**American International University- Bangladesh**
**Department of Electrical and Electronic Engineering**
EEE4212: Microprocessor and I/O Systems Laboratory

**Title:** Familiarization with Raspberry Pi

## Introduction:
The objective of this experiment is to get familiarized with Raspberry Pi.

## Theory and Methodology:
The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The most important thing about a Raspberry Pi is that, it is computer that costs $35 or less.

### Overview
Several generations of Raspberry Pis have been released. The first generation (**Raspberry Pi 1 - Model B**) was released in February 2012, followed by the simpler and cheaper **Model A.** The third generation/latest version, **Raspberry Pi 3 - Model B** was released in February 2016 and has on-board WiFi, Bluetooth and USB boot capabilities. All models feature a Broadcom system on a chip (SoC) with an integrated ARM compatible central processing unit (CPU) and on-chip graphics processing unit (GPU).
Processor speed ranges from 700 MHz to 1.2 GHz for the Pi 3; on-board memory ranges from 256 MB to 1 GB RAM. Secure Digital (SD) cards are used to store the operating system and program memory in either SDHC or Micro-SDHC sizes. The boards have one to four USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm phono jack for audio output. Lower-level output is provided by a number of GPIO pins which support common protocols like I²C. The B-models have an 8P8C Ethernet port and the Pi 3 and Pi Zero W have on-board Wi-Fi 802.11n and Bluetooth.

### Hardware
The Raspberry Pi hardware has evolved through several versions that feature variations in memory capacity and peripheral-device support. The Raspberry pi is classified into three families including Model B, Model A, Zero and Compute.

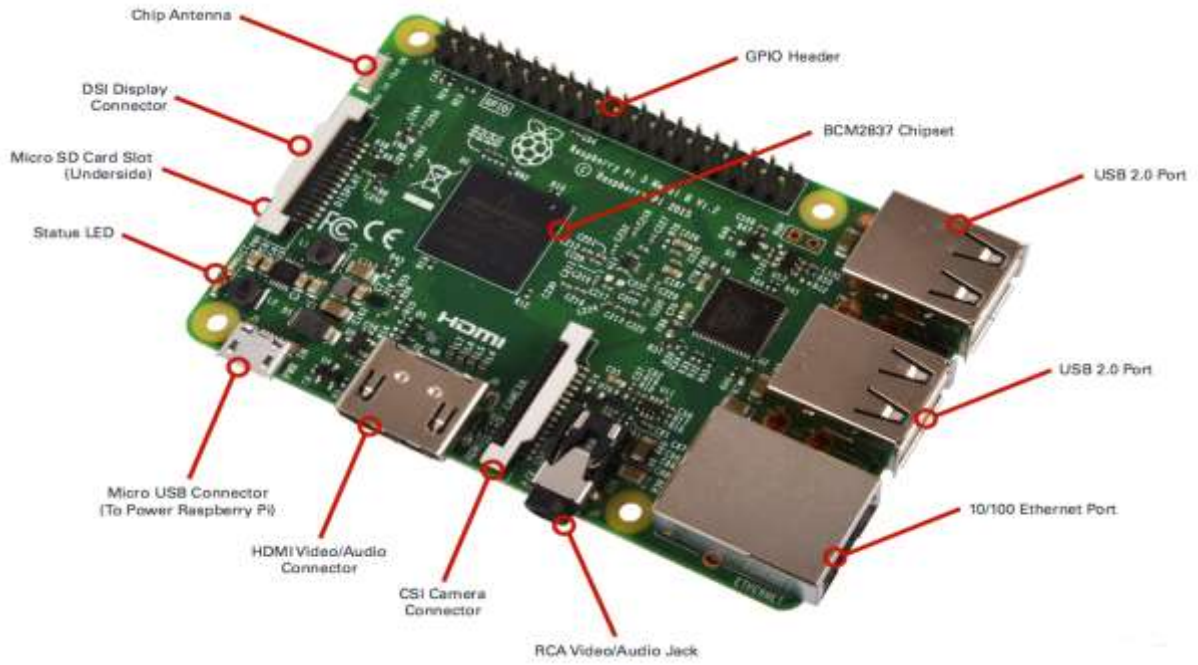


Figure 1: Raspberry Pi 3 - Model B

1. **Model B:** The Raspberry Pi Model B is the flagship Raspberry Pi Model, with the fastest CPU and most connectivity features. Model B family of Raspberry pi has several versions including Raspberry Pi Model B, Raspberry Pi Model B+, Raspberry Pi 2 - Model B, Raspberry Pi 2 - Model B v1.2 and Raspberry pi 3 - Model B. The latest revision (Raspberry Pi 3 -Model B) features a 1.2GHz Quad Core processor, and on board WiFi & Bluetooth as shown in Fig. 1. The Model B is the largest Pi available with the approximate size of a credit card.
2. **Model A:** The Raspberry Pi Model A is a cut down version of the Model B as shown in Fig. 2, featuring a smaller form factor and only one USB port. The Model A's do not include ethernet, and are designed to be lower cost, lower power consumption alternative to the Model B. Family of Model A has two versions including Raspberry Pi Model A and Raspberry Pi Model A+.

