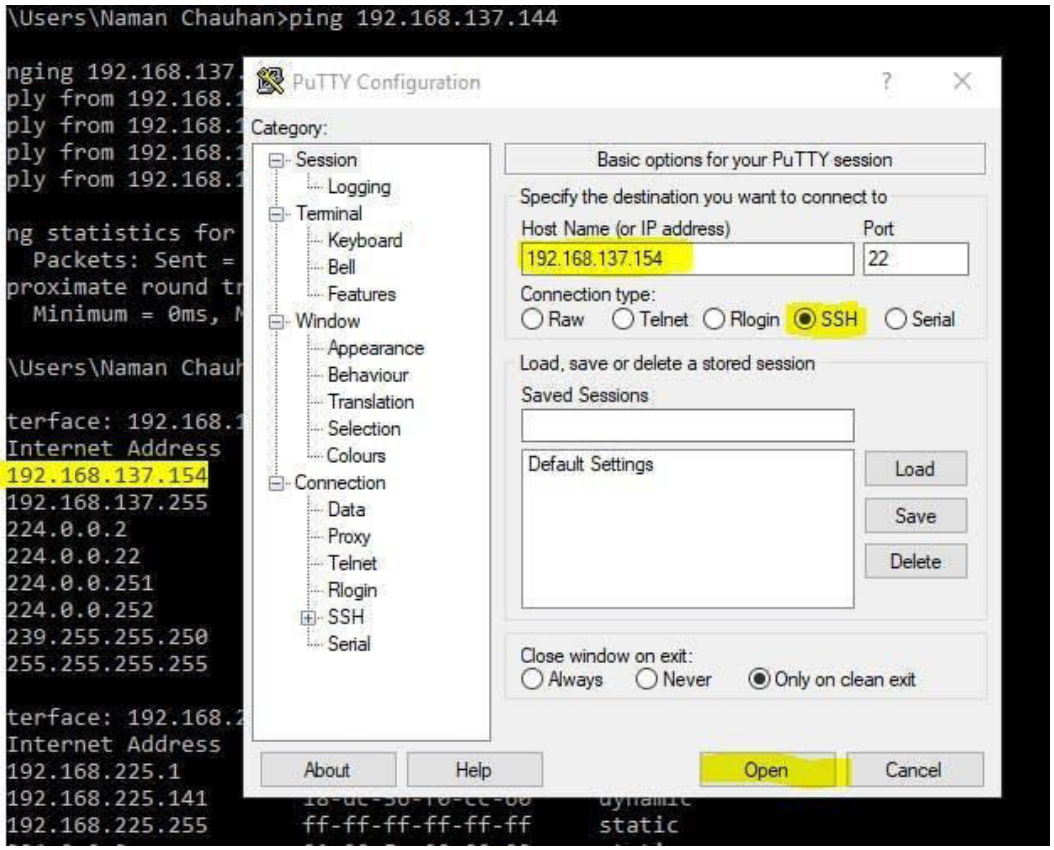
By default, the laptop will give a *dynamic IP* to the Raspberry Pi. Thus, we have to find out the IP address of Pi now.

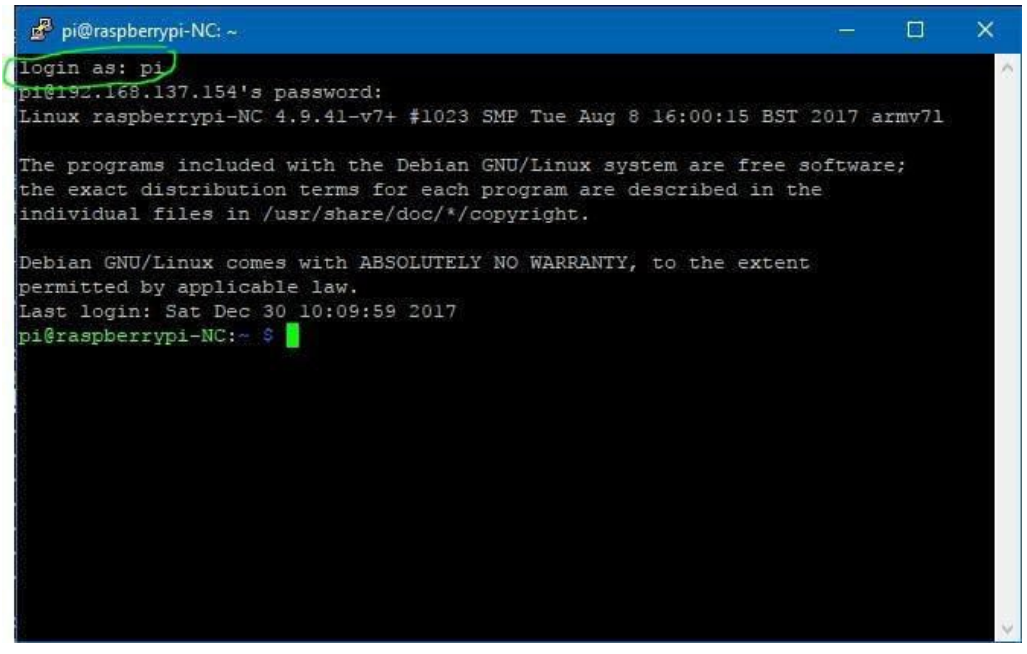
As shown above, the IP assigned to my Pi is **192.168.137.144**. To check the IP assigned to the connected Ethernet device, do the following. Considering that the IP assigned to your Pi is **192.168.137.144** and the subnet mask is **255.255.255.0** :

- Open the command prompt.
- Ping at **raspberrypi.mshome.net**.
- Stop the ping after 5 seconds.

Here, it is **192.168.137.154**. Note this somewhere.

PuTTY Configuration and VNC on Raspberry Pi



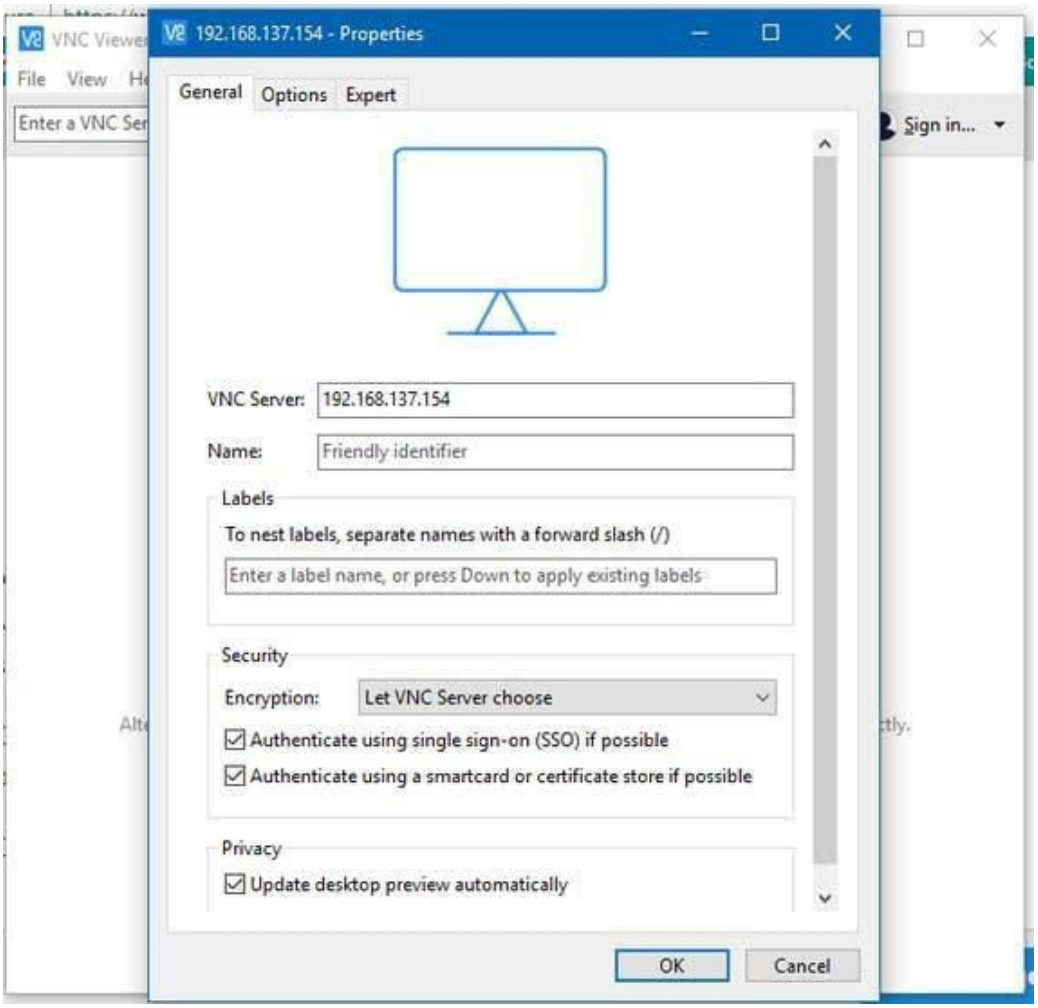


- In the **Host Name,** enter the **IP Address** we noted from the command line.
- Ensure that the **Connection Type** is **SSH.**
- Hit **Enter** or click on **Open** to proceed.
- Now, a **new window** will open. It looks like a normal terminal window of the computer but it is *Raspberry Pi's terminalwindow* accessible on your laptop.
- It is display- `login as:`
- Enter `pi` as the username.
- Enter the **password** you set for the Raspberry Pi. The **default password** is `raspberry`
- If the password is correct, the Pi will load and you will access the terminal window of the Pi.
- Now, you need to start the VNC Server. Enter after the $ sign - `sudo vncserver :1`
- This is to initialize the VNC Server on the Raspberry Pi.

