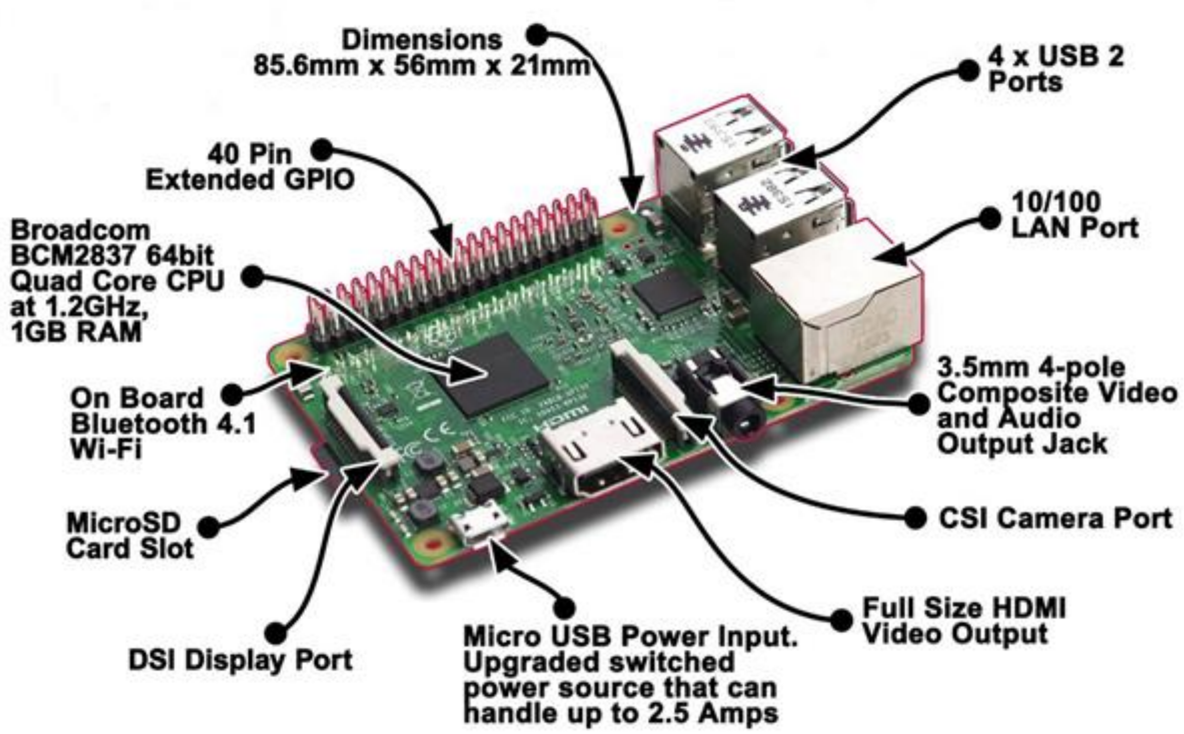
INDEX

Raspberry Pi Introduction


Dimensions
85.6mm x 56mm x 21mm
4 x USB 2 Ports
40 Pin Extended GPIO
10/100 LAN Port
Broadcom BCM2837 64bit Quad Core CPU at 1.2GHz, 1GB RAM
3.5mm 4-pole Composite Video and Audio Output Jack
On Board Bluetooth 4.1 Wi-Fi
MicroSD Card Slot
CSI Camera Port
Full Size HDMI Video Output
DSI Display Port
Micro USB Power Input. Upgraded switched power source that can handle up to 2.5 Amps

Raspberry Pi 3 Model B
The Raspberry Pi 3 Model B is the third generation Raspberry Pi. This powerful credit-card sized single board computer can be used for many applications and supersedes the original Raspberry Pi Model B+ and Raspberry Pi 2 Model B. Additionally it adds wireless LAN & Bluetooth connectivity making it the ideal solution for powerful connected designs.

Raspberry Pi 3 - Model B Technical Specification

- Broadcom BCM2387 chipset
- 1.2GHz Quad-Core ARM Cortex-A53
- 802.11 bgn Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
- 1GB RAM
- 64 Bit CPU
- 4 x USB ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- 10/100 BaseT Ethernet socketbr
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data
- Micro USB power source

USB ports

The Raspberry Pi 2 has four USB ports, allowing you connect it to keyboards, mice, WiFi dongles, and USB sticks containing all your files. Since the ports don't provide much power, if you want to add a USB hub to the Pi you'll need to find one that comes with an external power supply.

Ethernet port

The traditional way to connect to the internet is via a wire called an Ethernet cable. You'll find a few similar ports like this at the rear of your router at home that will let you connect the Raspberry Pi directly into it. This method is easier to set up than WiFi and may provide faster internet, but you're then limited by the length of the cable.

MicroSD card slot

A little SD card is used as the Raspberry Pi's hard drive. This is where the operating system will live once you've put it on there. Most computers won't be able to directly connect to a microSD card, but you can get an adaptor that plugs into normal SD card slots

Audio out

This looks like a headphone socket because that's exactly what it is. A 3.5mm jack to be precise, this allows you to connect the Pi to computer speakers, or you could even plug in your favourite headphones and have a Raspberry jam.

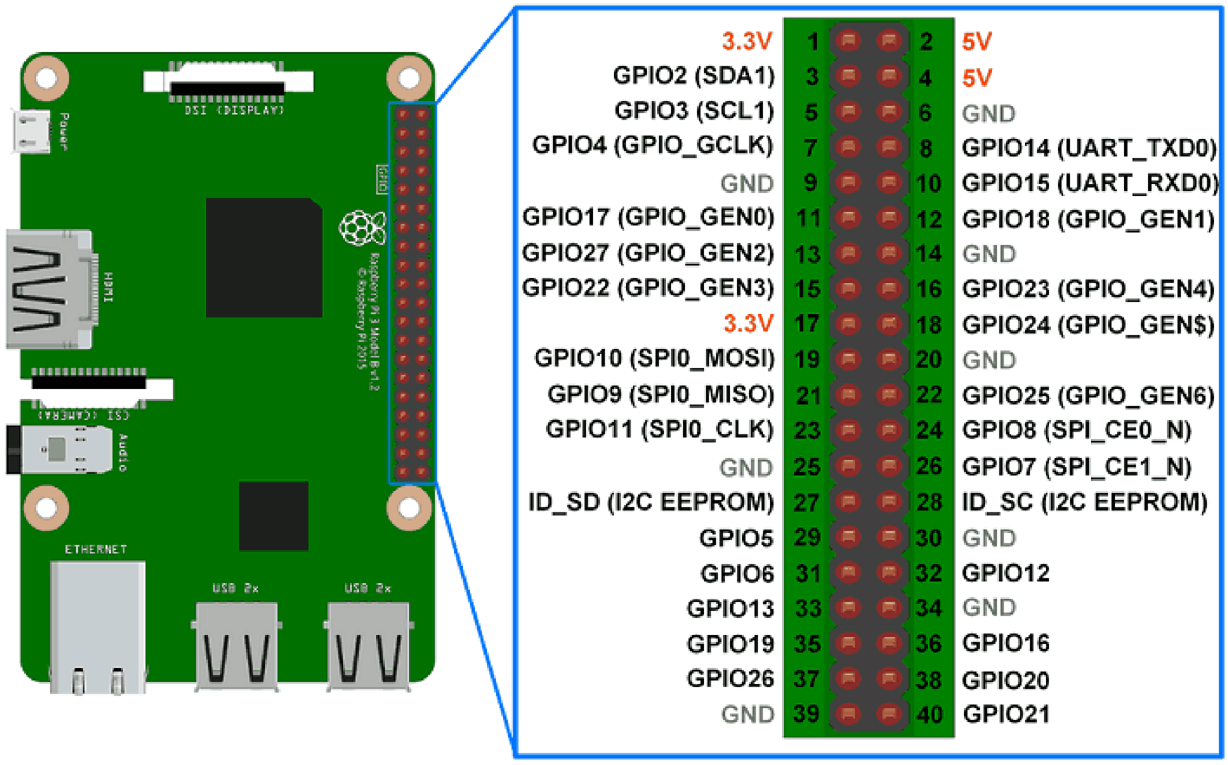
HDMI port

This is an HDMI port, the kind you'll find on the back of most modern TVs and computer monitors. Use a standard HDMI cable to connect your Raspberry Pi to your chosen screen, to see (and hear) whatever it's doing. You'll definitely need to plug it in to set up the Pi.

Power

This is the kind of small charging port you might find in your smartphone. This micro-USB port means you can power the Pi with the right kind of mobile phone charger or directly from your PC – however, it's best to use the official Raspberry Pi power supply to make sure the Pi is getting enough power.

GPIO Introduction


DSI (DISPLAY)
Power
HDMI
CSI (CAMERA)
Audio
ETHERNET
USB 2x
USB 2x
GPIO
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
3.3V 1 2 5V
GPIO2 (SDA1) 3 4 5V
GPIO3 (SCL1) 5 6 GND
GPIO4 (GPIO_GCLK) 7 8 GPIO14 (UART_TXD0)
GND 9 10 GPIO15 (UART_RXD0)
GPIO17 (GPIO_GEN0) 11 12 GPIO18 (GPIO_GEN1)
GPIO27 (GPIO_GEN2) 13 14 GND
GPIO22 (GPIO_GEN3) 15 16 GPIO23 (GPIO_GEN4)
3.3V 17 18 GPIO24 (GPIO_GEN$)
GPIO10 (SPI0_MOSI) 19 20 GND
GPIO9 (SPI0_MISO) 21 22 GPIO25 (GPIO_GEN6)
GPIO11 (SPI0_CLK) 23 24 GPIO8 (SPI_CE0_N)
GND 25 26 GPIO7 (SPI_CE1_N)
ID_SD (I2C EEPROM) 27 28 ID_SC (I2C EEPROM)
GPIO5 29 30 GND
GPIO6 31 32 GPIO12
GPIO13 33 34 GND
GPIO19 35 36 GPIO16
GPIO26 37 38 GPIO20
GND 39 40 GPIO21

The GPIO header comprises the general purpose input/output (GPIO) pins. They're a set of connections that have various functions, but their main one is to allow you to connect to the Raspberry Pi with an electronic circuit. These pins are an interface between the Pi and the outside world.

Of the 40 pins, 26 are GPIO pins and the others are power or ground pins. In GPIO you can use either pin numbers (BOARD) or physical numbering or the Broadcom GPIO numbers (BCM), but you can only use one system in each program.

The ground pins in physical numbering are 6,9,14,20,25,30,34 and 39. Raspberry Pi normally works with 3.3 power provided by pins 1 and 17. For devices that need 5V power supply, pins 2and 4 are provided.

Practical No: 1

Aim: Installing Raspbian OS, Familiarising with Raspberry Pi Components and interface, connecting to Ethernet, Monitor, and USB.

Steps For Installation Raspbian OS:

1. Format the SD card (with the help of SD card formatter)
2. Unzip the OS – Disk image file
3. Write the OS image file to SD card (win32 Disk image Writer)

Hardware requirement:

1. Raspberry Pi 3.
2. Monitor or TV: A monitor or TV is connected with HDMI cable with Raspberry Pi.
3. HDMI cable (High Definition Multimedia Interface)
4. Ethernet Cable: Ethernet cable will allow Pi to connect with the Internet.
5. USB keyboard: Any standard USB keyboard and mouse can be attached to Raspberry Pi.
6. USB Mouse: This is used to connect mouse with Raspberry Pi kit.
7. Micro USB Power Supply: We can use 5v, 2A power supply for all models of Raspberry Pi.
8. 8GB or larger microSD card: SD card is used to store the default OS, Raspbian.
9. SD card Reader: This is used to read the SD card.