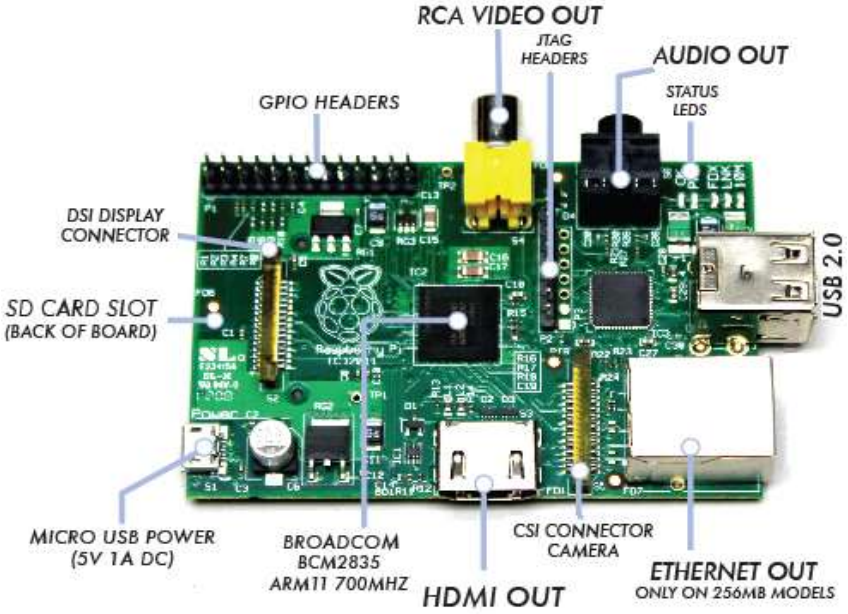


Figure 2: Raspberry Model A

3. **Zero:** The Raspberry Pi Zero is an ultra-low cost and ultra-small variant of the Raspberry Pi. It comes with an unpopulated GPIO header, and only a single micro USB port for connectivity as shown in Fig.3. The Raspberry Pi Zero W (Wireless) includes on board WiFi and Bluetooth. This family of Raspberry pi has three versions including Raspberry Pi Zero v1.2, Raspberry Pi Zero v1.3 and Raspberry Pi Zero Wireless.
4. **Compute:** The Compute Module features a 200 Pin DDR2-SODIMM layout and is designed for engineers who need Raspberry Pi functionality but need a smaller form factor for embedded designs. The Compute family has three different versions and those are: Compute Module, Compute Module 3 and Compute Module 3 Lite. Raspberry pi compute module 3 is shown in Fig.4.

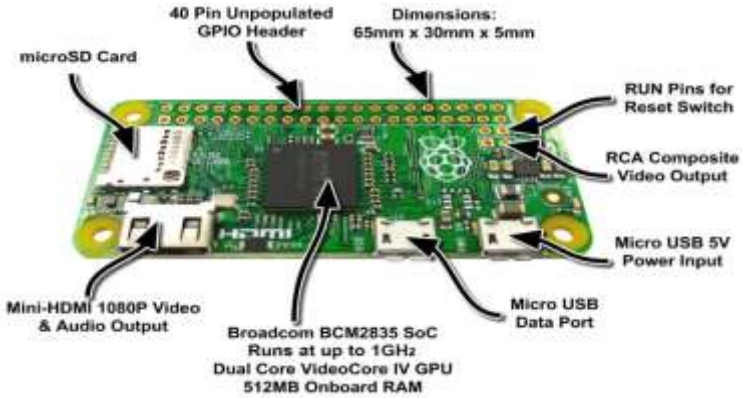


Figure 3: Raspberry Pi Zero

In this laboratory latest version, **Raspberry Pi 3 - Model B** will be used. Thus, only hardware specification of Raspberry Pi 3 – Model B is discussed.

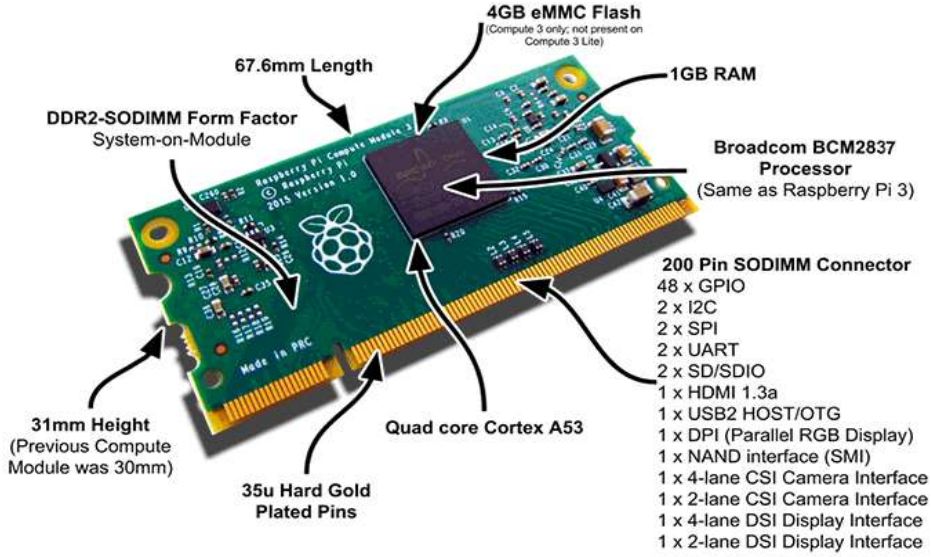


Figure 4: Raspberry Pi Compute Module 3

**Technical Specification**

1. **Processor**
   - Broadcom BCM2387 chipset.
   - 64-bit 1.2GHz Quad-Core ARM Cortex-A53.
2. **802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)**
   - IEEE 802.11 b / g / n Wi-Fi. Protocol: WEP, WPA WPA2, algorithms AES-CCMP (maximum key length of 256 bits), the maximum range of 100 meters.
   - IEEE 802.15 Bluetooth, symmetric encryption algorithm Advanced Encryption Standard (AES)with 128-bit key, the maximum range of 50 meters.
3. **GPU**
   - Dual Core Video Core IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated Open VG, and 1080p30 H.264 high-profile decode.
   - Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure.
4. **Memory**
   - 1GB LPDDR2.
5. **Operating System**
   - Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT
6. **Dimensions**
   - 85 x 56 x 17mm
7. **Power**
   - Micro USB socket 5V1, 2.5A
8. **Ethernet**
   - 10/100 Base T Ethernet socket.
9. **Video Output**
   - HDMI (rev 1.3 & 1.4)
   - Composite RCA (PAL and NTSC)
10. **Audio Output**
    - Audio Output 3.5mm jack
    - HDMI
    - USB 4 x USB 2.0 Connector
11. **GPIO Connector**