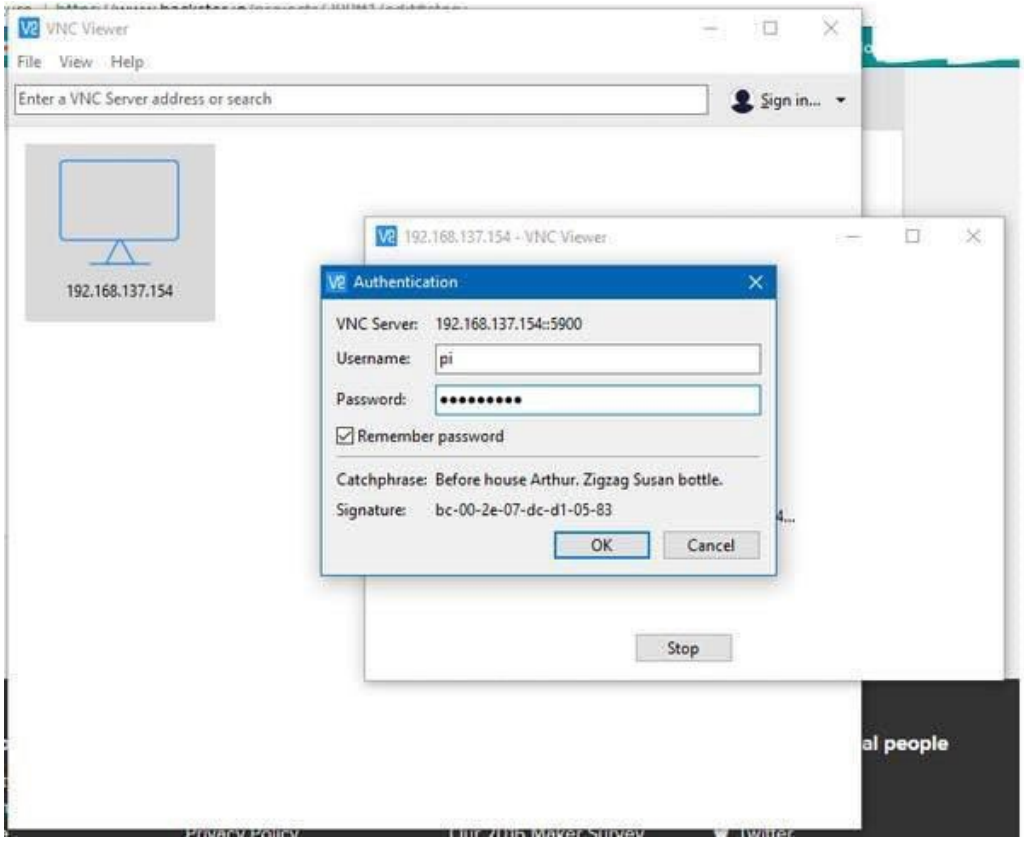
VNC Server and VNC Viewer on Laptop

Now, Raspberry is ready to connect using VNC. We just need to install the VNC server on the laptop.

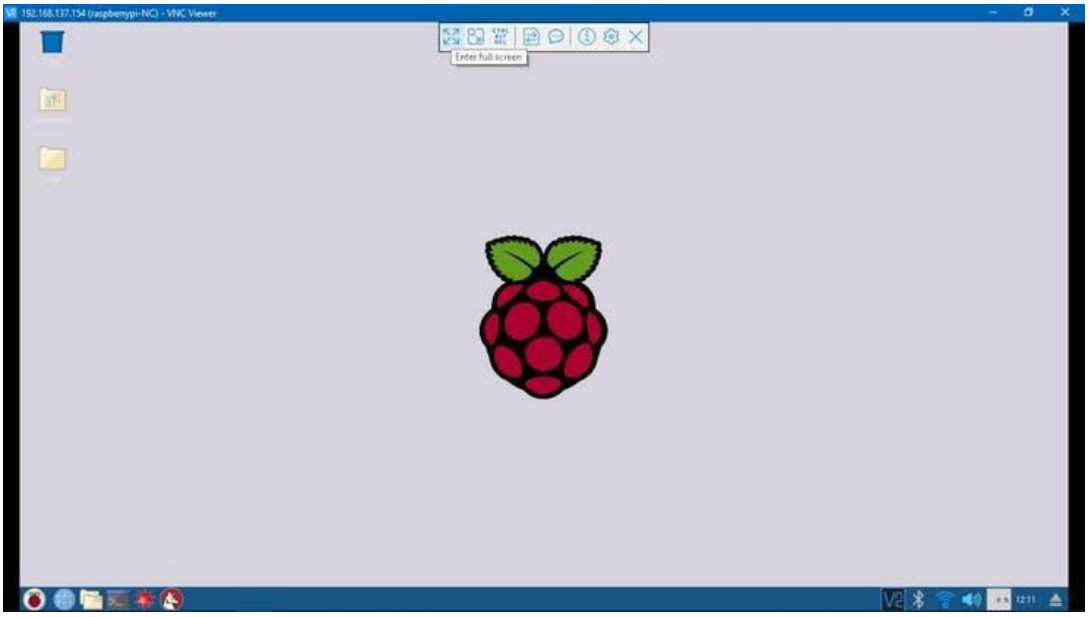
- Download **VNC Client** and install it. Now, download the **VNC Viewer**and install it on the laptop.
- Open the VNC Server and the VNC Viewer now.
- In the VNC Viewer, click on **File** > **New Connection**.
- Enter **IP Address** and in **Options** > **Picture Quality**, select **High.**



- Click OK. Now, double click on the IP Address.
- Enter **pi** in Username and your Pi's password (default is **raspberry**).
- Click on **Remember Password** so that you don't have to enter this next time.
- Click on **OK**.



As you hit enter and all the things are correct, the Raspberry Pi Desktop will load in a new window. You can go into a full-screen mode by clicking on the options available above on the window.

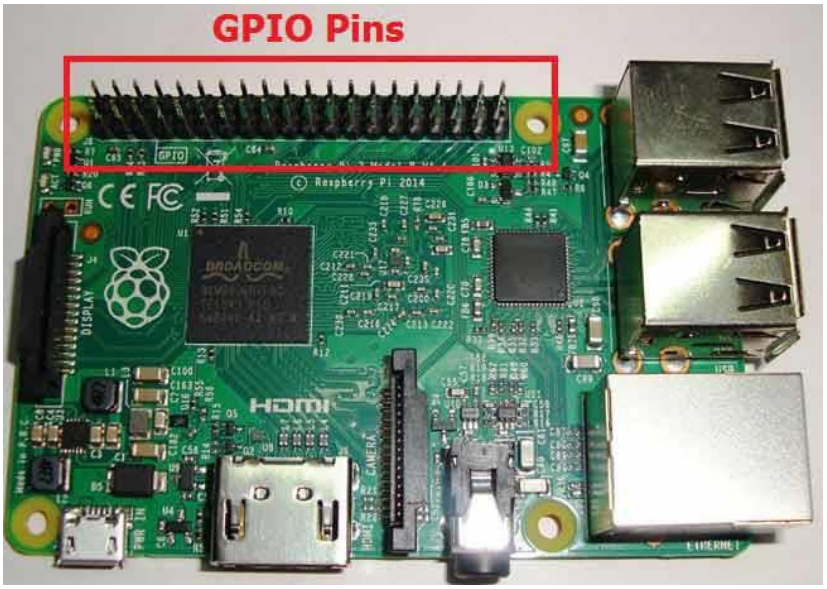


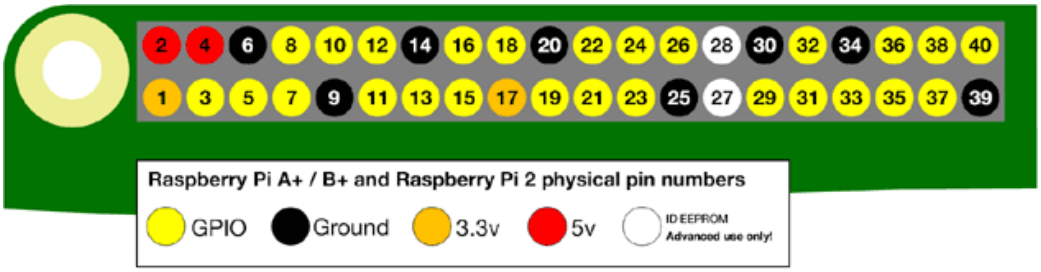
Experiment - 3

GPIO Programming

Programming of available GPIO pins of the corresponding device using native programming language. Interfacing of I/O devices like LED/Switch etc., and testing the functionality.

GPIO Pins :





As shown in above figure, there are 40output pins for the PI. But when you look at the second figure, you can see not all 40 pin out can be programmed to our use. These are only 26 GPIO pins which can be programmed. These pins go from **GPIO2 to GPIO27**.

These **26 GPIO pins can be programmed** as per need. Some of these pins also perform some special functions, we will discuss about that later. With special GPIO put aside, we have 17 GPIO remaining (Light green Cirl).