Installation Steps:

1. Download Raspberry Pi OS with zip format.
   Unzip with an unzip tool. After unzipping the file, you will get a disc image (ISO.img) file in the unzipped folder.
2. Now format the SD card before writing the disc image file on the SD card using any disk formatter.
3. The image file of the Operating System can be written on the SD card with a Disk imager tool (win 32 Disk imager tool).

Plugging on Raspberry Pi:

1. Begin by placing your SD card into the SD card slot on the Raspberry Pi.
2. Plug your keyboard and mouse into the USB ports into the Raspberry Pi.
3. Connect your monitor to Raspberry Pi with the HDMI cord.
4. With all hardware's properly attached with the Raspberry Pi, connect the micro USB power supply ON and boot your computer.

Practical No :2

Aim: Displaying different LED patterns with Raspberry Pi

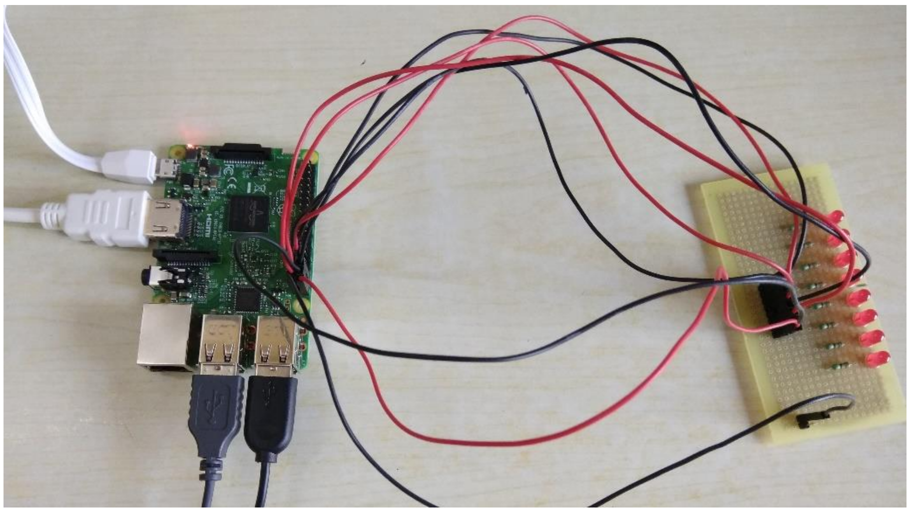
Components Required:

- Raspberry Pi
- Monitor
- HDMI Cable
- USB Keyboard and Mouse
- Micro USB power supply
- LED module
- Jumper wires

Wiring up the Circuit

- Connect the basic Pi setup
- Connect the LED module to the following physical GPIO pins of the Pi : 29,31,33,35,36,37,38,40
- Connect the ground pin of the LED module to ground of Raspberry pi
- Power on the Raspberry Pi

Circuit diagram:



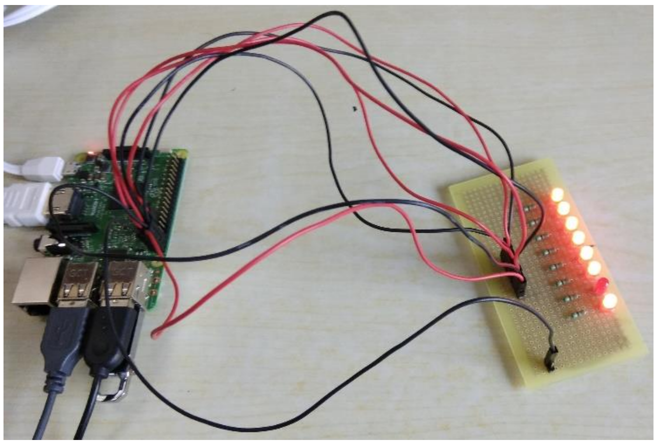
Software Guide:

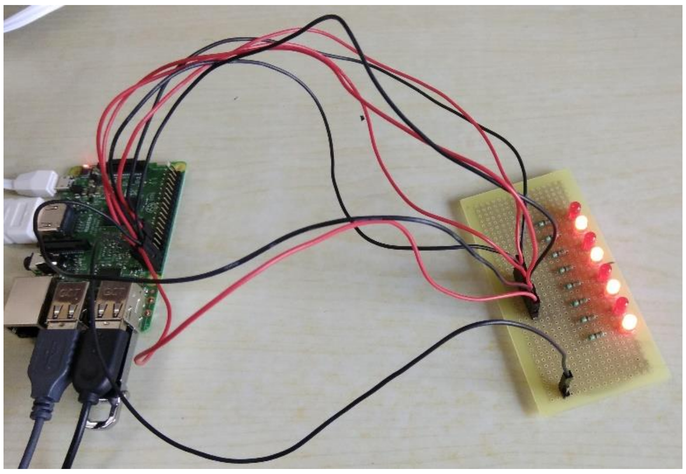
To open Python, click on the Application Menu, Goto Programming, click on Python 2.7(IDLE) an integrated development environment for Python 2.7. After opening the IDE, go to File and open new file to start writing the code.

Python Program:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
led1=29
led2=31
led3=33
led4=35
led5=36
led6=37
led7=38
led8=40
GPIO.setup(led1,GPIO.OUT)
GPIO.setup(led2,GPIO.OUT)
GPIO.setup(led3,GPIO.OUT)
GPIO.setup(led4,GPIO.OUT)
GPIO.setup(led5,GPIO.OUT)
GPIO.setup(led6,GPIO.OUT)
GPIO.setup(led7,GPIO.OUT)
GPIO.setup(led8,GPIO.OUT)
GPIO.output(led1,False)
GPIO.output(led2,False)
GPIO.output(led3,False)
GPIO.output(led4,False)
GPIO.output(led5,False)
GPIO.output(led6,False)
GPIO.output(led7,False)
GPIO.output(led8,False)

def ledpattern(ledVal1,ledVal2,ledVal3,ledVal4,ledVal5,ledVal6,ledVal7,ledVal8):
   GPIO.output(led1,ledVal1)
   GPIO.output(led2,ledVal2)
   GPIO.output(led3,ledVal3)
   GPIO.output(led4,ledVal4)
   GPIO.output(led5,ledVal5)
   GPIO.output(led6,ledVal6)
   GPIO.output(led7,ledVal7)
   GPIO.output(led8,ledVal8)

def patternOne():
   for i in range(0,3):
      ledpattern(1,0,1,0,1,0,1,0)
      time.sleep(1)
      ledpattern(0,1,0,1,0,1,0,1)
      time.sleep(1)



def patternTwo():
   for i in range(0,5):
      ledpattern(1,0,0,0,0,0,0,0)
      time.sleep(0.1)
      ledpattern(0,1,0,0,0,0,0,0)
      time.sleep(0.1)
      ledpattern(0,0,1,0,0,0,0,0)
      time.sleep(0.1)
      ledpattern(0,0,0,1,0,0,0,0)
      time.sleep(0.1)
      ledpattern(0,0,0,0,1,0,0,0)
      time.sleep(0.1)
      ledpattern(0,0,0,0,0,1,0,0)
      time.sleep(0.1)
      ledpattern(0,0,0,0,0,0,1,0)
      time.sleep(0.1)
      ledpattern(0,0,0,0,0,0,0,1)
      time.sleep(0.1)

def patternThree():
   for i in range(0,5):
      ledpattern(0,0,0,0,0,0,0,1)
      time.sleep(0.1)
      ledpattern(0,0,0,0,0,0,1,0)
      time.sleep(0.1)
      ledpattern(0,0,0,0,0,1,0,0)
      time.sleep(0.1)
      ledpattern(0,0,0,0,1,0,0,0)
      time.sleep(0.1)
      ledpattern(0,0,0,1,0,0,0,0)
      time.sleep(0.1)
      ledpattern(0,0,1,0,0,0,0,0)
      time.sleep(0.1)
      ledpattern(0,1,0,0,0,0,0,0)
      time.sleep(0.1)
      ledpattern(1,0,0,0,0,0,0,0)
      time.sleep(0.1)

def patternFour():
   for i in range(0,5):
      ledpattern(0,1,1,1,1,1,1,1)
      time.sleep(0.1)
      ledpattern(1,0,1,1,1,1,1,1)
      time.sleep(0.1)
      ledpattern(1,1,0,1,1,1,1,1)
      time.sleep(0.1)
      ledpattern(1,1,1,0,1,1,1,1)
      time.sleep(0.1)
      ledpattern(1,1,1,1,0,1,1,1)
      time.sleep(0.1)
      ledpattern(1,1,1,1,1,0,1,1)
      time.sleep(0.1)
      ledpattern(1,1,1,1,1,1,0,1)
      time.sleep(0.1)
      ledpattern(1,1,1,1,1,1,1,0)
      time.sleep(0.1)

def patternFive():
   for i in range(0,5):
      ledpattern(1,1,1,1,1,1,1,0)
      time.sleep(0.1)
      ledpattern(1,1,1,1,1,1,0,1)
      time.sleep(0.1)
      ledpattern(1,1,1,1,1,0,1,1)
      time.sleep(0.1)
      ledpattern(1,1,1,1,0,1,1,1)
      time.sleep(0.1)
      ledpattern(1,1,1,0,1,1,1,1)
      time.sleep(0.1)
      ledpattern(1,1,0,1,1,1,1,1)
      time.sleep(0.1)
      ledpattern(1,0,1,1,1,1,1,1)
      time.sleep(0.1)
      ledpattern(0,1,1,1,1,1,1,1)
      time.sleep(0.1)

try:
   while True:
      patternOne()
      patternTwo()
      patternThree()
      patternFour()
      patternFive()