- 40-pin 2.54 mm (100 mil) expansion header: 2x20 strip
- Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines


Figure 5: Raspberry Pi 3 - Model B GPIO pin

12. **Camera Connector**
    - 15-pin MIPI Camera Serial Interface (CSI-2)
13. **Display Connector**
    - Display Serial Interface (DSI) 15-way flat flex cable connector with two data lanes and a clock lane
14. **USB**
    - Four built-in USB ports provide enough connectivity for a mouse, keyboard, or anything else that you feel the RPi needs.
15. **Antenna**
    - There's no need to connect an external antenna to the Raspberry Pi 3. Its radios are connected to this chip antenna soldered directly to the board
16. **HDMI connector:**
    - The HDMI port provides digital video and audio output.14 different video resolutions are supported, and the HDMI signal can be converted to DVI (used by many monitors), composite (analog video signal usually carried over a yellow RCA connector), or SCART (a European standard for connecting audio-visual equipment) with external adapters.
17. **Status LED**

    There are five status LEDs on the corner of the board.

| | | |
|---|---|---|
| ACT | Green | Lights when the SD card is accessed/used |
| PWR | Red | steady ON when Pi is connected to 3.3V power |
| FDX | Green | On if network adapter is full duplex |
| LNK | Green | Network activity light On when Ethernet is connected |
| 100 | Yellow | On if the network connection is 100Mbps |

The status LEDs give information about the operating condition and any problems of the board e.g.

| Status LED | Possible Problem |
|---|---|
| Red power LED does not light, nothing on display | The power is not properly connected |
| The LED light is blinking | The red power LED should never blink. Blinking RED LED (PWR) means the 5V power supply is dropping out. Use a different power supply. |
| Red power LED is on, green LED is glowing faintly and steadily | Power supply is OK. But faint and steady green light (ACT) means SD card has some problem in starting the operating system (no boot code). |

**<u>Setting Up the Raspberry Pi:</u>**

**What You Will Need?**

1. Raspberry Pi
2. Monitor
3. Display and Connectivity Cable (HDMI/DVI)

4. Keyboard and Mouse
5. Power Supply (good-quality power supply that can supply at least 2A at 5V for the Model 3B)
6. SD Card (minimum 4 GB)
7. Ethernet (network) cable (Optional)
8. Audio Lead (Without an HDMI Cable an audio lead is necessary to produce sound)

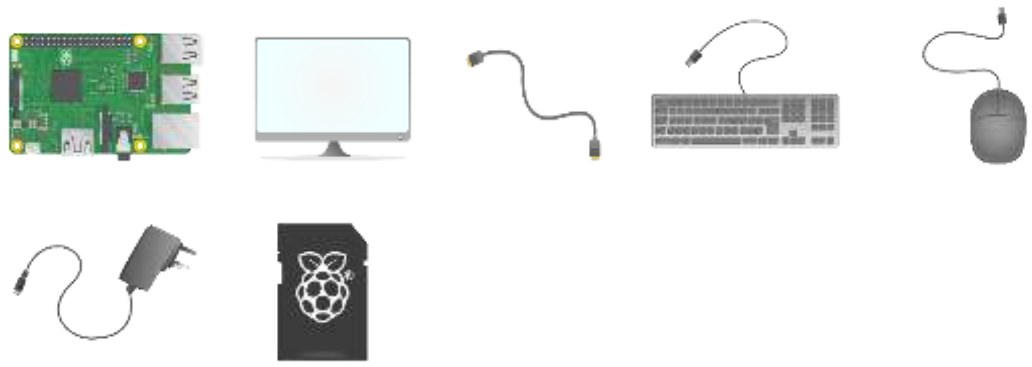

Figure 6: Apparatus for Setting Up Raspberry Pi

**Operating Systems**

When setting up Raspberry Pi, always keep in mind that you are actually setting up a small PC. Like a PC set up Pi must be set up with Operating System (OS). **Raspbian** is the official operating system for all models of the Raspberry Pi. However, there are third party operating systems like Ubuntu Mate, Snappy Ubuntu Core, Windows 10 IOT core, OSMC, Libreelec, Pinet and RISC OS. Logos of different OS for Pi is shown in following Fig. 7.


Figure 7: Logos of Different Operating Systems

**Getting an operating system**

The recommended operating system for use with the Raspberry Pi is called Raspbian. Raspbian is a version of GNU/Linux, designed specifically to work well with the Raspberry Pi. There are several options when it comes to getting hold of a copy of Raspbian.