Each of these 17 GPIO pins can deliver a maximum of **15mA current**. And the sum of currents from all GPIO cannot exceed 50mA. So we can draw a maximum of 3mA in average

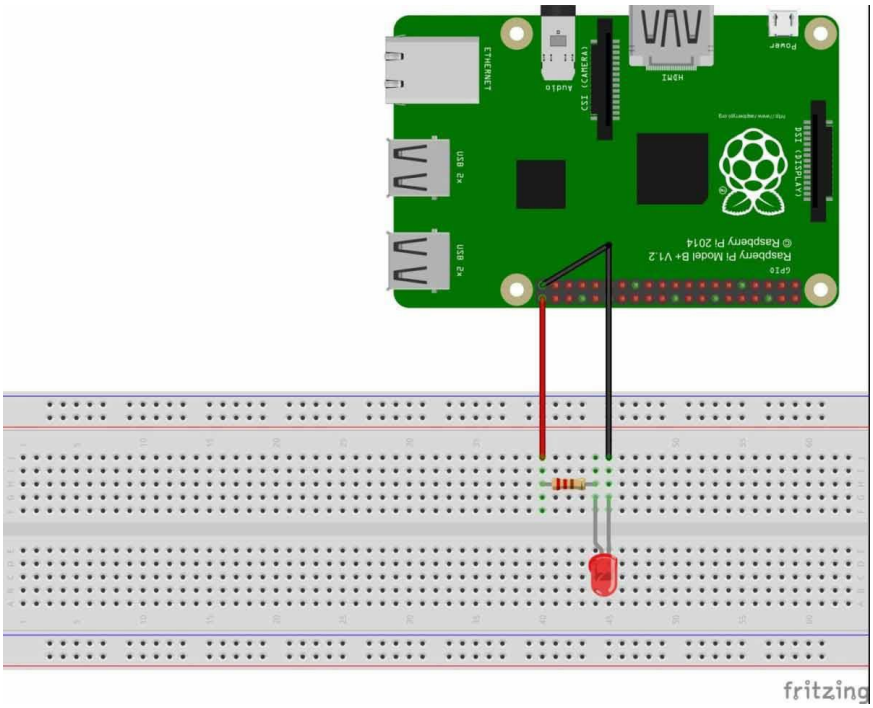
from each of these GPIO pins. So one should not tamper with these things unless you know what you are doing.

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, other than that we need:

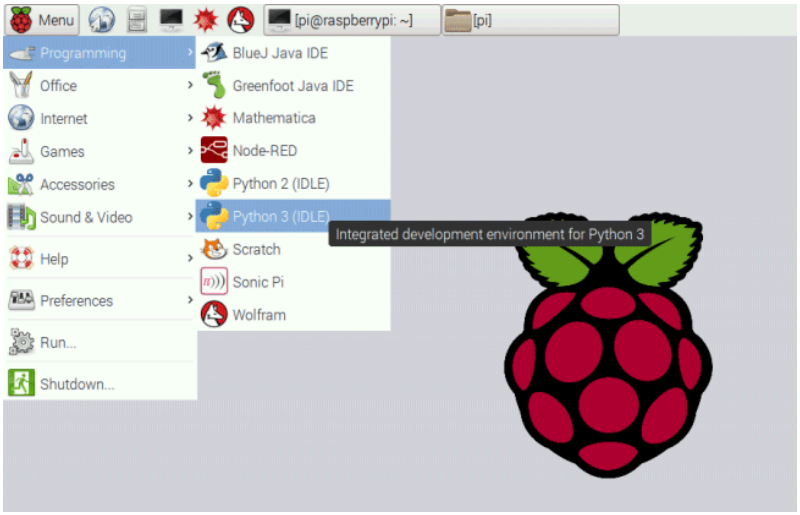
- Connecting pins
- 220Ω or 1KΩresistor
- LED
- Bread Board

Circuit Diagram:

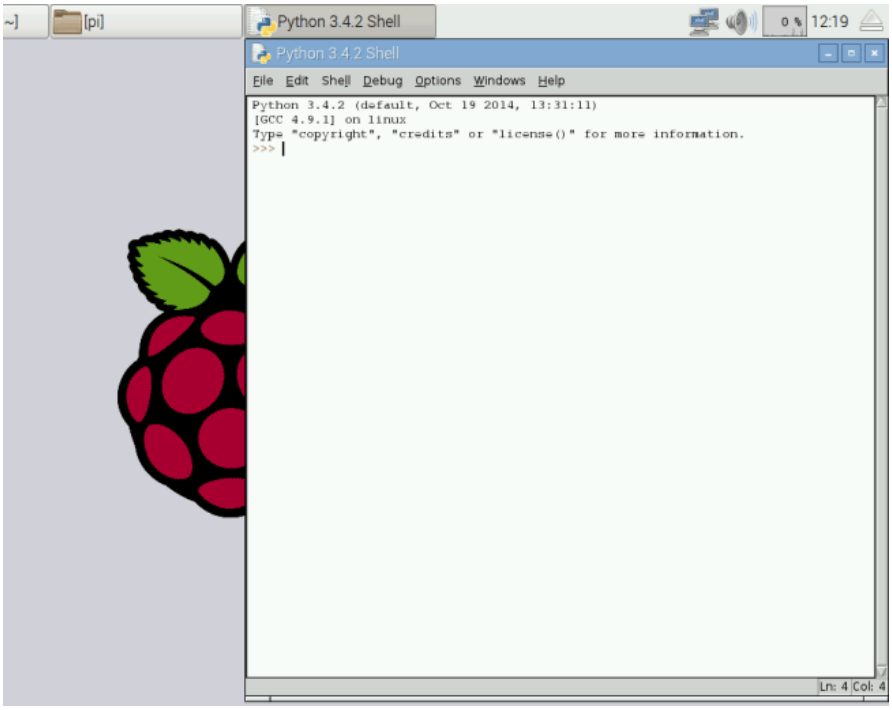


Steps:

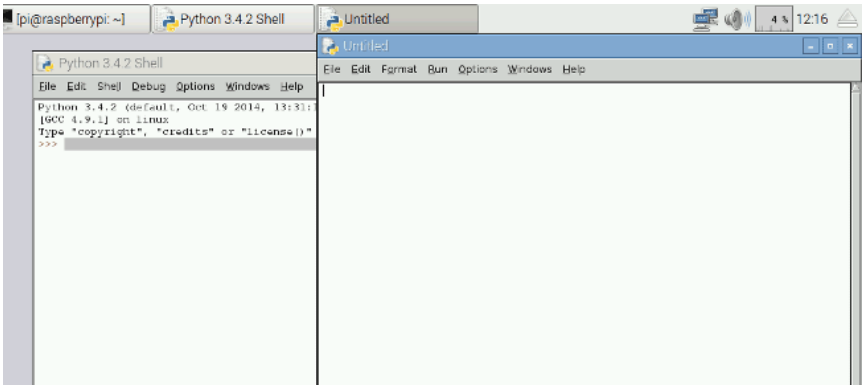
1. On the desktop, go the Start Menu and choose for the **PYTHON 3**, as shown in figure below.



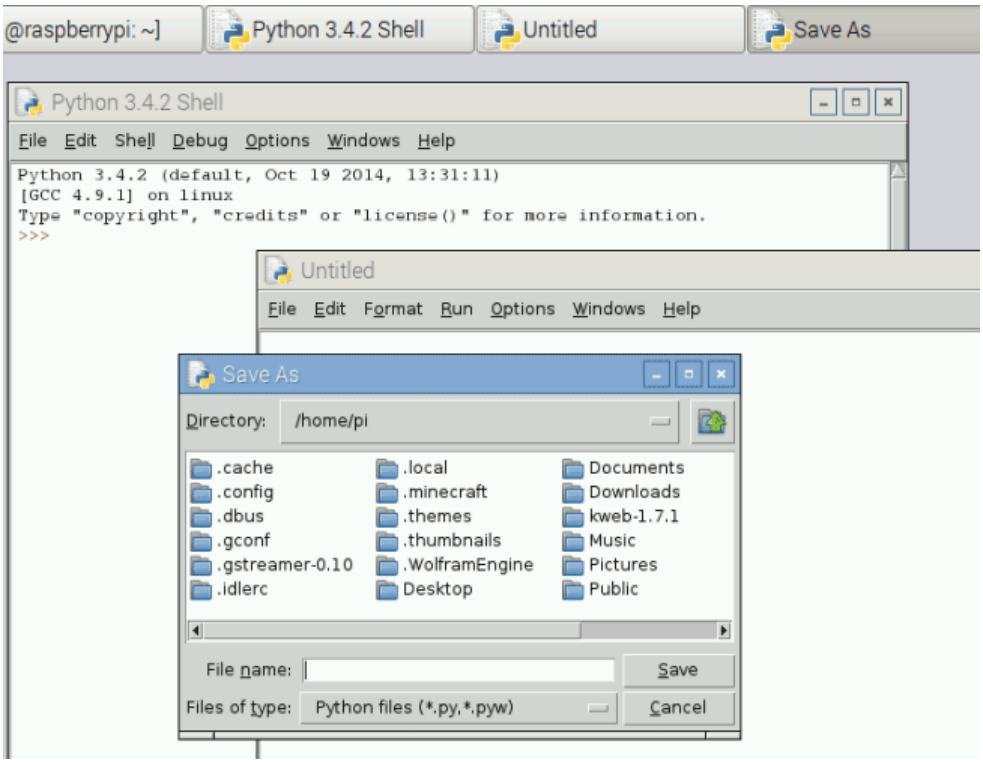
2. After that, PYHON will run and you will see a window as shown in below figure.



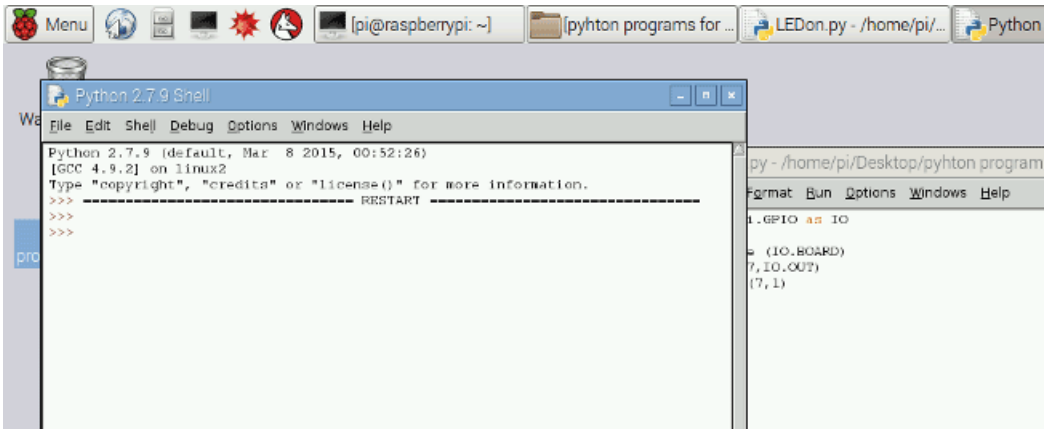
3. After that, click on *New File* in *File* Menu, You will see a new Window open,



4. Save this file as *blinky* on the desktop,



5. After that write the program for *blinky* as given below and execute the program by clicking on "RUN" on 'DEBUG' option.