finally:
   GPIO.cleanup()
```

Output:



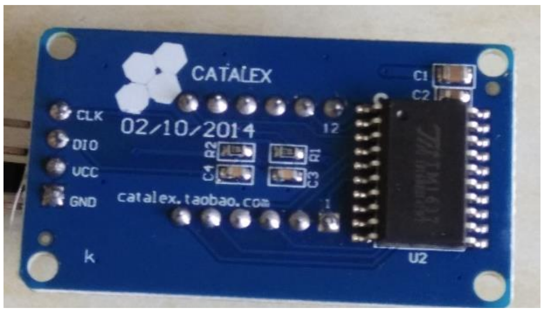


Practical no.3

Aim: Displaying Time over 4-Digit 7-Segment Display using Raspberry Pi

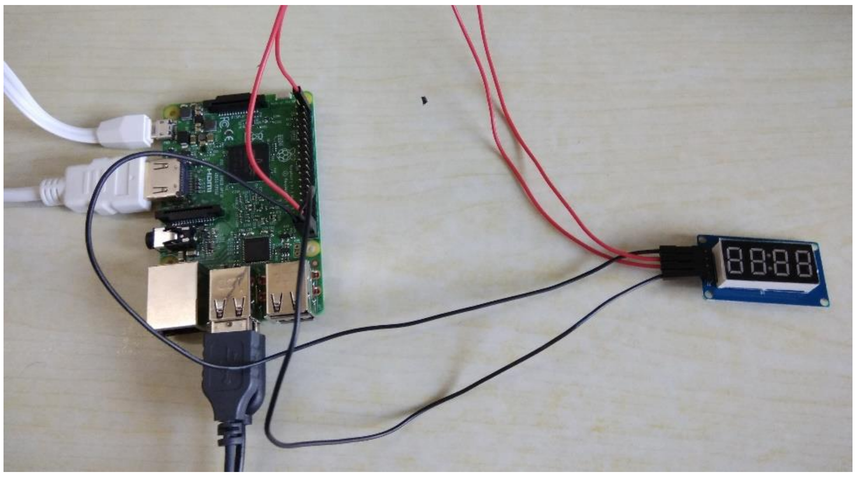
Components Used:

1. TM1637 4-Digit seven segment display board.
2. Connecting wires.
3. Raspberry Pi 3



Circuit connections:

1. Connect the Pin2(5V) of Rpi to Vcc pin of module.
2. Connect Pin 6 (GND) of Rpi to GND of module
3. Connect Pin38 (GPIO20) of Rpi to DIO of module
4. Lastly connect Pin 40 (GPIO21) of Rpi to CLK of module.



Software Guide:

1. Now to download libraries, open Web Browser on your Raspberry Pi and log on to the following link:

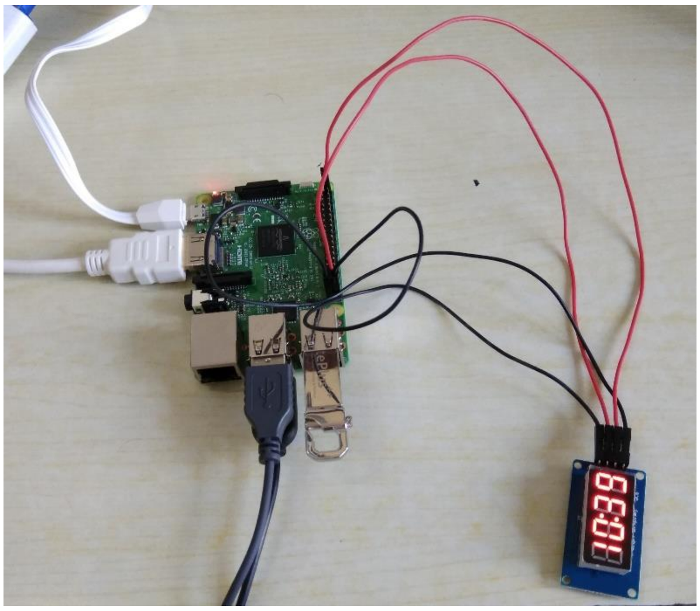
```
https://github.com/timwaizenegger/raspberrypi-examples/tree/master/actor-led-7segment-4numbers
```

   Click on the actor-led-7segment-4numbers.zip folder and now click on Download Button to download the file.
2. Now on your rpi move to /home/pi/Downloads/ location to find the zip file downloaded.
3. Unzip the file and try to execute the different example codes present in that folder in Python 2 idle.
4. Now open Python 2 Idle, create a new file, write the code given below and save it in the same folder i.e. actor-led-7segment-4numbers since the code below is depended on tm1637.py file which is present in the same folder.

Python Program:

```
#Program to display Time on 4-digit seven segment display

from time import sleep
import tm1637

try:
    import thread
except ImportError:
    import _thread as thread

# Initialize the clock (GND, VCC=3.3V, Example Pins are DIO-20 and CLK21)
Display = tm1637.TM1637(CLK=21, DIO=20, brightness=1.0)

try:
    print "Starting clock in the background (press CTRL + C to stop):"
    Display.StartClock(military_time=True)
    Display.SetBrightness(1.0)

    while True:
        Display.ShowDoublepoint(True)
        sleep(1)
        Display.ShowDoublepoint(False)
        sleep(1)

    Display.StopClock()
    thread.interrupt_main()
except KeyboardInterrupt:
    print "Properly closing the clock and open GPIO pins"
    Display.cleanup()
```

Output:



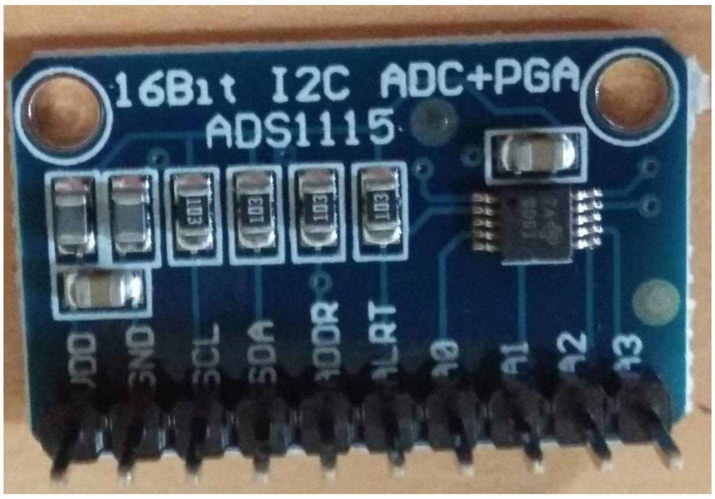
Practical No: 4

Aim: Raspberry Pi based Oscilloscope

The requirement for this practical can be classified into two:
1. Hardware Requirements
2. Software Requirements

<u>Hardware requirements</u>

To build this project, the following components/part are required;
1. Raspberry pi 2 (or any other model)
2. 8 or 16GB SD Card
3. LAN/Ethernet Cable
4. Power Supply or USB cable
5. ADS1115 ADC
6. LDR (Optional as its meant for test)
7. 10k or 1k resistor
8. Jumper wires
9. Breadboard
10. Monitor or any other way of seeing the pi's Desktop(VNC inclusive)

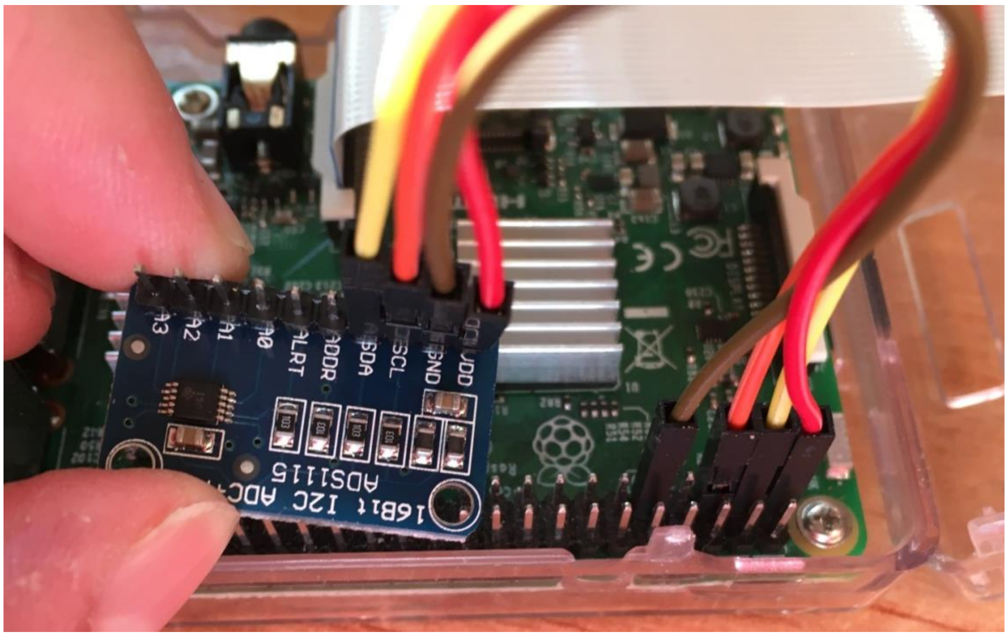


<u>Software Requirements</u>

The software requirements for this practical are the python modules (***matplotlib and drawnow***) that will be used for data visualization and the Adafruit module for interfacing with the ADS1115 ADC chip.

<u>Circuit Diagram:</u>

To convert the analog input signals to digital signals which can be visualized with the Raspberry Pi, we use the **ADS1115 ADC chip**. This chip becomes important because the Raspberry Pi, unlike Arduino and most micro-controllers, does not have an on-board analog to digital converter(ADC). This chip has high resolution(16bits) and has well documented datasheet and use instructions by Adafruit.



ADS1115 and Raspberry Pi Connections:

VDD – 3.3V (Pin 1)
GND – GND
SDA – SDA (Pin 3)
SCL – SCL (Pin 5)
With the connections all done, power up your pi and proceed to install the dependencies mentioned below.

Install Dependencies for Raspberry Pi Oscilloscope:

Before writing the python script to pull data from the ADC and plot it on a live graph, we need to **enable the I2C communication interface** of the raspberry pi and install the software requirements that were mentioned earlier. This will be done in below steps so its easy to follow:

Step 1: Enable Raspberry Pi I2C interface
To enable the I2C, from the terminal, run;

```
sudo raspi-config
```

When the configuration panels open, select interface options, select I2C and click enable.

Step 2: Update the Raspberry pi
The first thing I do before starting any project is updating the Pi. Through this, I am sure every thing on the OS is up to date and I won’t experience compatibility issue with any latest software I choose to install on the Pi. To do this, run below two commands:

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

Step 3: Install the Adafruit ADS1115 library for ADC

With the update done, we are now ready to install the dependencies starting with the Adafruit python module for the ADS115 chip. Ensure you are in the Raspberry Pi home directory by running;

```
cd ~
```

then install the build-essentials by running;

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

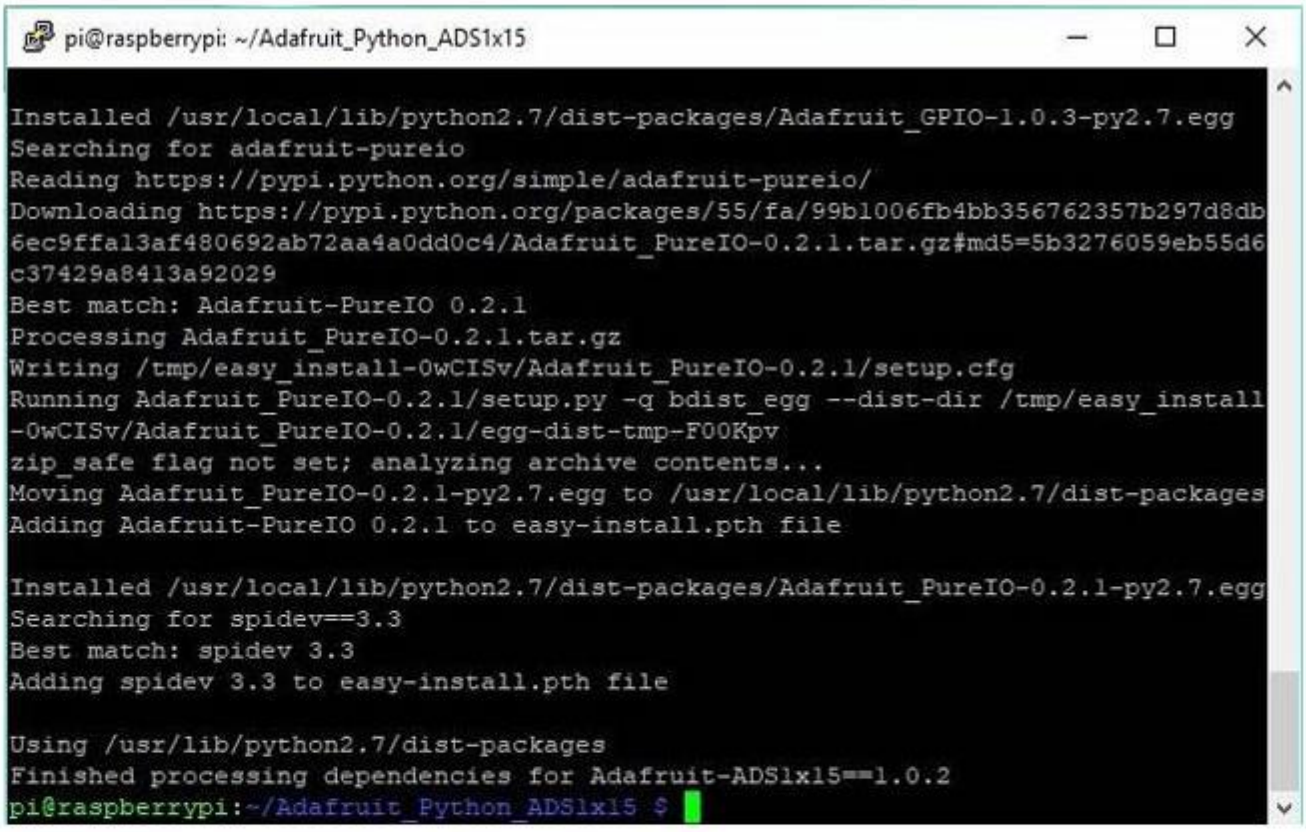
Next, clone the Adafruit git folder for the library by running;

```
git clone https://github.com/adafruit/Adafruit_Python_ADS1x15.git
```

Change into the cloned file's directory and run the setup file;

```
cd Adafruit_Python_ADS1x1z
sudo python setup.py install
```

After installation, your screen should look like the image below.