1. **Buy a pre-installed SD card:** The easiest way to get NOOBS or Raspbian is to buy an SD card with the software already installed. You can get a pre-installed Raspbian card from RS or The Pi Hut.
2. I**nstall Raspbian with NOOBS:** NOOBS stand for New Out Of Box Software, and if you've never played around with GNU/Linux before, then it's the best place to start. To begin with, it's always a good idea to make sure you have formatted your SD card. However, in this laboratory we are not going to use NOOBS for installing OS. Thus, the process of installing OS with NOOBS is not discussed.
3. **Download and image Raspbian directly:** An alternative of using NOOBS, to install Raspbian is to download and install the image directly. This is a faster process and is great if you need to image multiple cards for a workshop or class. The steps are given bellow:
    I. Using a computer with an SD card reader, visit the official Raspberry Pi Downloads page. Link: https://www.raspberrypi.org/downloads/

II. Download page will look like following Fig.8. Click on Raspbian

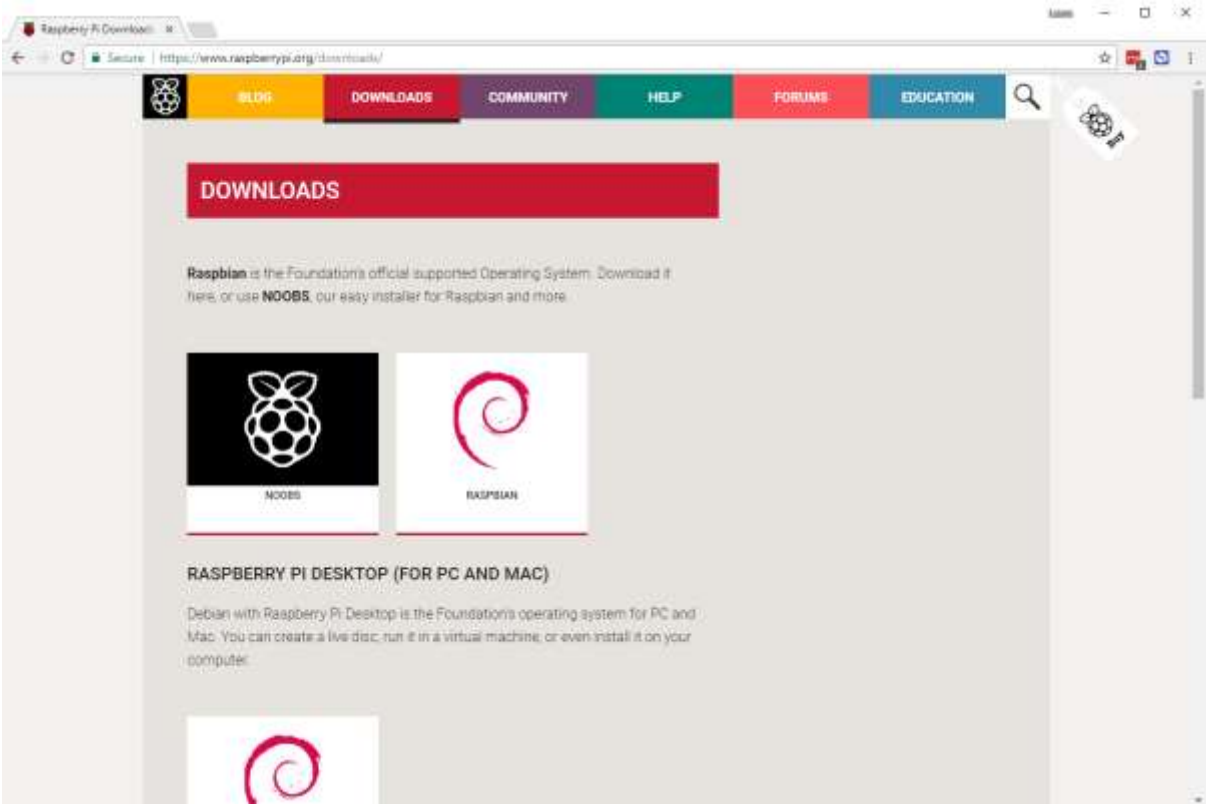


Figure 8: Download Page

III. Click on the **Download ZIP** button under 'Raspbian Stretch with desktop' and select a folder to save it to.


Figure 9: Raspbian Stretch with Desktop

IV. Extract the files from the zip.
V. Visit etcher.io and download and install the Etcher SD card image utility.
VI. Run Etcher and select the Raspbian image you unzipped on your computer or laptop.
VII. Select the SD card drive. Note that the software may have already selected the right drive.
VIII. Finally, click Burn to transfer Raspbian to the SD card. You'll see a progress bar that tells you how much is left to do. Once complete, the utility will automatically eject/unmount the SD card so it's safe to remove it from the computer.

**Plugging in your Raspberry Pi**

Now you have an operating system, you can slot your SD card into your Raspberry Pi and connect the power.

1. Begin by placing your SD card into the SD card slot on the Raspberry Pi. It will only fit one way.
2. Next, plug your keyboard and mouse into the USB ports on the Raspberry Pi.
3. Make sure that your monitor or TV is turned on, and that you have selected the right input (e.g. HDMI 1, DVI, etc).
4. Connect your HDMI cable from your Raspberry Pi to your monitor or TV.

5. If you intend to connect your Raspberry Pi to the internet, plug an Ethernet cable into the Ethernet port, or connect a WiFi dongle to one of the USB ports (unless you have a Raspberry Pi 3).
6. When you're happy that you have plugged all the cables and SD card in correctly, connect the micro USB power supply. This action will turn on and boot your Raspberry Pi.

**Power on your Raspberry Pi for the first time**

Remember that after booting the Pi, there might be situations when the user credentials like the "username" and password will be asked. Raspberry Pi comes with a default user name and password and so always use it whenever it is being asked. The credentials are:

```
login: pi

password: raspberry
```

When the Pi has been booted for the first time, a configuration screen called the "Setup Options" should appear and it will look like the Fig.10 below.