If the program has no errors in it, you will see a ">>>", which means the program is executed successfully. By this time you should see the LED blinking three times. If there were any errors in the program, the execution tells to correct it. Once the error is corrected execute the program again.

Program:

```
from gpiozero import LED
from time import sleep

led = LED(17)

while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

Experiment – 4

Interfacing Chronos eZ430

Chronos device is a programmable texas instruments watch which can be used for multiple purposes like PPT control, Mouse operations etc., Exploit the features of the device by interfacing with devices.

Using the eZ430 Chronos with a Raspberry Pi

The **eZ430 Chronos** development kit from Texas Instruments represents great value for money and provides a wristwatch with a wireless-enabled microcontroller, accelerometers and temperature and barometric pressure sensors, and a USB programmer and RF access point. In this post I take a look at what it takes to get it up and running with a Raspberry Pi.

The Chronos RF access point simply presents itself as a serial port to the operating system and drivers are included in Linux, and so any heavy lifting in enabling communications between the watch and host has already been done for us.

With the access point plugged into the Raspberry Pi USB we just need to install a few dependencies in order to run the TI supplied demonstration software and a simple example

Python script. Assuming that you are running Debian Linux this can be achieved using the command:

$ sudo apt-get install python-serial tcl8.5 tk8.5 xdotool

Chronos Control Center

Chronos Control Center is a GUI tool that provides a selection of applications which demonstrate the capabilities of the eZ430 Chronos. The Linux version of the software must have been developed with x86 architecture in mind as it's provided as a binary installer rather than a tar archive. However, since it's Tcl/Tk based it should run on just about any platform/architecture for which this software is available. It's trivial to repackage it so that it's not architecture-specific, and this just requires access to an Intel/AMD Linux machine on which to run the following commands:

$ unzip slac388a.zip

$ ./Chronos-Setup

$ tar zcvf ccc.tgz ~/Texas Instruments/eZ430-Chronos

Obviously if you installed the software to a location other than the default as part of the second step, you will need to use that location for the second argument in the third step. The ccc.tgz archive can then be copied to the Raspberry Pi and unpacked to a suitable location.

Control Center software running, with the access point enabled and the watch set to *ACC* mode and with RF enabled. Real-time data from the watch accelerometers is displayed, and by selecting *Mouse On* it's also possible to use the watch to control the Raspberry Pi mouse pointer through gesture. As can be seen the Control Center provides a number of other simple applications that can be selected via the tabs at the top.

Setting the time via a Python script

It should be possible to write host-based applications for the Chronos in just about any language that provides access to serial devices. When using the Python language this is achieved via the **pySerial** library, and with a **reasonably short script** it's possible to

configure the serial port, send the commands required to start up the RF access point, and then get the Raspberry Pi system time, format this into packets, transmit them to the watch and set the time accordingly.

Note that if you do wish to make use of the linked script you will need to change the line that configures the serial port parameters to read:

ser = serial.Serial('/dev/ttyACM0',115200,timeout=1)

Conclusion

Together the Chronos eZ430 and Raspberry Pi opens up all sorts of exciting possibilities, where data can be sourced from the watch sensors or the Internet, processed and pushed in either direction. With the relatively powerful processing capabilities of the Raspberry Pi being made use of, and its hardware capabilities further extended via the GPIO port. As such it would seem like a winning combination for low cost experimentation with wearable and ubiquitous computing. And with a little enhanced support from within the Python language, it is easy to see how the Chronos could become an incredibly fun accessory to Raspberry Pi-based learning in schools.

\

Experiment - 5

ON/OFF Control Based On Light Intensity

Using the light sensors, monitor the surrounding light intensity & automatically turn ON/OFF the high intensity LED's by taking some pre-defined threshold light intensity value.

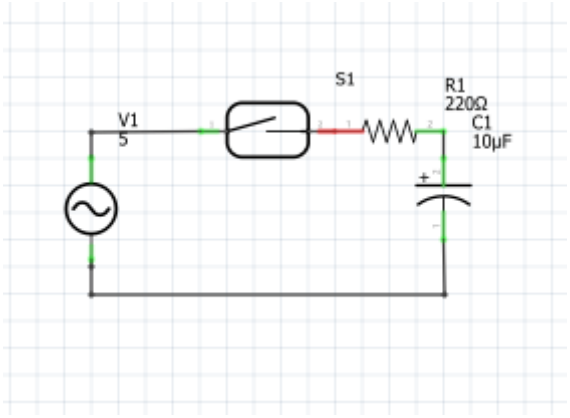
Measure the intensity of light in a room using a single photocell and a capacitor connected to the raspberry pi with a bit of code in python.

What is Photocell?

The Photocell is a light sensor in which the resistance varies according to the intensity of light. The resistance reduces when it is in brighter surroundings. We have to set up a threshold value for the measurements of the intensity because it cannot give the precise measurements. If the measurements are below the threshold then it is dark, else it is bright.

Role of a Capacitor

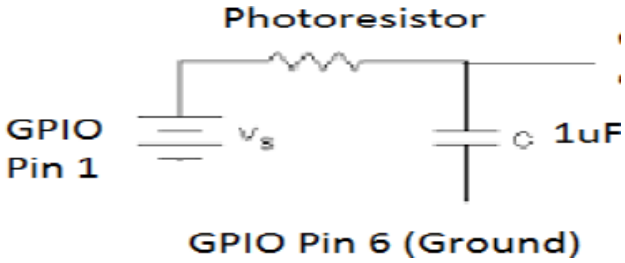
A Capacitor is an electrical component that can store electrical energy temporarily. It is measured in Farads which is characterized by capacitance. The capacitor consists of 2 conductors that can hold the electric charge and when it is fully charged the capacitor starts discharging. This kind of alternative behavior is used to generate AC.



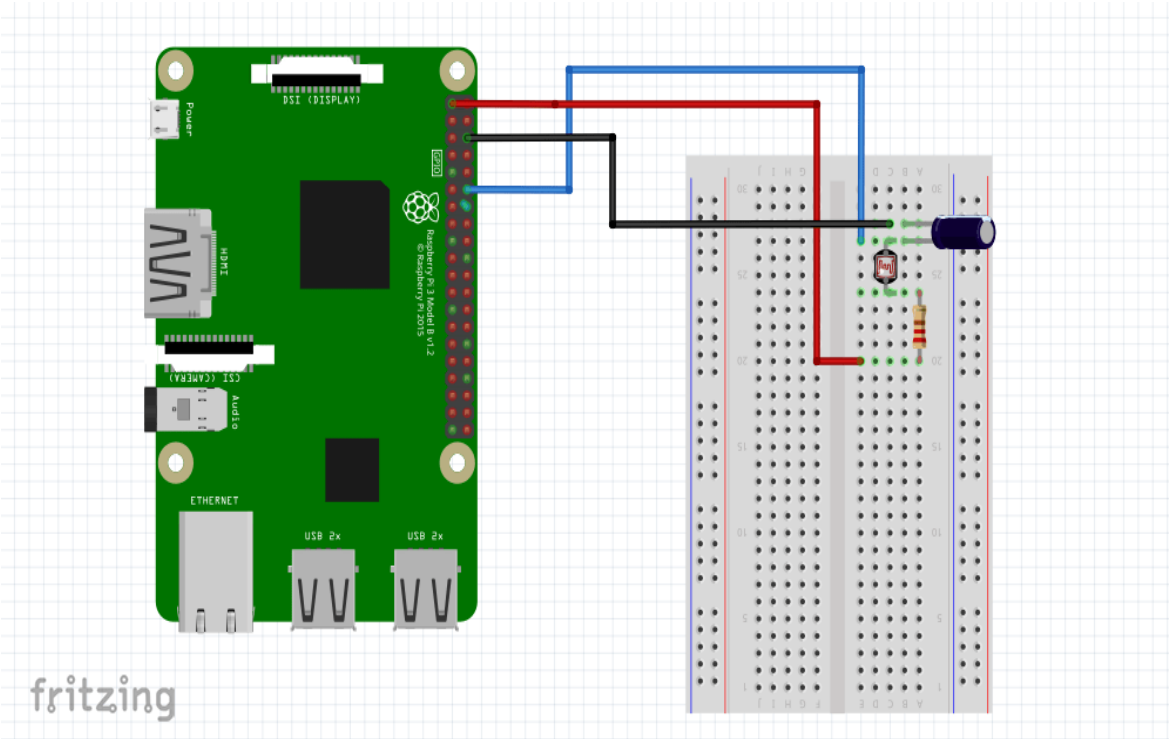
When the switch is pressed the current starts flowing and the capacitor starts charging up. The capacitor stops charging when the voltage at its end reaches the voltage of the battery. Then as there is no potential difference in the upper half of the circuit, no current flows there.
Things needed

- A Raspberry pi
- 1 x breadboard
- A Photocell
- A Resistor
- A Capacitor (1 microfarad)

Circuit:



We need to measure the resistance of the photoresistor. The Raspberry pi acts as the battery whereas the GPIO pin 1 provides 3.3 V to the photoresistor. Make the GPIO pin 12 as the bidirectional pin (input and output pin). **When the capacitor is charging it will take some time to reach a voltage that registers as high.** GPIO pin 6 is grounded which is connected to the negative side of the capacitor (short end). Check how long it takes for the input pin to become high and use the result to calculate the resistance of the photocell.



- Insert a photocell in a breadboard.
- Connect the GPIO pin 1 (3.3 V) to the resistor which is connected serial to the Photocell.
- Connect the other end of the photocell to the GPIO pin 12 and the Capacitor as shown in the diagram.
- GPIO pin 6 (ground) is connected to the other end of the capacitor (short end).

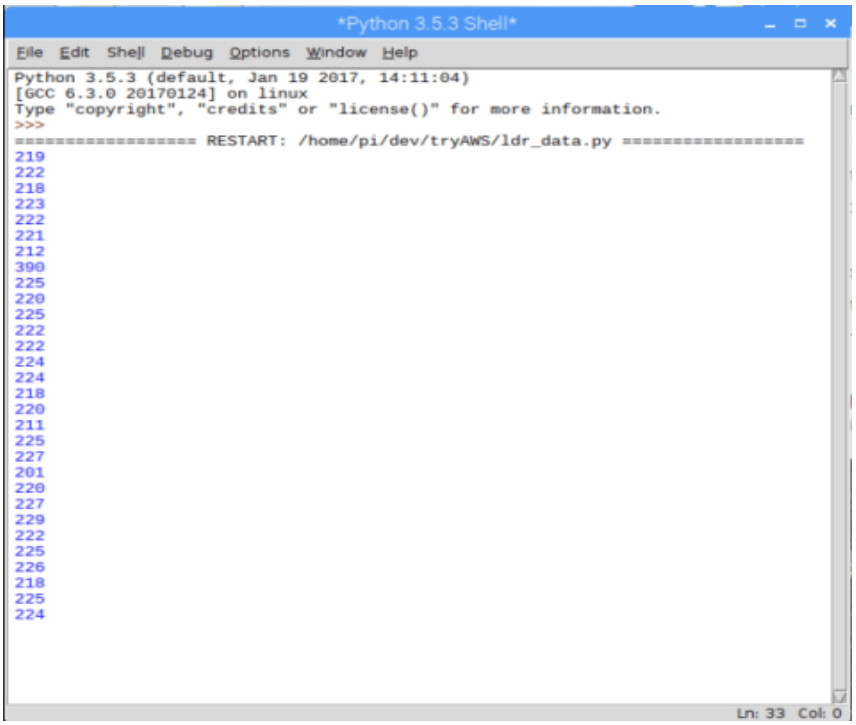
Code