```
Installed /usr/local/lib/python2.7/dist-packages/Adafruit_GPIO-1.0.3-py2.7.egg
Searching for adafruit-pureio
Reading https://pypi.python.org/simple/adafruit-pureio/
Downloading https://pypi.python.org/packages/55/fa/99b1006fb4bb356762357b297d8db
6ec9ffa13af480692ab72aa4a0dd0c4/Adafruit_PureIO-0.2.1.tar.gz#md5=5b3276059eb55d6
c37429a8413a92029
Best match: Adafruit-PureIO 0.2.1
Processing Adafruit_PureIO-0.2.1.tar.gz
Writing /tmp/easy_install-0wCISv/Adafruit_PureIO-0.2.1/setup.cfg
Running Adafruit_PureIO-0.2.1/setup.py -q bdist_egg --dist-dir /tmp/easy_install
-0wCISv/Adafruit_PureIO-0.2.1/egg-dist-tmp-F00Kpv
zip_safe flag not set; analyzing archive contents...
Moving Adafruit_PureIO-0.2.1-py2.7.egg to /usr/local/lib/python2.7/dist-packages
Adding Adafruit-PureIO 0.2.1 to easy-install.pth file

Installed /usr/local/lib/python2.7/dist-packages/Adafruit_PureIO-0.2.1-py2.7.egg
Searching for spidev==3.3
Best match: spidev 3.3
Adding spidev 3.3 to easy-install.pth file

Using /usr/lib/python2.7/dist-packages
Finished processing dependencies for Adafruit-ADS1x15==1.0.2
pi@raspberrypi:~/Adafruit_Python_ADS1x15 $
```

<u>Step 4: Test the library and 12C communication.</u>

To test the library and ensure the ADC can communicate with the raspberry pi over I2C, we will use an example script that comes with the library.

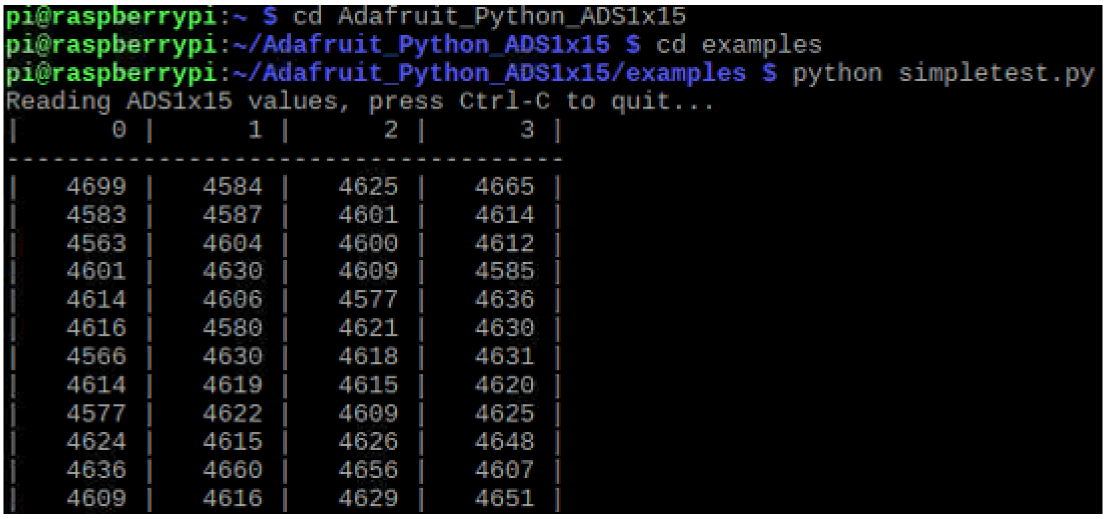
While still in the Adafruit_Python_ADS1x15 folder, change directory to the examples directory by running;

```
cd examples
```

Next, run the sampletest.py example which displays the value of the four channels on the ADC in a tabular form. Run the example using:

```
python simpletest.py
```

If the I2C module is enabled and connections good, you should see the data as shown in the image below.



```
pi@raspberrypi:~ $ cd Adafruit_Python_ADS1x15
pi@raspberrypi:~/Adafruit_Python_ADS1x15 $ cd examples
pi@raspberrypi:~/Adafruit_Python_ADS1x15/examples $ python simpletest.py
Reading ADS1x15 values, press Ctrl-C to quit...
|      0 |      1 |      2 |      3 |
-------------------------------------
|   4699 |   4584 |   4625 |   4665 |
|   4583 |   4587 |   4601 |   4614 |
|   4563 |   4604 |   4600 |   4612 |
|   4601 |   4630 |   4609 |   4585 |
|   4614 |   4606 |   4577 |   4636 |
|   4616 |   4580 |   4621 |   4630 |
|   4566 |   4630 |   4618 |   4631 |
|   4614 |   4619 |   4615 |   4620 |
|   4577 |   4622 |   4609 |   4625 |
|   4624 |   4615 |   4626 |   4648 |
|   4636 |   4660 |   4656 |   4607 |
|   4609 |   4616 |   4629 |   4651 |
```

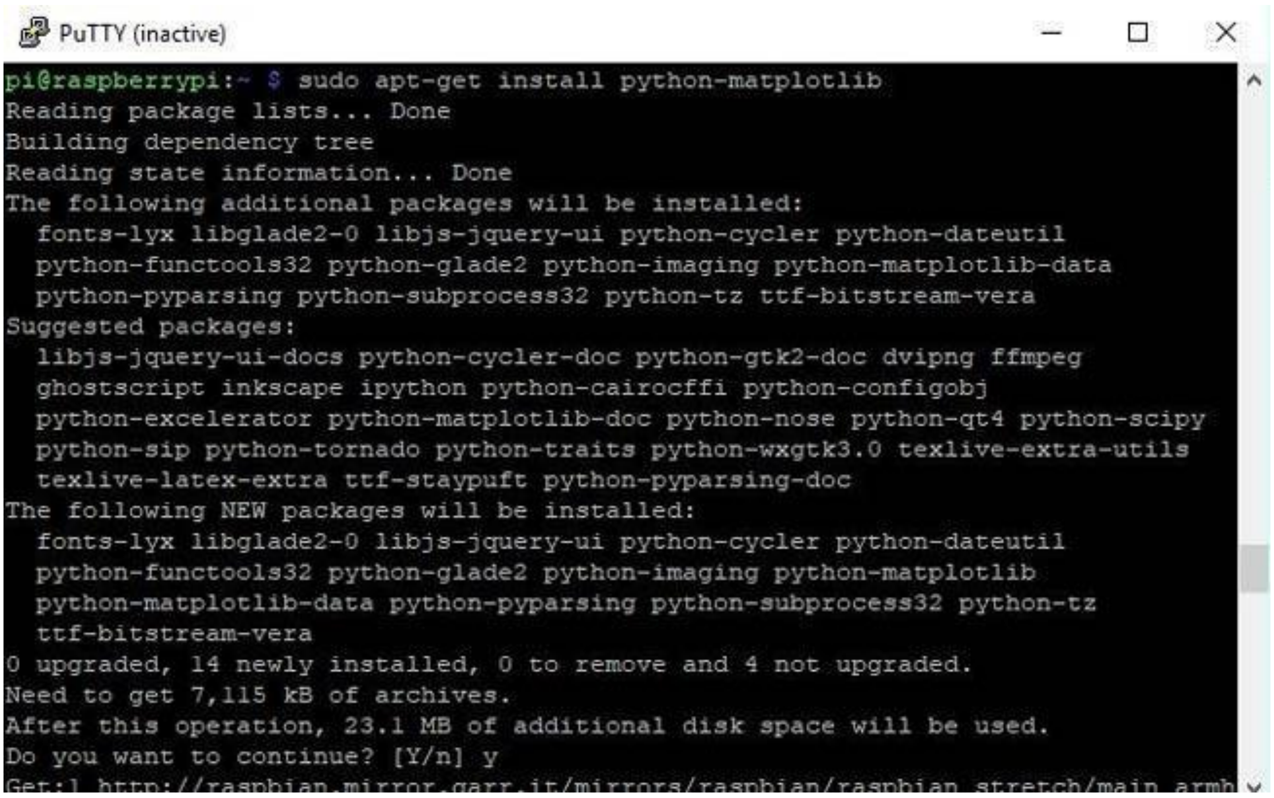
If an error occurs, check to ensure the ADC is well connected to the PI and I2C communication is enabled on the Pi.

<u>Step 5: Install *Matplotlib*</u>

To visualize the data we need to install the *matplotlib* module which is used to plot all kind of graphs in python. This can be done by running;

```
sudo apt-get install python-matplotlib
```

You should see an outcome like the image below.



```
pi@raspberrypi:~ $ sudo apt-get install python-matplotlib
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  fonts-lyx libglade2-0 libjs-jquery-ui python-cycler python-dateutil
  python-functools32 python-glade2 python-imaging python-matplotlib-data
  python-pyparsing python-subprocess32 python-tz ttf-bitstream-vera
Suggested packages:
  libjs-jquery-ui-docs python-cycler-doc python-gtk2-doc dvipng ffmpeg
  ghostscript inkscape ipython python-cairocffi python-configobj
  python-excelerator python-matplotlib-doc python-nose python-qt4 python-scipy
  python-sip python-tornado python-traits python-wxgtk3.0 texlive-extra-utils
  texlive-latex-extra ttf-staypuft python-pyparsing-doc
The following NEW packages will be installed:
  fonts-lyx libglade2-0 libjs-jquery-ui python-cycler python-dateutil
  python-functools32 python-glade2 python-imaging python-matplotlib
  python-matplotlib-data python-pyparsing python-subprocess32 python-tz
  ttf-bitstream-vera
0 upgraded, 14 newly installed, 0 to remove and 4 not upgraded.
Need to get 7,115 kB of archives.
After this operation, 23.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://raspbian.mirror.garr.it/mirrors/raspbian/raspbian stretch/main armh
```

<u>Step6: Install the *Drawnow* python module</u>

Lastly, we need to install the *drawnow* python module. This module helps us provide live updates to the data plot.

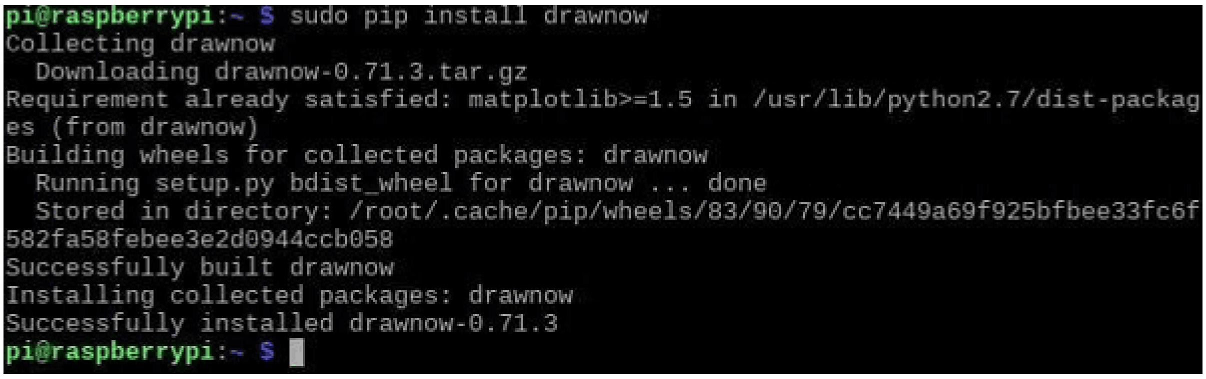
We will be installing *drawnow* via the python package installer; *pip*, so we need to ensure it is installed. This can be done by running;

```
sudo apt-get install python-pip
```

We can then use pip to install the *drawnow* package by running:

```
sudo pip install drawnow
```

You should get an outcome like the image below after running it.