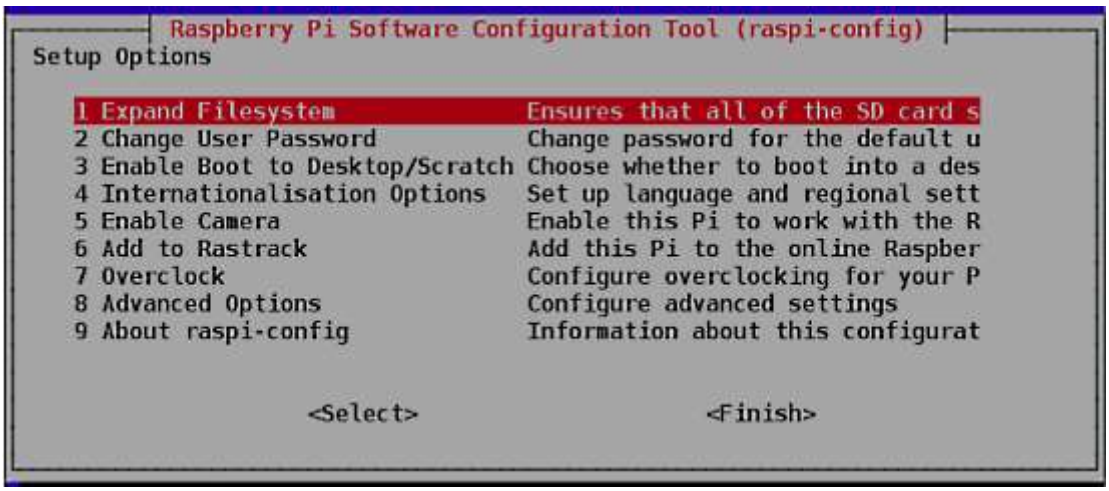


Figure 10 : Raspberry Pi Configuration

If you have missed the "Setup Options" screen, it's not a problem, you can always get it by typing the following command in the terminal.

```
sudo raspi-config
```

Once you execute this command the "Setup Options" screen will come up as shown in the Fig. 10. Now that the Setup Options window is up, we will have to set a few things. After completing each of the steps below, if it asks to reboot the Pi, please do so. After the reboot, if you don't get the "Setup Options" screen, then follow the command given above to get the screen/window.

- **The First thing to do**
  Select the first option in the list of the setup options window, that is select the "Expand Filesystem" option and hit the enter key. We do this to make use of all the space present on the SD card as a full partition. All this does is, expand the OS to fit the whole space on the SD card which can then be used as the storage memory for the Pi.
- **The second thing to do**
  Select the third option in the list of the setup options window, that is select the "Enable Boot to Desktop/Scratch" option and hit the enter key. It will take you to another window called the "choose boot option" window that looks like the Fig.11 below.

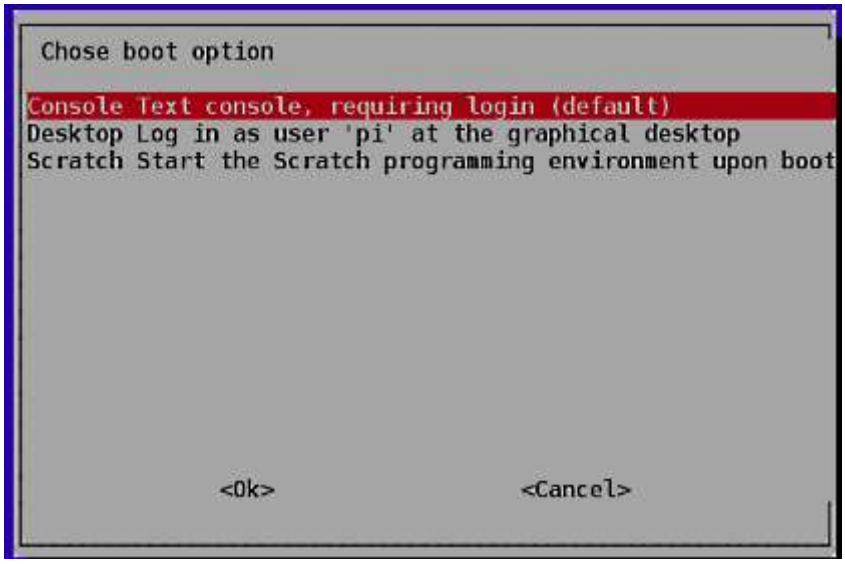


Figure 11 : Boot option

In the "choose boot option window", select the second option, that is, "Desktop Log in as user 'pi' at the graphical desktop" and hit the enter button. Once done, you will be taken back to the "Setup Options" page, if not select the "OK" button at the bottom of this window and you will be taken back to the previous window. We do this because we want to boot into the desktop environment which we are familiar with. If we don't do this step, then the Raspberry Pi boots into a terminal each time with no GUI options.
Once, both the steps are done, select the "finish" button at the bottom of the page and it should reboot automatically. If it doesn't, then use the following command in the terminal to reboot.

| *sudo reboot* |
|---|

After the reboot from the previous step, if everything went right, then you will end up on the desktop which looks like the image below.