```
#measuring the light intensity using a photocell
 import RPi.GPIO as GPIO,time,os              #import the libraries
 DEBUG=1
 GPIO.setmode(GPIO.BOARD)
 GPIO.setwarnings(False)
 def RCtime(RCpin):  # function start
    reading=0
    GPIO.setup(RCpin,GPIO.OUT)
    GPIO.output(RCpin,GPIO.LOW)
    time.sleep(2)                          # time to discharge capacitor
    GPIO.setup(RCpin,GPIO.IN)
    while (GPIO.input(RCpin) == GPIO.LOW):
  # the loop will run till the capacitor  is charged
      reading += 1
 # measuring time which in turn is measuring resistance
    return reading
# function
```

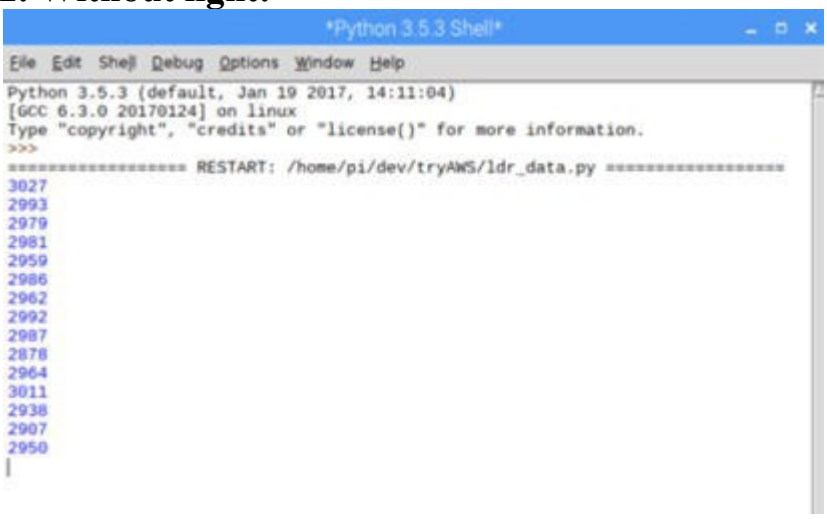
```
while True:
    print RCtime(12)      # calling the function
```

Output

1. **With light:**



2. Without light:



Experiment - 6

Battery Voltage Range Indicator

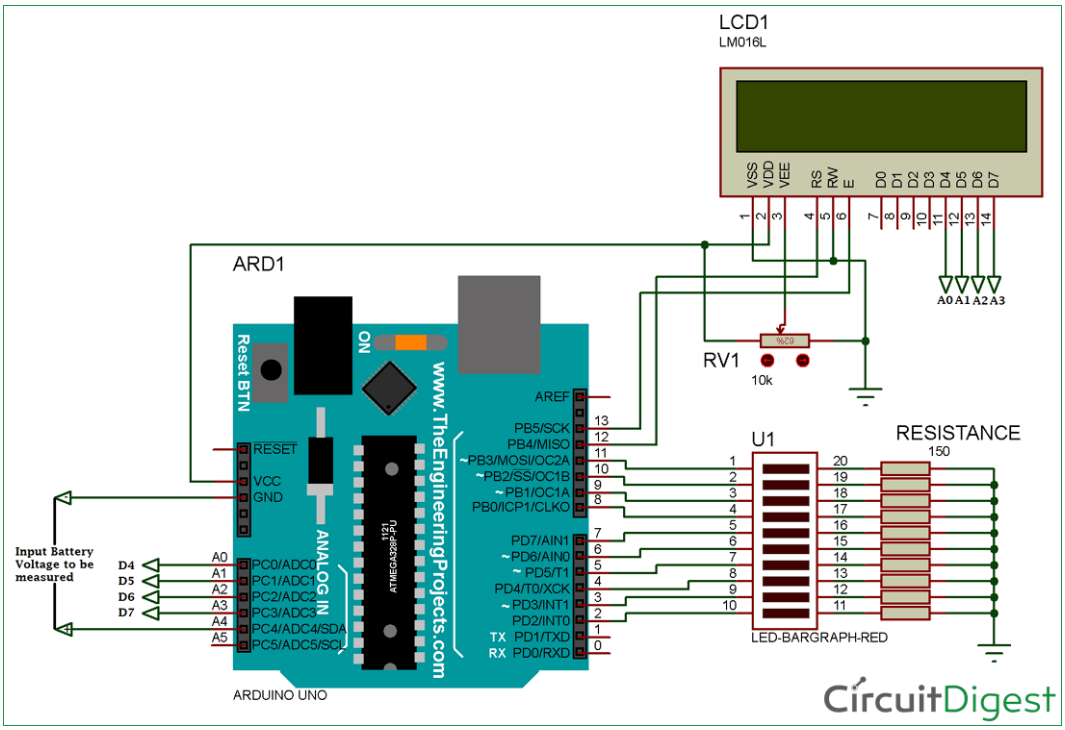
Monitor the voltage level of the battery and indicating the same using multiple LED's (for ex: for 3V battery and 3 led's, turn on 3 led's for 2-3V, 2 led's for 1-2V, 1 led for 0.1-1V & turn off all for 0V)

Battery Voltage Indicator using Arduino and LED Bar Graph

Batteries come with a certain voltage limit and if the voltage goes beyond the prescribed limits while charging or discharging, the life of the battery get affected or reduced. Whenever we use a battery powered project, sometimes we need to check the battery voltage level, whether it is needed to be charged or replaced. This circuit will help you to monitor the voltage of your battery. This **Arduino battery voltage indicator** indicates the status of the battery by glowing LEDs on a **10 Segment LED Bar Graph** according to the battery voltage. It also shows your battery voltage on the LCD connected to the Arduino.

Material Required

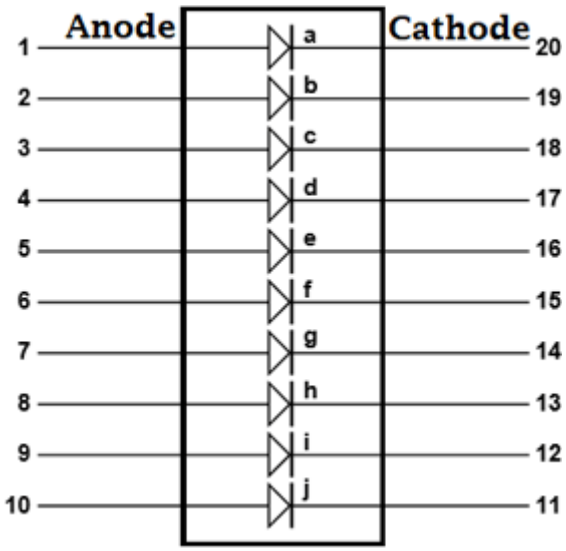
- Arduino UNO
- 10 Segment LED Bar Graph
- LCD (16*2)
- Potentiometer-10k
- Resistor (100ohm-10;330ohm)
- Battery (to be tested)
- Connecting wires
- 12v adapter for Arduino



LED Bar Graph

The LED bar graph comes in industrial standard size with a low power consumption. The bar is categorized for luminous intensity. The product itself remains within RoHS compliant version. It has a forward voltage of up to 2.6v. The power dissipation per segment is 65mW. The operating temperature of the LED bar graph is -40°C to 80°C. There are many application for the LED bar graph like Audio equipment, Instrument panels, and Digital readout display.

Pin Diagram



Pin Configuration

Pin	Function	Pin	Function
1	Anode a	11	Cathode j
2	Anode b	12	Cathode i
3	Anode c	13	Cathode h
4	Anode d	14	Cathode g
5	Anode e	15	Cathode f
6	Anode f	16	Cathode e
7	Anode g	17	Cathode d
8	Anode h	18	Cathode c
9	Anode i	19	Cathode b
10	Anode j	20	Cathode a

Working of Battery Voltage Indicator

Battery Voltage Indicator just read the value from Arduino Analog pin and convert it into a digital value by using the Analog to Digital Conversion (ADC) formula. The **Arduino Uno ADC** is of 10-bit resolution (so the integer values from 0 - 2^10 = 1024 values). This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. So if we multiply input *anlogValue* to (5/1024), then we get the digital value of input voltage. Learn here how to use ADC input in Arduino. Then the digital value is used to glow the LED bar Graph accordingly.

CODE:

```
#include <LiquidCrystal.h>

const int rs = 12, en = 13, d0 = A0, d1 = A1, d2 = A2, d3 = A3;
LiquidCrystal lcd(rs, en, d0, d1, d2, d3);
const int analogPin = A4;
float analogValue;
float input_voltage;

int ledPins[] = {
2, 3, 4, 5, 6, 7, 8, 9, 10, 11
};       // an array of pin numbers to which LEDs are attached
int pinCount = 10;           // the number of pins (i.e. the length of the array)

void setup()
{
  Serial.begin(9600);      //  opens serial port, sets data rate to 9600 bps
  lcd.begin(16, 2);       //// set up the LCD's number of columns and rows:
  pinMode(A0,OUTPUT);
  pinMode(A1,OUTPUT);
```

```
  pinMode(A2,OUTPUT);
  pinMode(A3,OUTPUT);
  pinMode(A4,INPUT);
  lcd.print("Voltage Level");

}
void LED_function(int stage)
{
    for (int j=2; j<=11; j++)
  {
   digitalWrite(j,LOW);
  }
  for (int i=1, l=2; i<=stage; i++,l++)
  {
   digitalWrite(l,HIGH);
   //delay(30);
  }

}
void loop()
{
//  Conversion formula for voltage
  analogValue = analogRead (A4);
  Serial.println(analogValue);
  delay (1000);
  input_voltage = (analogValue * 5.0) / 1024.0;
  lcd.setCursor(0, 1);
  lcd.print("Voltage= ");
  lcd.print(input_voltage);
  Serial.println(input_voltage);
  delay(100);

if (input_voltage < 0.50 && input_voltage >= 0.00 )
{
digitalWrite(2, HIGH);
delay (30);
digitalWrite(2, LOW);
delay (30);
}
else if (input_voltage < 1.00 && input_voltage >= 0.50)
{
LED_function(2);
}
else if (input_voltage < 1.50 && input_voltage >= 1.00)
{
LED_function(3);
}
else if (input_voltage < 2.00 && input_voltage >= 1.50)
{
LED_function(4);
}
else if (input_voltage < 2.50 && input_voltage >= 2.00)
{
```

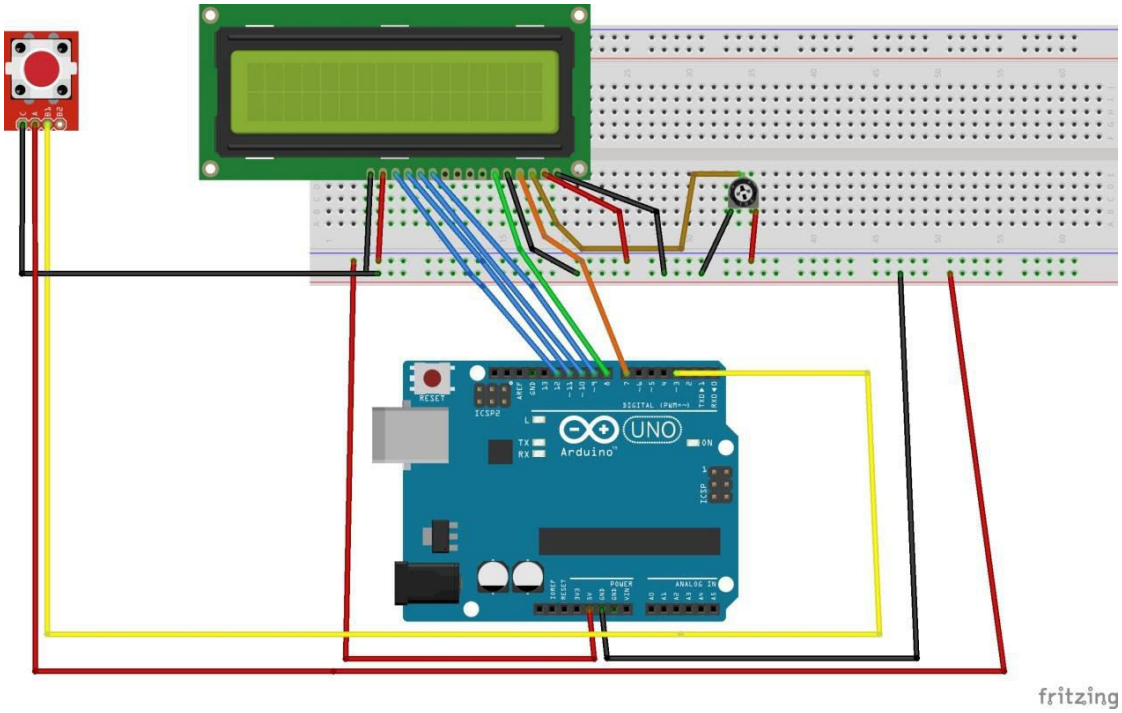
```
LED_function(5);
}
else if (input_voltage < 3.00 && input_voltage >= 2.50)
{
LED_function(6);
}
else if (input_voltage < 3.50 && input_voltage >= 3.00)
{
LED_function(7);
}
else if (input_voltage < 4.00 && input_voltage >= 3.50)
{
LED_function(8);
}
else if (input_voltage < 4.50 && input_voltage >= 4.00)
{
LED_function(9);
}
else if (input_voltage < 5.00 && input_voltage >= 4.50)
{
LED_function(10);
}

}
```

Experiment - 7

Dice Game Simulation

Instead of using the conventional dice, generate a random value similar to dice value and display the same using a 16X2 LCD. A possible extension could be to provide the user with option of selecting single or double dice game.




```
#include <LiquidCrystal.h>
long randNumber;
int Led = 13; //define LED port
int Shock = 2; //define shock port
int val;//define digital variable val
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(7, 8, 9, 10, 11, 12 );
byte customChar[] = {
  B00000,
  B00000,
  B11111,
  B11001,
  B10101,
  B10011,
```

```
  B11111,
  B00000
};
void  setup()
{

  lcd.begin(16, 2);
  lcd.createChar(0, customChar);
  lcd.home();
  pinMode(Led, OUTPUT); //define LED as a output port
  randomSeed(analogRead(0));
  pinMode(Shock, INPUT); //define shock sensor as a output port
  lcd.write(byte( 0));
  lcd.print("Digital dice");
  lcd.write(byte( 0));
  delay(1000);
}

void  loop()
{

  val = digitalRead(Shock); //read the value of the digital interface 3 assigned to val
  if (val == LOW) //when the shock sensor have signal do the following
  {
    lcd.clear();
    lcd.print("Rolling dice...");
    delay(4000);
    lcd.clear();
    lcd.setCursor(0, 0);
    randNumber = random(1,7);
    lcd.print("Dice 1 = ");
    lcd.print(randNumber);

    lcd.setCursor(0, 1);
    randNumber = random(1,7);
```

```
    lcd.print("Dice 2 = ");
    lcd.print(randNumber);


  }

 delay(150);
}
```

Experiment - 8

Displaying RSS News Feed On Display Interface

Displaying the RSS news feed headlines on a LCD display connected to device. This can be adapted to other websites like twitter or other information websites. Python can be used to acquire data from the internet.

Keeping up to date with the latest news is tough and sometimes we need a little help. RSS feeds provide a great way to quickly digest lots of news quickly. Sure you could visit an RSS feed or have an RSS reader on your computer, but what if you could have a simple, dedicated device that only shows the headlines?

Here's a Raspberry Pi project that will use Python code to read an RSS feed, the Tom's Hardware feed for example, and display the top five headlines on an LCD screen.

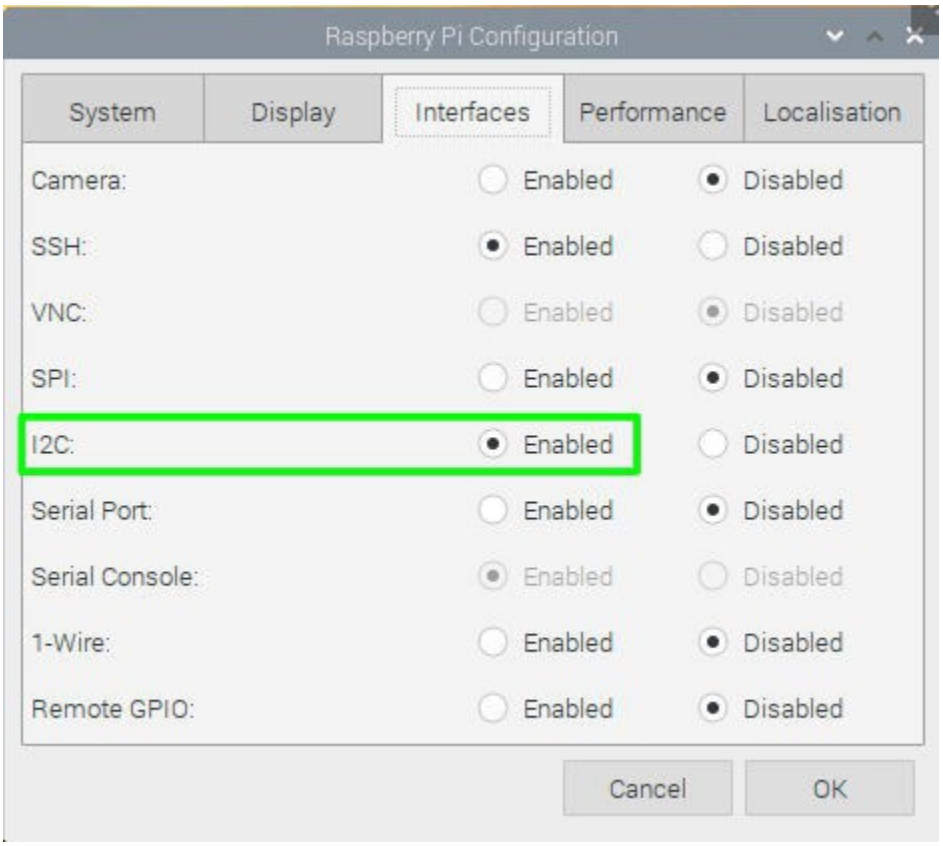
To build this project you will need:

- Any model of Raspberry Pi with Raspberry Pi OS and GPIO Pins
- An I2C LCD screen such as this one
- 4 x Female to female jumper wires

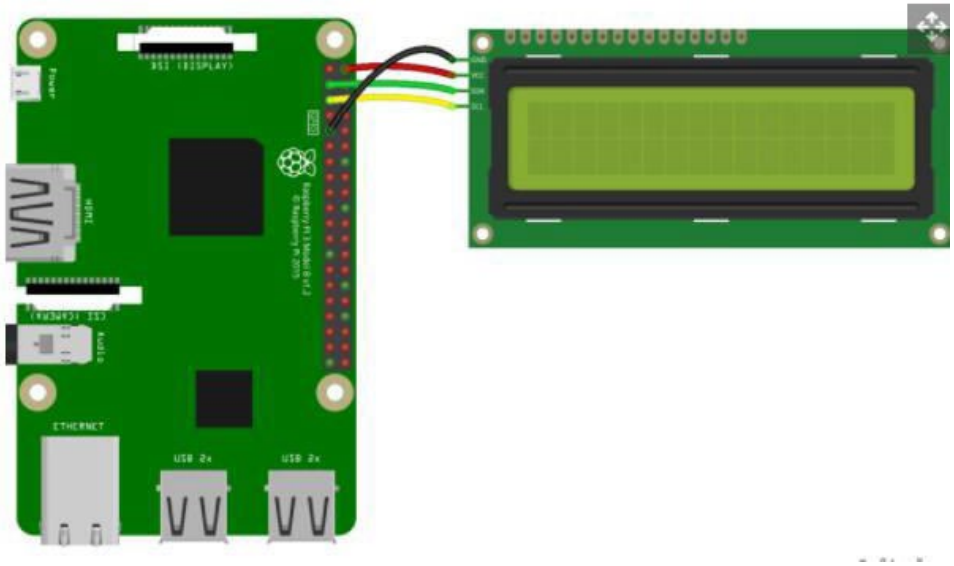
1. **Install Python libraries** to use LCD screens and work with RSS feeds by entering the following commands:

```
sudo pip3 install rpi-lcd feedparser
```

2. **Enable the I2C interface** via Preferences >> Raspberry Pi Configuration



3. Connect the I2C LCD screen as per the diagram.



4. **Launch Thonny.** You can find it on the start menu under Programming.

5. **In a new file import libraries of Python code** to use the LCD screen, control the pace of the project, read the RSS feed and finally manipulate text into chunks.

```
from rpi_lcd import LCD
from time import sleep
import feedparser
import textwrap
```

6. **Create an object, called "tom"** which will store the RSS feed data from Tom's Hardware.

```
tom = feedparser.parse("https://www.tomshardware.com/uk/feeds/all")
```

7. **Create a connection to the LCD** and then pause the code for 1 second.

```
lcd = LCD()
sleep(1)
```

8. **Use a for loop** to repeat code five times. If you want more than 5 headlines, change the (5) to a higher number.

```
for i in range(5):
```

9. **Print the entries from the Tom's Hardware RSS feed.** The value of i is incremented each time the for loop goes round, to a maximum of five.

```
    print(tom['entries'][i]['title'])
```

10. **Create an object called split** and use that to save 16 character chunks of the RSS feed. The chunk size is set by the 16 character screen size of the LCD.

```
split = textwrap.wrap(text, 16)
```

11. **Create an object called split** and use that to save 16 character chunks of the RSS feed. The chunk size is set by the 16 character screen size of the LCD.

```
split = textwrap.wrap(text, 16)
```

12. **Print "Tom's Hardware"** (or the name of your news source) **to the first line of the LCD screen.**

```
lcd.text("Tom's Hardware", 1)
```

13. **Create another for loop** to print the contents of the split object to the LCD screen.

```
    for i in range(len(split)):
```

```
            lcd.text(split[i], 2)
            sleep(0.5)
```

14. **Add a one second pause before clearing the LCD screen.**

```
    sleep(1)
lcd.clear()
```

15. **Save the code as TomsRSSFeed.py**

Code:

```
from rpi_lcd import LCD
from time import sleep
import feedparser
import textwrap
tom = feedparser.parse("https://www.tomshardware.com/uk/feeds/all")

lcd = LCD()
sleep(5)
for i in range(5):
            print(tom['entries'][i]['title'])
            text = tom['entries'][i]['title']
            split = textwrap.wrap(text, 16)
            lcd.text("Tom's Hardware", 1)
            for i in range(len(split)):
            lcd.text(split[i], 2)
            sleep(0.5)
            sleep(1)
lcd.clear()
```

Experiment - 9

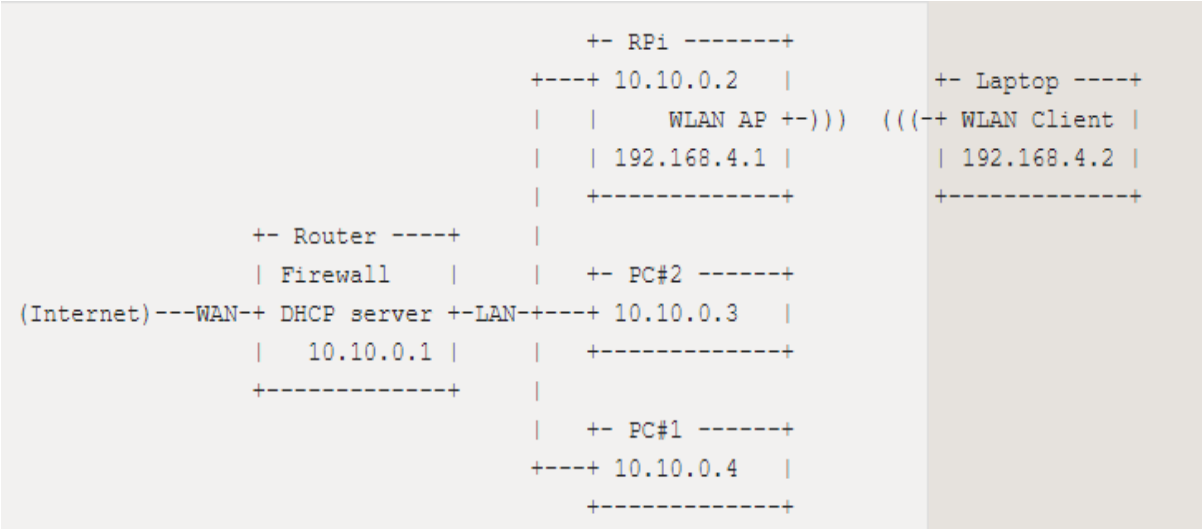
Porting Openwrt

To the Device Attempt to use the device while connecting to a wifi network using a USB dongle and at the same time providing a wireless access point to the dongle.

Setting up a Raspberry Pi as 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 we start

- 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 the access point and network 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.

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:

```
# https://www.raspberrypi.org/documentation/configuration/wireless/access-point-routed.md
# 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 access point 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.

Run your new wireless access point

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!

Experiment - 10

Hosting a website on Board

Building and hosting a simple website(static/dynamic) on the device and make it accessible online. There is a need to install server(eg: Apache) and thereby host the website.

Setting up an Apache Web Server on a Raspberry Pi

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, and with additional modules can serve dynamic web pages using scripting languages such as PHP.

Install Apache

First, update the available packages by typing the following command into the Terminal:

```
sudo apt update
```

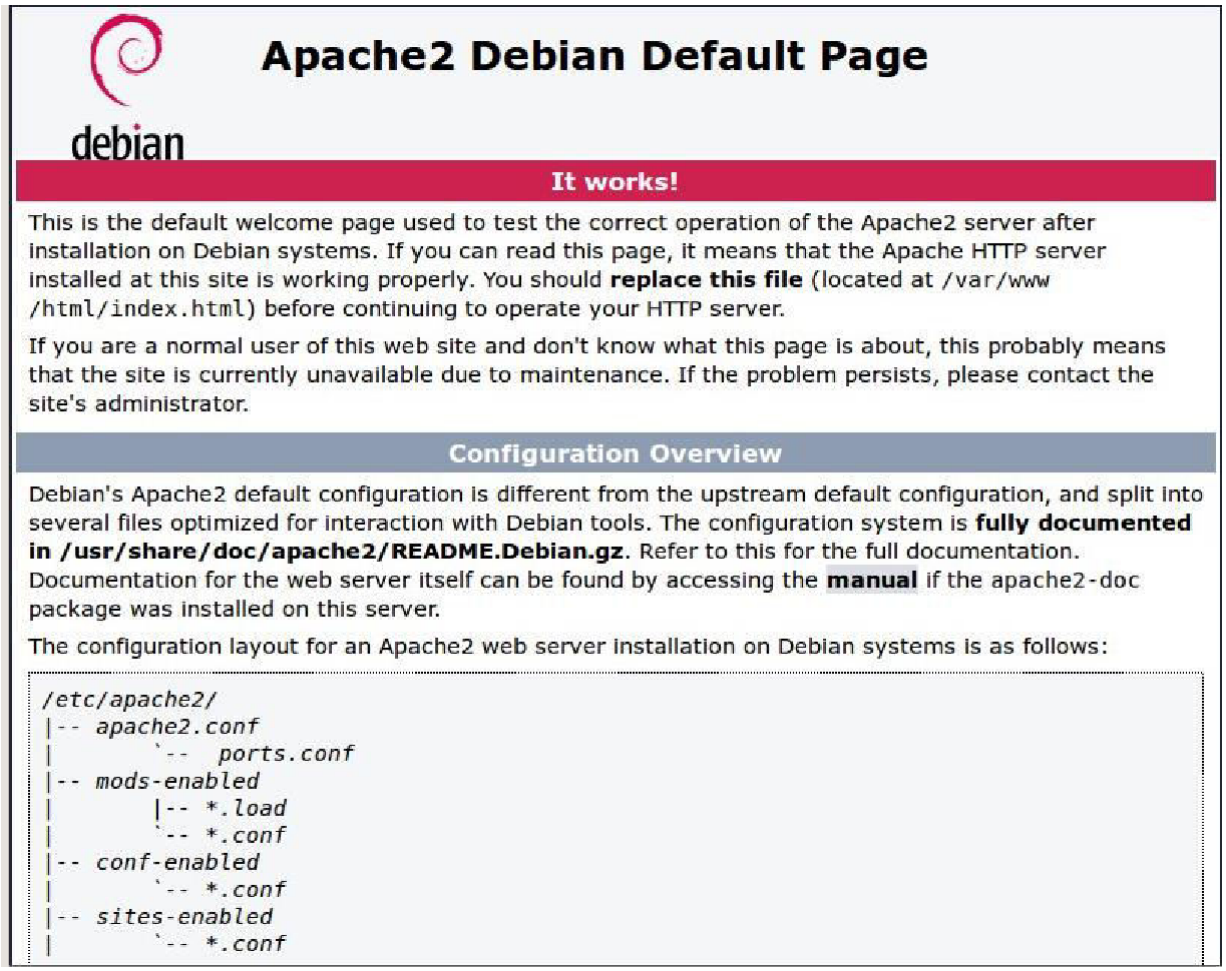
Then, install the apache2 package with this command:

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

Test the web server

By default, Apache puts a test HTML file in the web folder. This default web page is served when you browse to `http://localhost/` on the Pi itself, or `http://192.168.1.10` (whatever the Pi's IP address is) from another computer on the network. To find the Pi's IP address, type `hostname -I` at the command line (or read more about finding your IP address).

Browse to the default web page either on the Pi or from another computer on the network and you should see the following:



This means you have Apache working!

Changing the default web page

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

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

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

This will show you:

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

This shows that by default there is one file in `/var/www/html/` called `index.html` and it is owned by the `root` user (as is the enclosing folder). In order to edit the file, you need to change its ownership to your own username. Change the owner of the file (the default `pi` user is assumed here) using `sudo chown pi: index.html` .

You can now try editing this file and then refreshing the browser to see the web page change.
Your own website
If you know HTML you can put your own HTML files and other assets in this directory and serve them as a website on your local network.

Additional - install PHP

To allow your Apache server to process PHP files, you'll need to install the latest version of PHP and the PHP module for Apache. Type the following command to install these:

```
sudo apt install php libapache2-mod-php -y
```

Now remove the `index.html` file:

```
sudo rm index.html
```

and create the file `index.php` :

```
sudo nano index.php
```

Put some PHP content in it:

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

Now save and refresh your browser. You should see "hello world". This is not dynamic but still served by PHP. Try something dynamic:

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

or show your PHP info:

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

Experiment – 11

Webcam Server

Interfacing the regular usb webcam with the device and turn it into fully functional IP webcam & test the functionality.

Using a standard USB webcam

Rather than using the Raspberry Pi camera module, you can use a standard USB webcam to take pictures and video on the Raspberry Pi.

Note that the quality and configurability of the camera module is highly superior to a standard USB webcam.

Install fswebcam

First, install the `fswebcam` package:

```
sudo apt install fswebcam
```

Add your user to `video` group

If you are not using the default `pi` user account, you need to add your username to the `video` group, otherwise you will see 'permission denied' errors.

```
sudo usermod -a -G video <username>
```

To check that the user has been added to the group correctly, use the `groups` command.

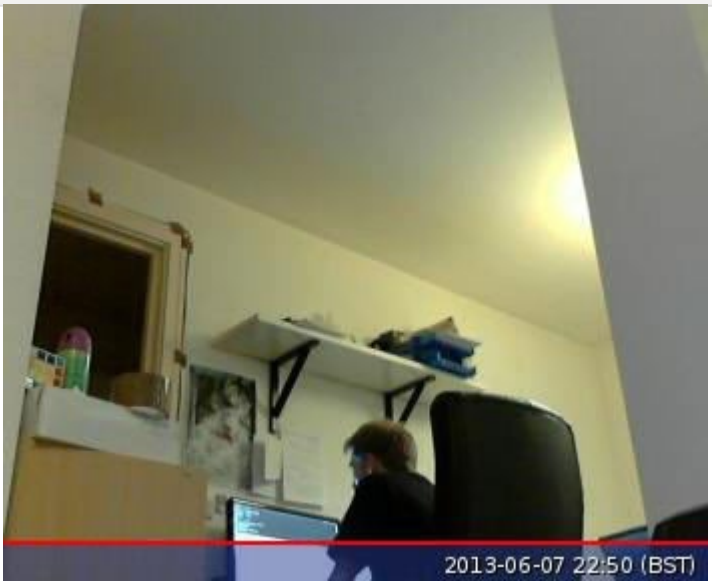
Basic usage

Enter the command `fswebcam` followed by a filename and a picture will be taken using the webcam, and saved to the filename specified:

```
fswebcam image.jpg
```

This command will show the following information:

```
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Corrupt JPEG data: 2 extraneous bytes before marker 0xd4
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to 'image.jpg'.
```



Note the small default resolution used, and the presence of a banner showing the timestamp.

Specify resolution

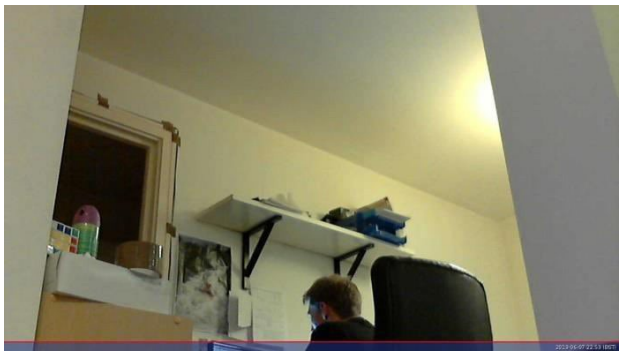
The webcam used in this example has a resolution of `1280 x 720` so to specify the resolution I want the image to be taken at, use the `-r` flag:

```
fswebcam -r 1280x720 image2.jpg
```

This command will show the following information:

```
--- Opening /dev/video0...
```

```
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
--- Capturing frame...
Corrupt JPEG data: 1 extraneous bytes before marker 0xd5
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to 'image2.jpg'.
```



Picture now taken at the full resolution of the webcam, with the banner present.

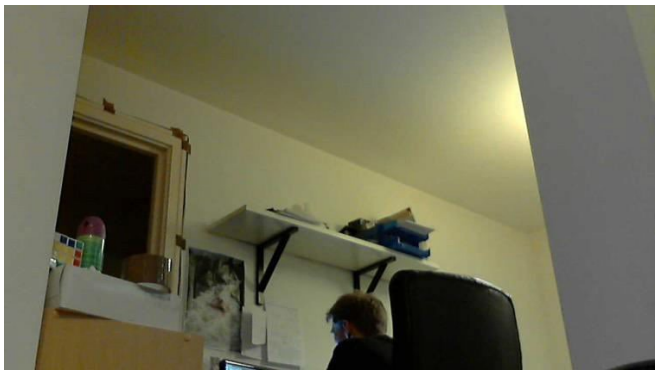
Specify no banner

Now add the `--no-banner` flag:

```
fswebcam -r 1280x720 --no-banner image3.jpg
```

which shows the following information:

```
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
--- Capturing frame...
Corrupt JPEG data: 2 extraneous bytes before marker 0xd6
Captured frame in 0.00 seconds.
--- Processing captured image...
Disabling banner.
Writing JPEG image to 'image3.jpg'.
```



Now the picture is taken at full resolution with no banner.

Bad Pictures

You may experience poor quality pictures with a USB webcam, such as this accidentally artistic piece:

Some webcams are more reliable than others, but this sort of issue may occur with poor quality webcams. If the problem persists, ensure your system is up to date. Also try other webcams, but you'll get the best performance from the Raspberry Pi camera module.

Bash script

You can write a Bash script which takes a picture with the webcam. The script below saves the images in the `/home/pi/webcam` directory, so create the `webcam` subdirectory first with:

```
mkdir webcam
```

To create a script, open up your editor of choice and write the following example code:

```
#!/bin/bash
DATE=$(date +"%Y-%m-%d_%H%M")
fswebcam -r 1280x720 --no-banner /home/pi/webcam/$DATE.jpg
```

This script will take a picture and name the file with a timestamp. Say we saved it as `webcam.sh`, we would first make the file executable:

```
chmod +x webcam.sh
```

Then run with:

```
./webcam.sh
```

Which would run the commands in the file and give the usual output:

```
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
--- Capturing frame...
Corrupt JPEG data: 2 extraneous bytes before marker 0xd6
Captured frame in 0.00 seconds.
--- Processing captured image...
```

```
Disabling banner.
Writing JPEG image to '/home/pi/webcam/2013-06-07_2338.jpg'.
```

Time-lapse using cron

You can use `cron` to schedule taking a picture at a given interval, such as every minute to capture a time-lapse.

First open the cron table for editing:

```
crontab -e
```

This will either ask which editor you would like to use, or open in your default editor. Once you have the file open in an editor, add the following line to schedule taking a picture every minute (referring to the Bash script from above):

```
* * * * * /home/pi/webcam.sh 2>&1
```

Save and exit and you should see the message:

```
crontab: installing new crontab
```

Ensure your script does not save each picture taken with the same filename. This will overwrite the picture each time.