```
pi@raspberrypi:~ $ sudo pip install drawnow
Collecting drawnow
  Downloading drawnow-0.71.3.tar.gz
Requirement already satisfied: matplotlib>=1.5 in /usr/lib/python2.7/dist-packag
es (from drawnow)
Building wheels for collected packages: drawnow
  Running setup.py bdist_wheel for drawnow ... done
  Stored in directory: /root/.cache/pip/wheels/83/90/79/cc7449a69f925bfbee33fc6f
582fa58febee3e2d0944ccb058
Successfully built drawnow
Installing collected packages: drawnow
Successfully installed drawnow-0.71.3
pi@raspberrypi:~ $ 
```

With all the dependencies installed, then next step is to write the code.

Python Code for Raspberry Pi Oscilloscope:

The python code for this **Pi Oscilloscope** is fairly simple especially if you are familiar with the python *matplotlib* module.
At this stage it is important to switch to a monitor through which you can see your Raspberry Pi's desktop, as the graph being plotted won't show on the terminal.

With the monitor as the interface **open a new python file**.

```
sudo nano scope.py
```

With the file created, the first thing we do is import the modules we will be using;

```
import time
import matplotlib.pyplot as plt
from drawnow import *
import Adafruit_ADS1x15
```

Next, we **create an instance of the ADS1x15 library** specifying the ADS1115 ADC

```
adc = Adafruit_ADS1x15.ADS1115()
```

Next, we set the gain of the ADC. There are different ranges of gain and should be chosen based on the voltage you are expecting at the input of the ADC. For this practical, we will be using a gain of 1.

```
GAIN = 1
```

Next, we need to create the array variables that will be used to store the data to be plotted and another one to serve as count.

```
Val = [ ]
cnt = 0
```

Next, we make the plot interactive to help us in **enabling to plot the data live**.

```
plt.ion()
```

Next, we start continuous ADC conversion **specifying the ADC channel**, in this case, channel 0 and we also specify the gain. It should be noted that all the four ADC channels on the ADS1115 can be read at the same time, but 1 channel is enough for this demonstration.

```
adc.start_adc(0, gain=GAIN)
```

Next we create a function *def makeFig*, to **create and set the attributes of the graph** which will hold our live plot. We first of all set the limits of the y-axis using *ylim*, after which we input the title of the plot, and the label name before we specify the data that will be plotted and its plot style and color using *plt.plot().* We can also state the channel (as channel 0 was stated) so we can identify each signal when the four channels of the ADC are being used. *plt.legend* is used to specify where we want the information about that signal(e.g Channel 0) displayed on the figure.

```
plt.ylim(-5000,5000)
plt.title('Osciloscope')
plt.grid(True)

plt.ylabel('ADC outputs')
plt.plot(val, 'ro-', label='lux')
plt.legend(loc='lower right')
```

Next we write the *while* loop which will be used constantly read data from the ADC and update the plot accordingly.
The first thing we do is **read the ADC conversion value**

```
value = adc.get_last_result()
```

Next we print the value on the terminal just to give us another way of confirming the plotted data. We wait a few seconds after printing then we append the data to the list (val) created to store the data for that channel.

```
print('Channel 0: {0}'.format(value))
time.sleep(0.5)
val.append(int(value))
```

We then call *drawnow* to update the plot.

```
drawnow(makeFig)
```

To ensure the latest data is what is available on the plot, we delete the data at index 0 after every 50 data counts.

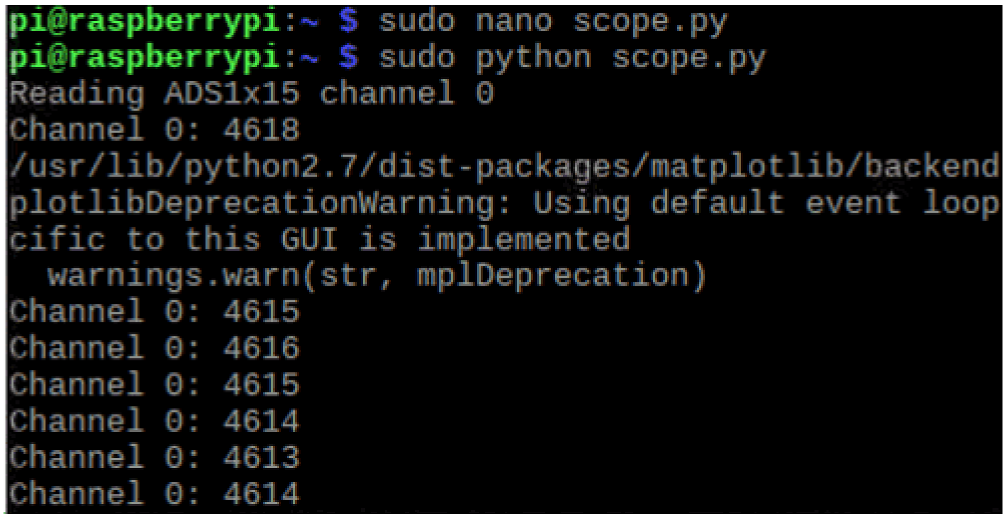
```
cnt = cnt+1
if(cnt>50):
val.pop(0)
```

Save the code and run using;

```
sudo python scope.py
```

After a few minutes, you should see the ADC data being printed on the terminal. Occasionally you may get a warning from *matplotlib* (as shown in the image below) which should be suppressed but it doesn't affect the data being displayed or the plot in anyway. To suppress the warning however, the following lines of code can be added after the import lines in our code.

```
import warnings
import matplotlib.cbook
warnings.filterwarnings("ignore", category=matplotlib.cbook.mplDeprecation)
```

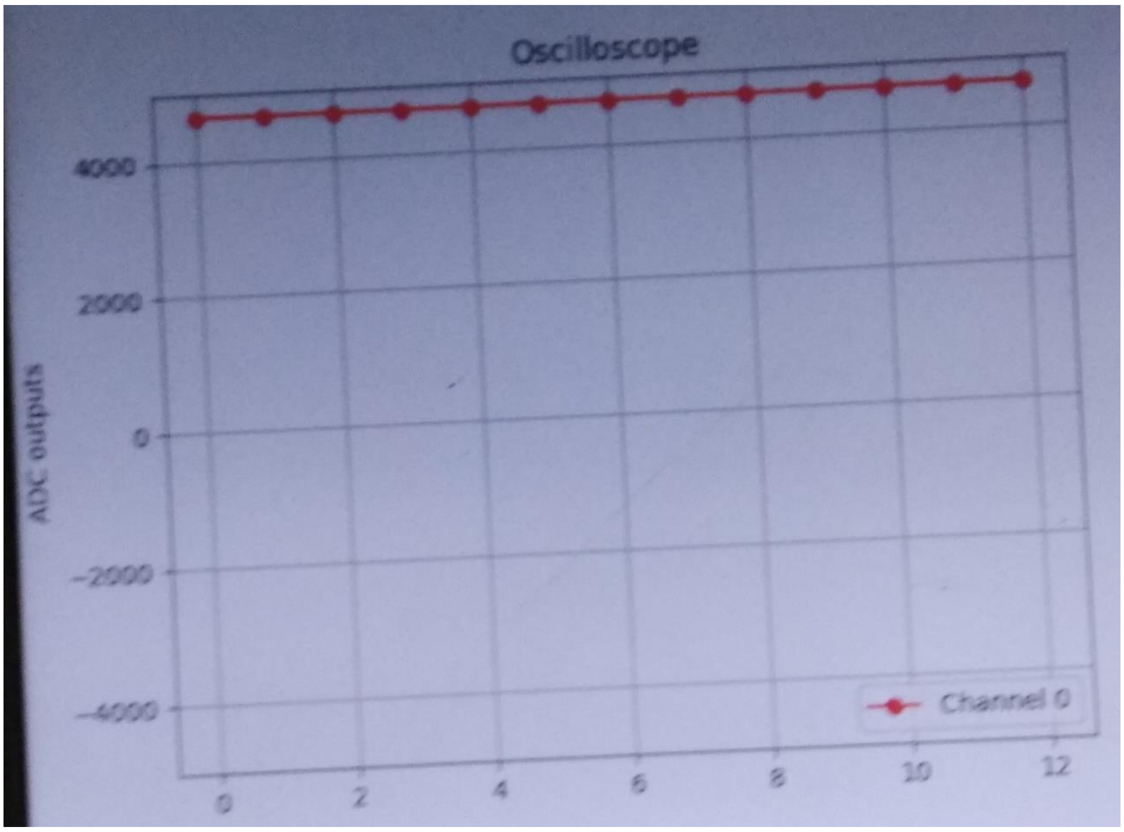


```
pi@raspberrypi:~ $ sudo nano scope.py
pi@raspberrypi:~ $ sudo python scope.py
Reading ADS1x15 channel 0
Channel 0: 4618
/usr/lib/python2.7/dist-packages/matplotlib/backend
plotlibDeprecationWarning: Using default event loop
cific to this GUI is implemented
  warnings.warn(str, mplDeprecation)
Channel 0: 4615
Channel 0: 4616
Channel 0: 4615
Channel 0: 4614
Channel 0: 4613
Channel 0: 4614
```

Python Program:

```
import time
import matplotlib.pyplot as plt
from drawnow import *
import Adafruit_ADS1x15
adc=Adafruit_ADS1x15.ADS1115()