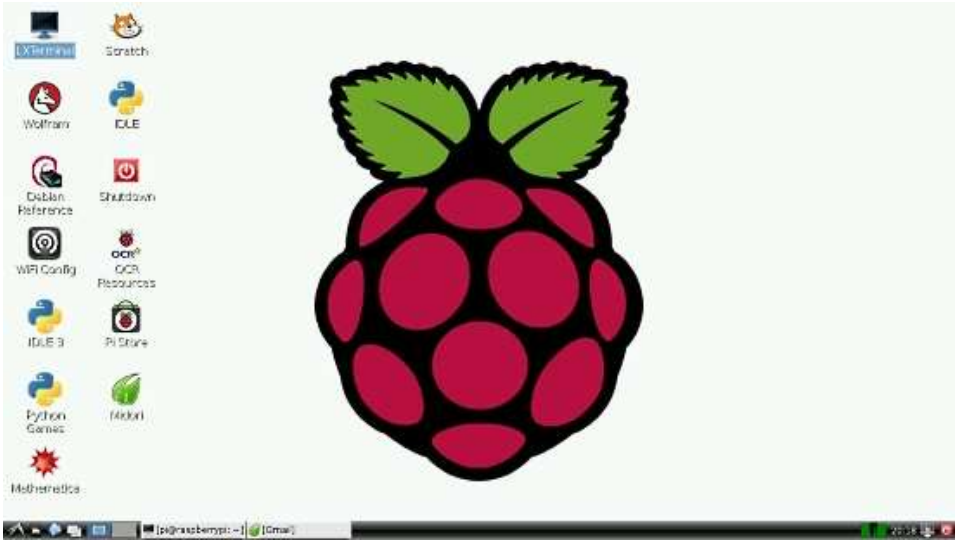


Figure 12: Linux GUI

**Familiarization with the Linux Commands**
In this section, we will learn about the Linux Operating system and some common commands.
Here are some fundamental and common Linux commands with example usage:
**Filesystem**
***LS*:** The **ls** command lists the content of the current directory (or one that is specified). It can be used with the **-l** flag to display additional information (permissions, owner, group, size, date and timestamp

of last edit) about each file and directory in a list format. The **-a** flag allows you to view files beginning with . (i.e. dotfiles).

***CD*:** Using **cd** changes the current directory to the one specified. You can use relative (i.e. **cd directoryA**) or absolute (i.e. **cd /home/pi/directoryA**) paths.

***PWD*:** The **pwd** command displays the name of the present working directory: on a Raspberry Pi, entering **pwd** will output something like **/home/pi**.

***MKDIR*:** You can use **mkdir** to create a new directory, e.g. **mkdir newDir** would create the directory **newDir** in the present working directory.

***RMDIR*:** To remove empty directories, use **rmdir**. So, for example, **rmdir oldDir** will remove the directory **oldDir** only if it is empty.

***RM*:** The command **rmremoves** the specified file (or recursively from a directory when used with -r). Be careful with this command: files deleted in this way are mostly gone for good!

***CP*:** Using **cp** makes a copy of a file and places it at the specified location (this is similar to copying and pasting). For example**, cp ~/fileA /home/otherUser/** would copy the file fileA from your home directory to that of the user otherUser (assuming you have permission to copy it there). This command can either take FILE FILE (cp fileA fileB), FILE DIR (cp fileA /directoryB/) or -r DIR DIR (which recursively copies the contents of directories) as arguments.

***MV*:** The **mv** command moves a file and places it at the specified location (so where **cp** performs a 'copy-paste', **mv** performs a 'cut-paste'). The usage is similar to **cp**. So **mv ~/fileA /home/otherUser/** would move the file fileA from your home directory to that of the user otherUser. This command can either take FILE FILE (mv fileA fileB), FILE DIR (mv fileA /directoryB/) or DIR DIR (mv /directoryB /directoryC) as arguments. This command is also useful as a method to rename files and directories after they've been created.

***TOUCH*:** The command **touch** sets the last modified time-stamp of the specified file(s) or creates it if it does not already exist.

***CAT*:** You can use **cat** to list the contents of file(s), e.g. **cat thisFile** will display the contents of thisFile. Can be used to list the contents of multiple files, i.e. **cat *.txt** will list the contents of all .txt files in the current directory.

***HEAD*:** The **head** command displays the beginning of a file. Can be used with **-n** to specify the number of lines to show (by default ten), or with **-c** to specify the number of bytes.

***TAIL:* The** opposite of **head**, **tail** displays the end of a file. The starting point in the file can be specified either through **-b** for 512-byte blocks, **-c** for bytes, or **-n** for number of lines.

***CHMOD:*** You would normally use **chmod** to change the permissions for a file. The **chmod** command can use symbols **u** (user that owns the file), **g** (the files group), and **o** (other users) and the permissions **r** (read), **w** (write), and **x** (execute). Using chmod **u+x *filename*** will add execute permission for the owner of the file.

***CHOWN:*** The **chown** command changes the user and/or group that owns a file. It normally needs to be run as root using **sudo e.g. sudo chown pi:root *filename*** will change the owner to pi and the group to root.

***SSH:* ssh** denotes the secure shell. Connect to another computer using an encrypted network connection. For more details see SSH (secure shell)
***SCP:*** The **scp** command copies a file from one computer to another using **ssh**. For more details see **SCP** (secure copy).

***SUDO:*** The **sudo** command enables you to run a command as a superuser, or another user. Use **sudo -s** for a superuser shell. For more details see Root user / sudo.

***DD:*** The **dd** command copies a file converting the file as specified. It is often used to copy an entire disk to a single file or back again. So, for example, **dd if=/dev/sdd of=backup.img** will create a backup image from an SD card or USB disk drive at **/dev/sdd.** Make sure to use the correct drive when copying an image to the SD card as it can overwrite the entire disk.

***DF:*** Use **df** to display the disk space available and used on the mounted filesystems. Use **df -h** to see the output in a human-readable format using **M** for MBs rather than showing number of bytes.

***UNZIP:*** The unzip command extracts the files from a compressed zip file.

***TAR:*** Use **tar** to store or extract files from a tape archive file. It can also reduce the space required by compressing the file similar to a zip file.To create a compressed file, use **tar -cvzf *filename.tar.gz* *directory/*** To extract the contents of a file, use **tar -xvzf *filename.tar.gz***

***PIPES:*** A **pipe** allows the output from one command to be used as the input for another command. The pipe symbol is a vertical line |. For example, to only show the first ten entries of the **ls** command it can be piped through the head **command ls | head.**

***TREE:*** Use the **tree** command to show a directory and all subdirectories and files indented as a **tree** structure.