GAIN=1
val=[]
cnt=0
plt.ion()
adc.start_adc(0,gain=GAIN)
print('Reading ADS1x15 channel 0')
def makeFig():
  plt.ylim(-5000,5000)
  plt.title('Oscilloscope')
  plt.grid(True);
  plt.ylabel('ADC outputs')
  plt.plot(val,'ro-',label='Channel 0')
  plt.legend(loc='lower right')
while (True):
    value=adc.get_last_result()
    print('Channel 0: {0}'.format(value))
    time.sleep(0.5)
    val.append(int(value))
    drawnow(makeFig)
    plt.pause(.000001)
    cnt=cnt+1
    if(cnt>50):
        val.pop(0)
```

Output:



Practical No. 5

Aim: Controlling Raspberry Pi with Telegram Bot with Raspberry Pi, sending messages.

<u>Requirements:</u>

1. Raspberry Pi connected to Internet
2. A mobile running Telegram Application.

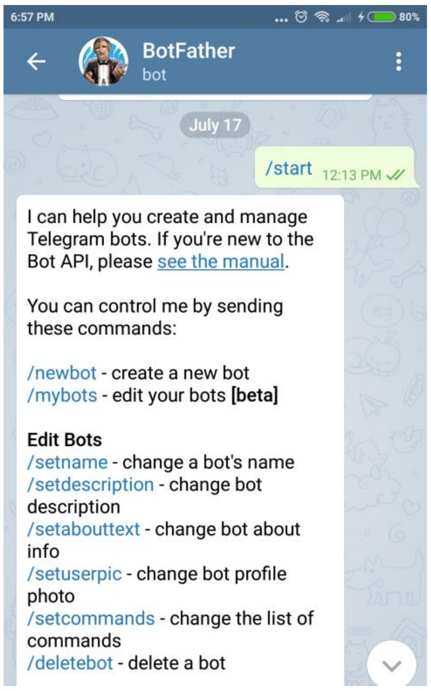
<u>Telegram: -</u> It is a chat based application, very similar to WhatsApp. One special features of this application is that they support bots. Meaning this smart phone application can not only be used by humans but also by machine.

<u>Step 1: Installing Telegram App</u>

a) Installing Telegram app on your PC from web.telegram.org or you can download from your playstore of your mobile.
b) Open an account in telegram by registering your phone number.

<u>Step2: Talk to BotFather</u>

a) The next step would be to request the Bot Father to create us a new Bot.
b) On the top right corner of the Home screen there will be a search icon, click on it to search for the name “botfather”.
c) botfather is a Bot(wizard) which will guide you to create a new bot for you.
d) Click on **start** and select **/newbot** as shown in the picture below. Now, the bot will ask for few details like name of your Bot and the user name of the bot.
e) Fill those details and remember the username as we will needing it in future.


6:57 PM

BotFather
bot

July 17

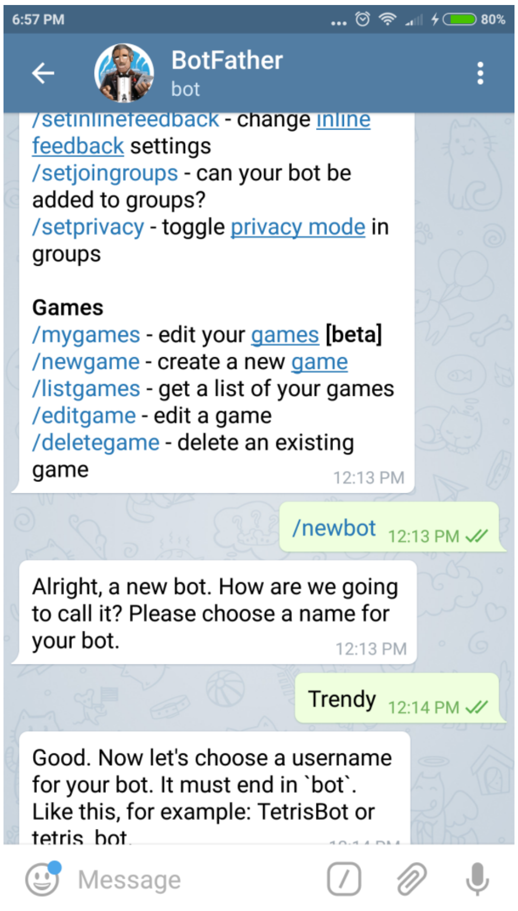
/start 12:13 PM

I can help you create and manage Telegram bots. If you're new to the Bot API, please see the manual.

You can control me by sending these commands:

/newbot - create a new bot
/mybots - edit your bots **[beta]**

Edit Bots
/setname - change a bot's name
/setdescription - change bot description
/setabouttext - change bot about info
/setuserpic - change bot profile photo
/setcommands - change the list of commands
/deletebot - delete a bot


6:57 PM

BotFather
bot

/setinlinefeedback - change inline feedback settings
/setjoingroups - can your bot be added to groups?
/setprivacy - toggle privacy mode in groups

Games
/mygames - edit your games **[beta]**
/newgame - create a new game
/listgames - get a list of your games
/editgame - edit a game
/deletegame - delete an existing game 12:13 PM

/newbot 12:13 PM

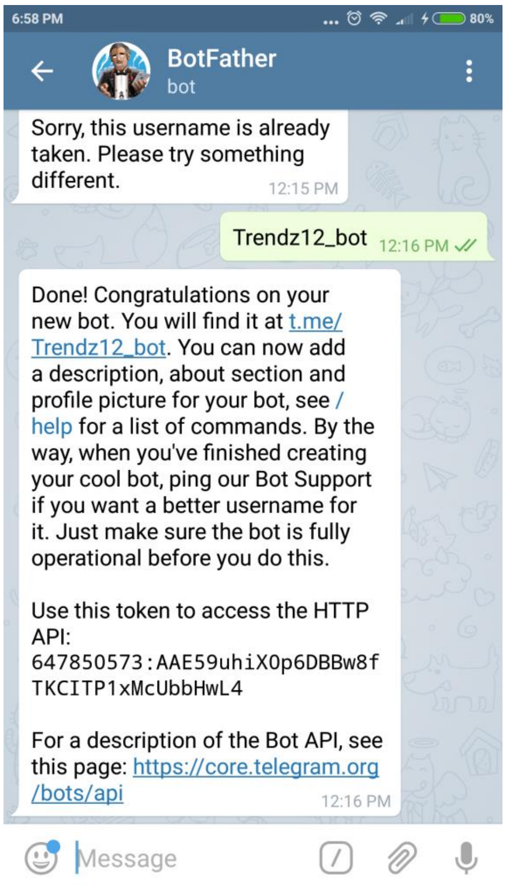
Alright, a new bot. How are we going to call it? Please choose a name for your bot. 12:13 PM

Trendy 12:14 PM

Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot.

<u>Step3: Getting your token for access</u>

I have named bot as Trendy and the username as Trendz12_bot. After this process the botfather will give you a Token for access. This is like the password for your bot, people can control program your bot using this token key.



6:58 PM ... 80%

BotFather
bot

Sorry, this username is already taken. Please try something different. 12:15 PM

Trendz12_bot 12:16 PM

Done! Congratulations on your new bot. You will find it at t.me/Trendz12_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.

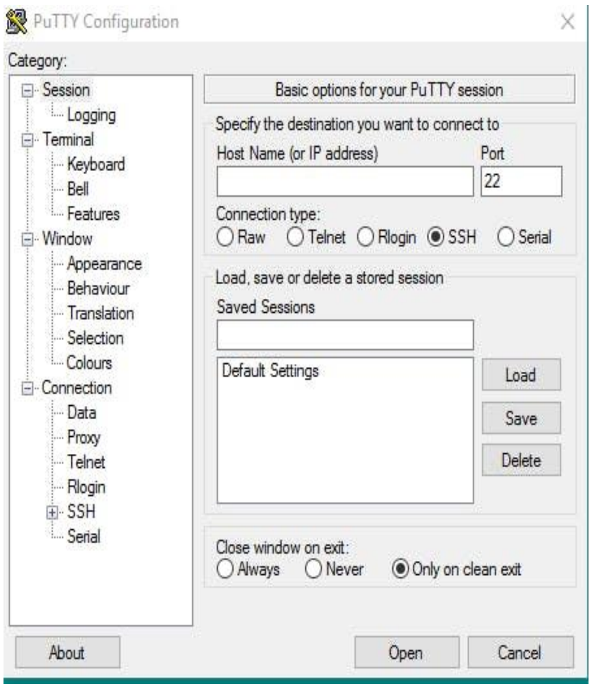
Use this token to access the HTTP API:
647850573:AAE59uhiXOp6DBBw8fTKCITP1xMcUbbHwL4

For a description of the Bot API, see this page: https://core.telegram.org/bots/api 12:16 PM

Step 4: Setup Raspberry Pi

Step 5: Install TelegramBot on Raspberry Pi

5.1 Open Putty



PuTTY Configuration

Category:
- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
  - Serial

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

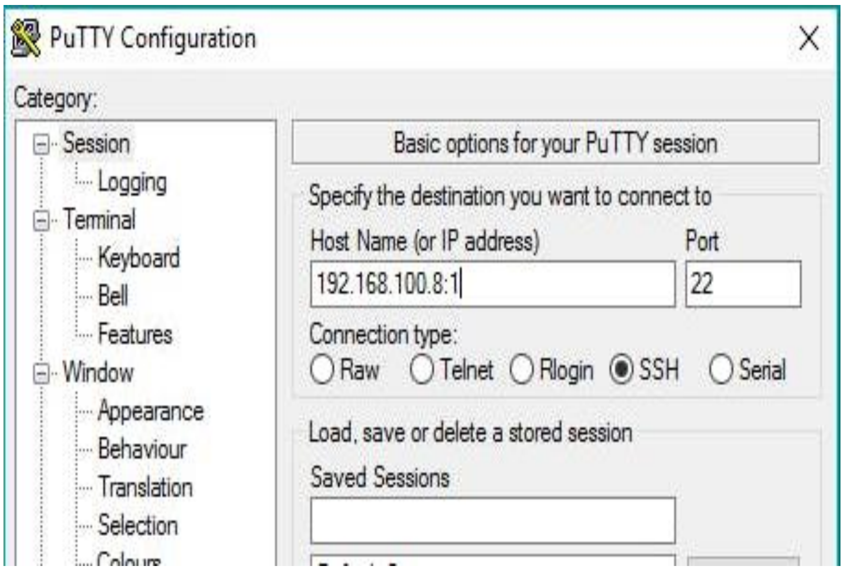
Load, save or delete a stored session

Saved Sessions

Default Settings

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

5.2 Connect Pi via SSH



Specify the destination you want to connect to

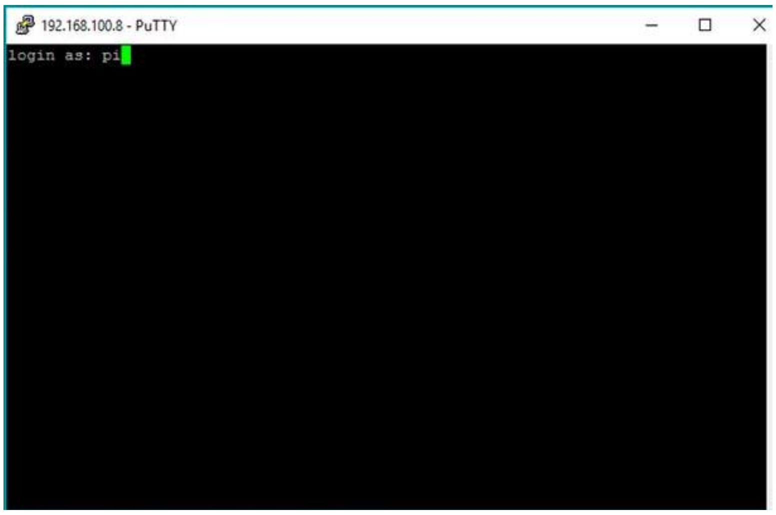
Host Name (or IP address): 192.168.100.8:1

Port: 22

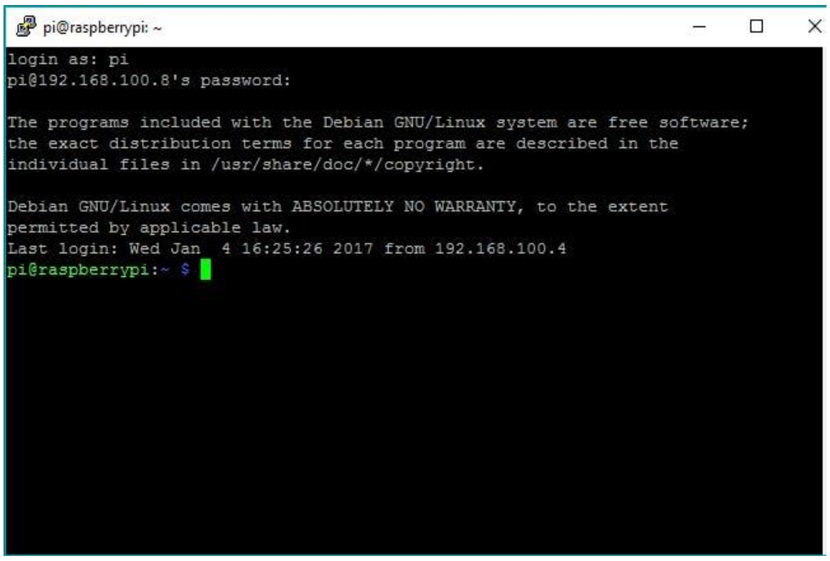
Connection type: ○ Raw ○ Telnet ○ Rlogin [✓ SSH] ○ Serial

Load, save or delete a stored session

Saved Sessions



```
login as: pi
```

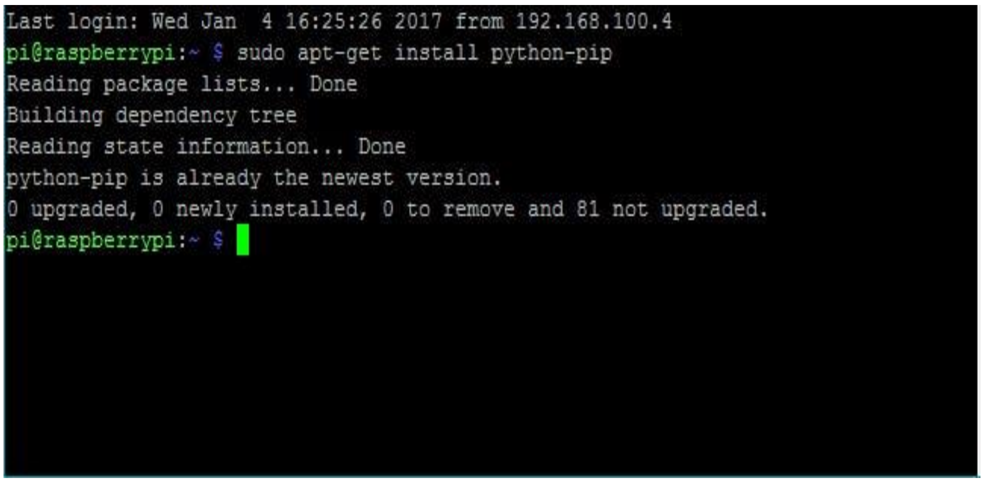


```
login as: pi
pi@192.168.100.8's password:

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jan  4 16:25:26 2017 from 192.168.100.4
pi@raspberrypi:~ $
```

<u>Step 6: Telepot for installing Telegram on Raspberry Pi</u>

Using Telegram Bot in Raspberry Pi is made possible by the python package called Telepot. We need to install this package on Raspberry Pi by using the following commands on Linux terminal.

```
sudo apt-get install python-pip
```



```
Last login: Wed Jan  4 16:25:26 2017 from 192.168.100.4
pi@raspberrypi:~ $ sudo apt-get install python-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-pip is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 81 not upgraded.
pi@raspberrypi:~ $
```

```
sudo pip install telepot
```

Step 7: Run the Python Code

<u>Python Program:</u>

```
import sys
import time
import telepot
import RPi.GPIO as GPIO


#LED
def on(pin):
    GPIO.output(pin,GPIO.HIGH)
    return
def off(pin):
    GPIO.output(pin,GPIO.LOW)
    return

# to use Raspberry Pi board pin numbers
GPIO.setmode(GPIO.BOARD)
# set up GPIO output channel
GPIO.setup(11, GPIO.OUT)

def handle(msg):
    chat_id = msg['chat']['id']
    command = msg['text']

    print('Got command: %s' % command)
    if command == 'on':
        bot.sendMessage(chat_id, on(11))
    elif command =='off':
            bot.sendMessage(chat_id, off(11))

bot = telepot.Bot('Bot Token')
bot.message_loop(handle)
print('I am listening...')

while 1:
    try:
        time.sleep(10)
    except KeyboardInterrupt:
            print('\n Program interrupted')
            GPIO.cleanup()
            exit()
    except:
            print('Other error or exception occured!')
            GPIO.cleanup()
```

7.2 Paste your Bot Token here

```
bot = telepot.Bot('Bot Token')
```

7.3 Run the Code

```
python telegrambot.py
```

All set, now time to connect the Pi and LED.

Practical No: 6

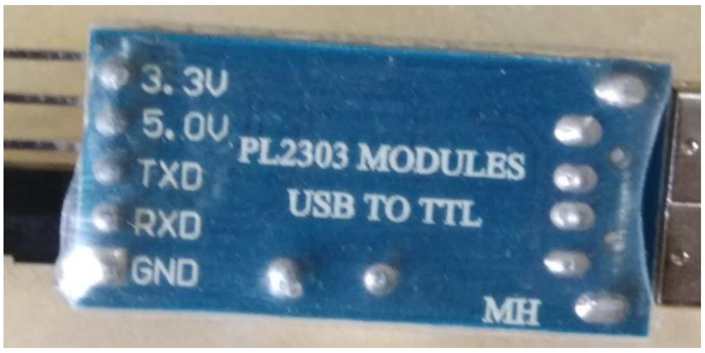
Aim: Fingerprint Sensor interfacing with Raspberry Pi

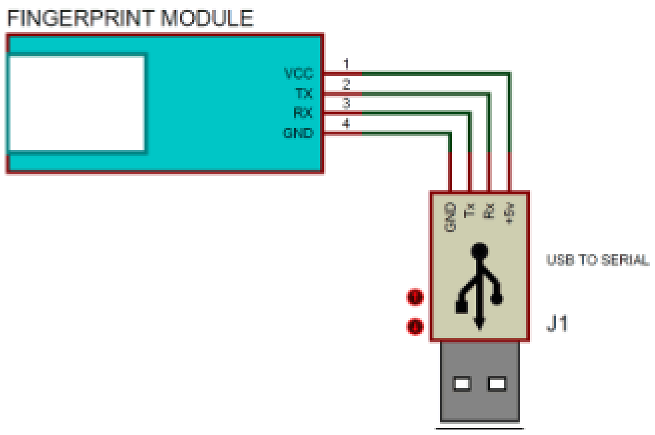
Hardware components required:

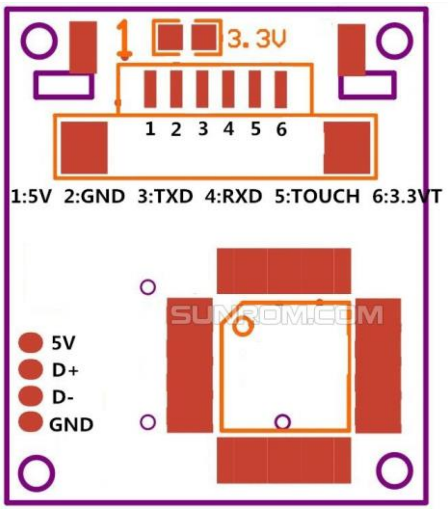
For completing this lesson, you will require the following things along with your initial raspberry pi setup.
1) Fingerprint sensor
2) USB to TTL/UART converter
3) Connecting wires
4) Push Buttons
5) 16x2 LCD
6) LED
7) Breadboard

Circuit Diagram and Explanation:

Fingerprint Sensor: It is an intelligent module which can get fingerprint, process it, verify fingerprint, search and store fingerprints.
Fingerprint processing includes two parts: fingerprint enrolment and fingerprint matching.
To enroll fingerprint, the user needs to enter the finger 2 – 4 times for every finger, process finger images and store generated templates on module.
USB TTL adapter: A USB TTL adapter with 3.3V and 5V voltage output can be used for many serial modules. These sensors can be read via UART. The Raspberry Pi has two pins (pin 8 / GPIO14 and pin 10 / GPIO 15), but they work with 3.3V. Since there are different fingerprint sensors, which do not all work with 3.3V, a USB UART converter is recommended. Some models can be used with both 3.3V and 5V voltage.







Software guide:

After making all the connections we need to power up raspberry Pi and get it ready with terminal open. Now we need to install fingerprint library for Raspberry Pi in python language by following the below steps:

1) To install this library, root privileges are required. So first we enter in root by given command:

```
sudo bash
```

2) Then download some required packages by using given commands:

```
wget -O- http://apt.pm-codeworks.de/pm-codeworks.de.gpg|apt-key add-

wget http://apt.pm-codeworks .de/pm-codeworks.list-P/etc/apt/sources.list.d/
```

3) After this we need to update the Raspberry pi and install the downloaded finger print sensor library:

```
a) sudo apt-get update

b) sudo apt-get install python-fingerprint

c) Now exit root by typing exit
```

4) After installing library now, we need to check USB port on which your finger print sensor is connected, by using given the command:

```
ls/dev/ttyUSB* (or lsusb)
```

5) Now in python code replace the USB port number with the one you got on the screen after executing the command in step 4.
6) Type in following command to download fingerprint code from the following link:

```
https://github.com/bastianraschke/pyfingerprint
```

<u>Python Program:</u>

<u>Enroll.py</u>

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import time
from pyfingerprint.pyfingerprint import PyFingerprint


## Enrolls new finger
## Tries to initialize the sensor
try:
    f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The given fingerprint sensor password is wrong!')

except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Exception message: ' + str(e))
    exit(1)

## Gets some sensor information
print('Currently used templates: ' + str(f.getTemplateCount()) +'/'+ str(f.getStorageCapacity()))

## Tries to enroll new finger
try:
    print('Waiting for finger...')
    ## Wait that finger is read
    while ( f.readImage() == False ):
        pass

    ## Converts read image to characteristics and stores it in charbuffer 1
    f.convertImage(0x01)

    ## Checks if finger is already enrolled
    result = f.searchTemplate()
    positionNumber = result[0]

    if ( positionNumber >= 0 ):
        print('Template already exists at position #' + str(positionNumber))
        exit(0)

    print('Remove finger...')
    time.sleep(2)

    print('Waiting for same finger again...')

    ## Wait that finger is read again
    while ( f.readImage() == False ):
        pass

    ## Converts read image to characteristics and stores it in charbuffer 2
    f.convertImage(0x02)

    ## Compares the charbuffers
    if ( f.compareCharacteristics() == 0 ):
        raise Exception('Fingers do not match')

    ## Creates a template
    f.createTemplate()

    ## Saves template at new position number
    positionNumber = f.storeTemplate()
    print('Finger enrolled successfully!')
    print('New template position #' + str(positionNumber))

except Exception as e:
    print('Operation failed!')
    print('Exception message: ' + str(e))
    exit(1)
```