**&:** Run a command in the background with **&**, freeing up the shell for future commands.

***WGET:*** Download a file from the web directly to the computer with **wget**. So **wget** https://www.raspberrypi.org/documentation/linux/usage/commands.md will download this file to your computer as commands.md

***CURL:*** Use **curl** to download or upload a file to/from a server. By default, it will output the file contents of the file to the screen.

***MAN:*** Show the manual page for a file with **man**. To find out more, run **man man** to view the manual page of the man command.

**Search**

***GREP:*** Use **grep** to search inside files for certain search patterns. For example, **grep "search" *.txt** will look in all the files in the current directory ending with .txt for the string search.
The **grep** command supports regular expressions which allows special letter combinations to be included in the search.

**AWK: awk** is a programming language useful for searching and manipulating text files.

***FIND:*** The **find** command searches a directory and subdirectories for files matching certain patterns.

***WHEREIS*:** Use **whereis** to find the location of a command. It looks through standard program locations until it finds the requested command.

**Networking**

***PING*:** The **ping** utility is usually used to check if communication can be made with another host. It can be used with default settings by just specifying a hostname (e.g. **ping raspberrypi.org**) or an IP address (e.g. **ping 8.8.8.8**). It can specify the number of packets to send with the **-c** flag.

***NMAP*: nmap** is a network exploration and scanning tool. It can return port and OS information about a host or a range of hosts. Running just **nmap** will display the options available as well as example usage.

***HOSTNAME*:** The hostname command displays the current hostname of the system. A privileged (super) user can set the hostname to a new one by supplying it as an argument (e.g. **hostname new-host**).

***IFCONFIG*:** Use **ifconfig** to display the network configuration details for the interfaces on the current system when run without any arguments (**i.e. ifconfig**). By supplying the command with the name of an interface (**e.g. eth0 or lo**) you can then alter the configuration: check the manual page for more details.

**LAB Exercise: Simply Glowing an LED**

**Introduction:** In this experiment an LED will be controlled by using Raspberry Pi. Python will be used to blinking an LED. This experiment will give a basic idea of Python language as well as importing GPIO pins of Raspberry Pi. This experiment will be done from Linux environment thus enable to use of terminal and shell scripting.

**Theories**

One of the biggest selling points of the Raspberry Pi is its GPIO, or General-Purpose Input/Output ports. They are the little pins sticking out of the circuit board and allow to plug various devices into Pi. With a little programming, one can easily control them or detect what they are doing. Programing Language Python will be used in this experiment. As Raspberry Pi runs on Linux environment it is always advised to use text editors like Gvim, Nano editor, Emacs Editor and Pico Editor. However, when you installed your Raspbian it comes with Integrated development environment for Python.

**Install RPi.GPIO Python Library**

The RPi.GPIO Python library allows you to easily configure and read-write the input/output pins on the Pi's GPIO header within a Python script. If you are using a fresh image you don't need to install it. If the library is not pre-installed, you can install it by following commands from Linux terminal.
If the package exists in the Raspbian repository is can be installed using apt-get. First you need to update the available package versions :

```
sudo apt-get update
```

Then attempt to install the RPi.GPIO package :

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

You can install them manually as well and to do that follow the link : https://www.raspberrypi-spy.co.uk/2012/05/install-rpi-gpio-python-library/

**Apparatus:**

1) Activated Raspberry pi
2) LED
3) Resistor (220Ω)
4) Breadboard
5) Jumper wires

**Experimental Procedure:**

1) The first step of this lab task is to setup the circuit as shown in Fig13.
2) After setting up the circuit Raspberry Pi should be powered On interfaced with Monitor and Key Board.
3) Open the terminal.
4) Write your code according to the following code example in Gvim text editor. You can use any editor.
5) After that give commands from linux to run to code in Raspberry Pi.

1. **Setting up the circuit**

The first step in this project is to design a simple LED circuit. Then we will make the LED circuit controllable from the Raspberry Pi by connecting the circuit to the general-purpose input/output (GPIO) pins on the Raspberry Pi.

A simple LED circuit consists of a LED and resistor. The resistor is used to limit the current that is being drawn and is called a current limiting resistor. Without the resistor the LED would run at too high of a voltage, resulting in too much current being drawn which in turn would instantly burn the LED, and likely also the GPIO port on the Raspberry Pi.

To calculate the resistor value we need to examine the specifications of the LED. Specifically we need to find the forward voltage (VF) and the forward current (IF). A regular red LED has a forward voltage (VF) of 1.7V and forward current of 20mA (IF). Additionally we need to know the output voltage of the Raspberry Pi which is 3.3V.

We can then calculate the resistor size needed to limit the current to the LED's maximum forward current (IF) using ohm's law like this:

$$R_{\Omega} = \frac{3.3V - VF}{IF} = \frac{3.3 - 1.7}{20mA} = 80\Omega$$

Unfortunately, 80 ohm is not a standard size of a resistor. To solve this, we can either combine multiple resistors, or round up to a standard size. In this case we would round up to 100 ohms. With the value calculated for the current limiting resistor we can now hook the LED and resistor up to GPIO pin 8 on the Raspberry Pi. The resistor and LED need to be in series like the diagram below.

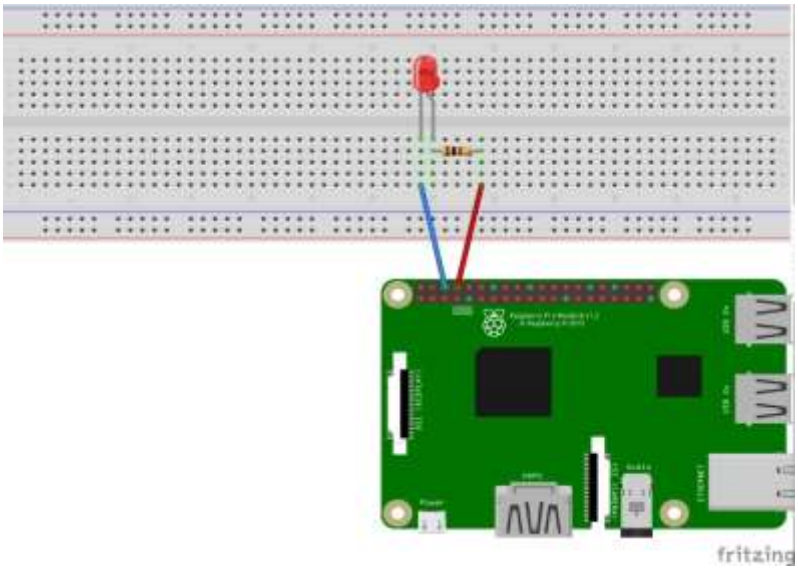


Figure 13 : Setting Up Circuit

When hooking up the circuit note the polarity of the LED. You will notice that the LED has a long and short lead. The long lead is the positive side also called the anode, the short lead is the negative side called the cathode. The long should be connected to the resistor and the short lead should be connected to ground via the blue jumper wire and pin 6 on the Raspberry Pi as shown on the Fig13. To find the pin number refer to Fig.5 showing the physical pin numbers on the Raspberry Pi.

2. **Writing the Python Software to blink the LED**

First of all, GVim Editor for writing the Python Program will be used here. Vim is a Command Line Editor and is a very simple and easy to use text editor. o install the Vim Editor (by default, Raspbian has Vi editor and to get full features of GVim, enter the following code in the SSH / Linux Terminal.

```
sudo apt-get install vim-gnome
```

Now open a blank Python file using GVim editor with the file name being *blinkLed.py*. For this, use the following command.

```
gvim blinkLed.py
```

After opening the blinkLed.py file Write the code as given :

**Simple Python Code for Blinking LED**

```
#!/usr/bin/env python
import RPi.GPIO as GPIO        #RPi.GPIO can be referred ads GPIO from now
import time
ledpin = 22                    #pin 22
GPIO.setwarnings(False)  #Ignore Warning For Now
GPIO.setmode(GPIO.BOARD)    #GPIO Numbering of Pins Board wise.   BCM can
also be used.
GPIO.setup(ledpin,GPIO.OUT)    #Set ledpin as output
GPIO.output(ledpin,GPIO.LOW) #Set ledpin to Low to turn off the LED
while True:                # Forever Loop
    print 'LED on'              # LED On will be printed in Terminal
    GPIO.output(ledpin,GPIO.HIGH) # LED On
    time.sleep(1.0)                #wait 1 sec
    print 'LED OFF'
    GPIO.output(ledpin,GPIO.LOW) #LED OFF
    time.sleep(1.0) #wait for 1 sec
```

After writing the code save it and run following commands to execute your codes for blinking LED.

| *sudo python blinkLed.py* |
|---|

If everything goes well, your LED should blink at an interval 1 second i.e. on for one second and off for other second.
For writing Python you have to very careful of using '**TAB**'. The most "pythonic" way is to use **4 spaces** per indentation level. Never mix **tabs** and **space.**

To stop the blinking in terminal press Ctrl + C.

In Python you can define your own function very easily and you can call the functions for your specific task. Now we are going to see the same code which is done by defining function.

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import time

LedPin = 11    # pin11

def setup():
        GPIO.setwarnings(False)          # Warnings ignored
        GPIO.setmode(GPIO.BOARD)       # Numbers GPIOs by physical location
        GPIO.setup(LedPin, GPIO.OUT)   # Set LedPin's mode is output
        GPIO.output(LedPin, GPIO.HIGH) # Set LedPin high(+3.3V) to off led

def loop():
        while True:
                print 'led on'
                GPIO.output(LedPin, GPIO.LOW)  # led on
                time.sleep(0.5)
                print 'led off'
                GPIO.output(LedPin, GPIO.HIGH) # led off
                time.sleep(0.5)

def destroy():
        GPIO.output(LedPin, GPIO.HIGH)     # led off
        GPIO.cleanup()                     # Release resource

if __name__ == '__main__':     # Program start from here
        setup()
        try:
                loop()
```

```
        except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child
program destroy() will be  executed.
                destroy()
```

**Lab Task:** Design a traffic control system using RED, YELLOW and GREEN LEDs with 2 seconds interval of glowing state of each LED.

**Questions for report writing:**

1) Include all codes and scripts into lab report

**Conclusion**

In this Laboratory exercise, blinking of LED using Raspberry Pi and Python Program has been shown. This project will help to understand some basics of the GPIO Pins of Raspberry Pi. With this project as reference, many other projects like driving motors, connecting LCDs, etc can be developed.

**Reference(s):**

1) Raspberry pi datasheet.
2) https://www.raspberrypi.org/documentation/linux/
3) https://www.raspberrypi.org/documentation/remote-access/ssh/
4) https://www.raspberrypi.org/documentation/remote-access/ssh/scp.md
5) https://www.raspberrypi.org/documentation/linux/usage/root.md
6) https://www.raspberrypi.org/documentation/usage/python/