SEARCH CODING:

search.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import hashlib
from pyfingerprint.pyfingerprint import PyFingerprint
```

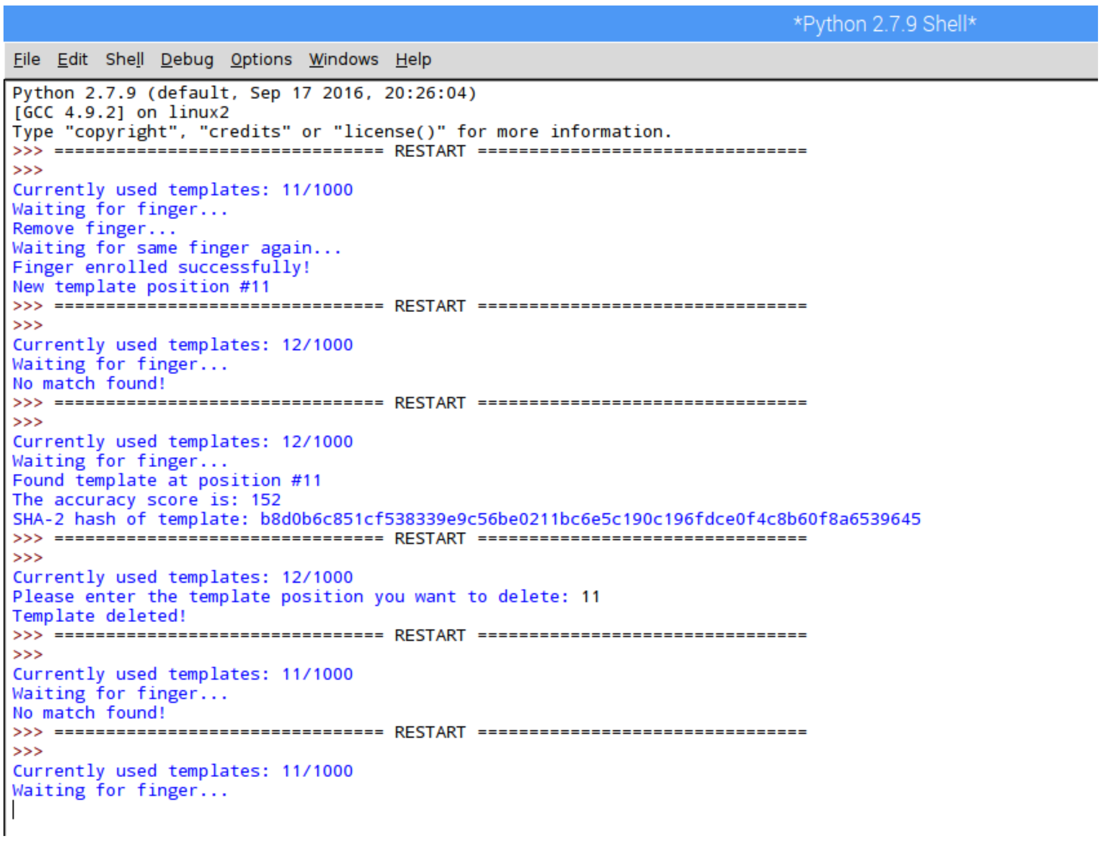
```
## Search for a finger
##
## Tries to initialize the sensor
try:
    f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The given fingerprint sensor password is wrong!')

except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Exception message: ' + str(e))
    exit(1)

## Gets some sensor information
print('Currently used templates: ' + str(f.getTemplateCount()) +'/'+ str(f.getStorageCapacity()))

## Tries to search the finger and calculate hash
try:
    print('Waiting for finger...')

    ## Wait that finger is read
    while ( f.readImage() == False ):
        pass

    ## Converts read image to characteristics and stores it in charbuffer 1
    f.convertImage(0x01)

    ## Searchs template
    result = f.searchTemplate()

    positionNumber = result[0]
    accuracyScore = result[1]

    if ( positionNumber == -1 ):
        print('No match found!')
        exit(0)
    else:
        print('Found template at position #' + str(positionNumber))
        print('The accuracy score is: ' + str(accuracyScore))


    ## OPTIONAL stuff
    ##

    ## Loads the found template to charbuffer 1
    f.loadTemplate(positionNumber, 0x01)

    ## Downloads the characteristics of template loaded in charbuffer 1
    characterics = str(f.downloadCharacteristics(0x01)).encode('utf-8')

    ## Hashes characteristics of template
    print('SHA-2 hash of template: ' + hashlib.sha256(characterics).hexdigest())

except Exception as e:
    print('Operation failed!')
    print('Exception message: ' + str(e))
    exit(1)
```

DELETE CODING:

delete.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pyfingerprint.pyfingerprint import PyFingerprint


## Deletes a finger from sensor
##
## Tries to initialize the sensor
try:
    f = PyFingerprint('/dev/ttyUSB0', 57600, 0xFFFFFFFF, 0x00000000)

    if ( f.verifyPassword() == False ):
        raise ValueError('The given fingerprint sensor password is wrong!')

except Exception as e:
    print('The fingerprint sensor could not be initialized!')
    print('Exception message: ' + str(e))
    exit(1)

## Gets some sensor information
print('Currently used templates: ' + str(f.getTemplateCount()) +'/'+ str(f.getStorageCapacity()))
## Tries to delete the template of the finger
try:
    positionNumber = input('Please enter the template position you want to delete: ')
    positionNumber = int(positionNumber)

    if ( f.deleteTemplate(positionNumber) == True ):
        print('Template deleted!')

except Exception as e:
    print('Operation failed!')
    print('Exception message: ' + str(e))
    exit(1)
```

OUTPUT:



```
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Currently used templates: 11/1000
Waiting for finger...
Remove finger...
Waiting for same finger again...
Finger enrolled successfully!
New template position #11
>>> ================================ RESTART ================================
>>>
Currently used templates: 12/1000
Waiting for finger...
No match found!
>>> ================================ RESTART ================================
>>>
Currently used templates: 12/1000
Waiting for finger...
Found template at position #11
The accuracy score is: 152
SHA-2 hash of template: b8d0b6c851cf538339e9c56be0211bc6e5c190c196fdce0f4c8b60f8a6539645
>>> ================================ RESTART ================================
>>>
Currently used templates: 12/1000
Please enter the template position you want to delete: 11
Template deleted!
>>> ================================ RESTART ================================
>>>
Currently used templates: 11/1000
Waiting for finger...
No match found!
>>> ================================ RESTART ================================
>>>
Currently used templates: 11/1000
Waiting for finger...
```

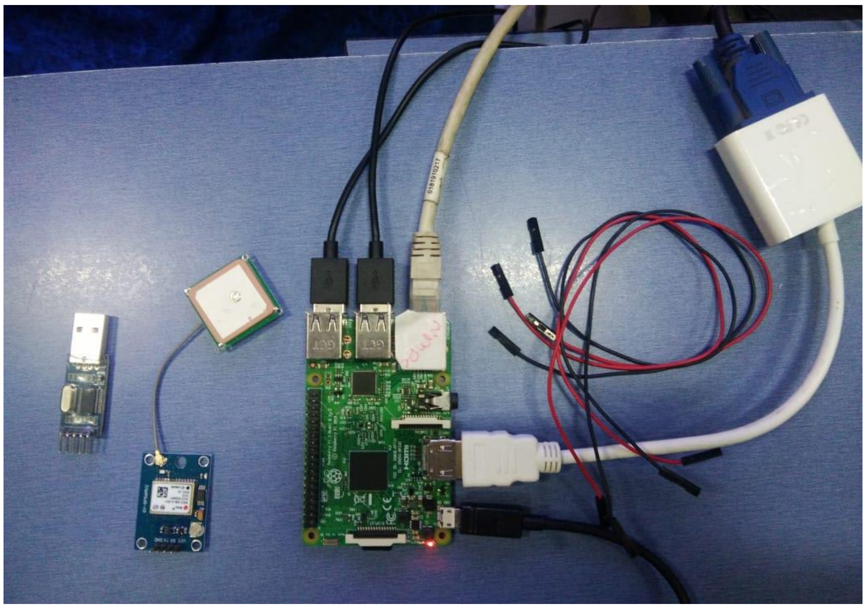
Practical No. 7

Aim: GPS Module Interfacing with Raspberry Pi

Hardware Guide:

You will require the following things with your initial raspberry pi setup

1. GPS Module
2. USB to TTL converter
3. Connecting wires



GPS Module

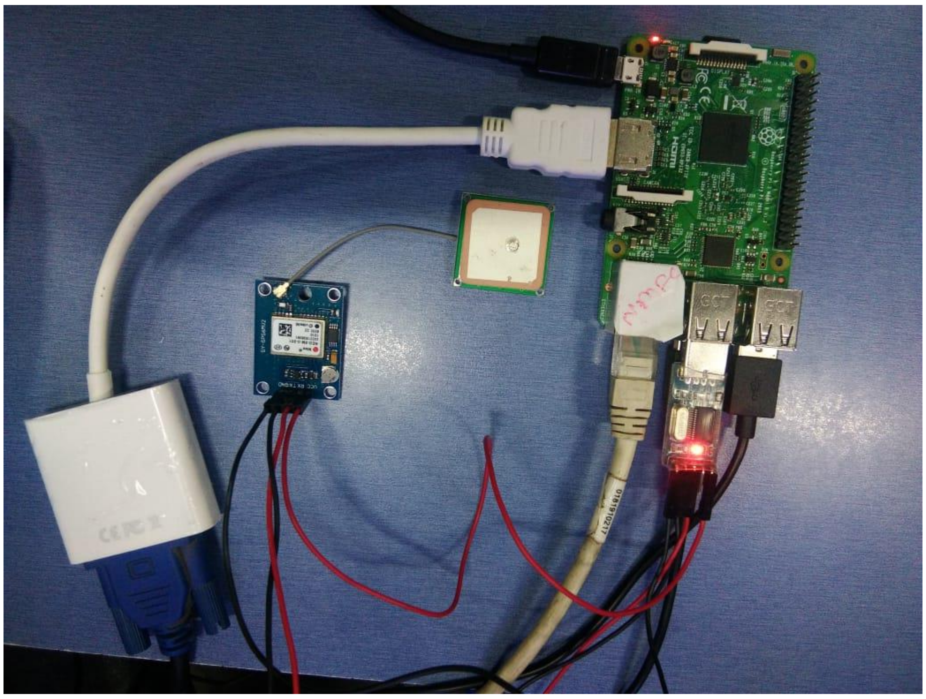
Global Positing System (GPS) makes use of signals sent by satellites in space and ground stations on earth to accurately determine their position on earth. Radio Frequency signals sent from satellites and ground stations contain time stamps of the time when the signals were transmitted. Using information from 3 or more satellites the exact position of the GPS can be triangulated. GPS stands for Global Positioning System and used to detect the Latitude and Longitude of any location on the Earth, with exact UTC time (Universal Time Coordinated). This device receives the coordinates from the satellite for each and every second, with time and date. **GPS module** sends the data related to tracking position in real time, and it sends so many data in NMEA format (see the screenshot below). NMEA format consist several sentences, in which we only need one sentence. This sentence starts from **$GPGGA** and contains the coordinates, time and other useful information. This **GPGGA** is referred to **Global Positioning System Fix Data**.

We can extract coordinate from $GPGGA string by counting the commas in the string. Suppose you find $GPGGA string and stores it in an array, then Latitude can be found.

Identifier	**Description**
$GPGGA	Global Positioning system fix data
HHMMSS.SSS	Time in hour minute seconds and milliseconds format.
Latitude	Latitude (Coordinate)
N	Direction N=North, S=South
Longitude	Longitude(Coordinate)
E	Direction E= East, W=West
FQ	Fix Quality Data
NOS	No. of Satellites being Used
HPD	Horizontal Dilution of Precision
Altitude	Altitude from sea level
M	Meter
Height	Height
Checksum	Checksum Data

CONNECT YOUR CIRCUIT:

1. Connect the VCC pin of GPS Module to 3.3V pin of USB to TTL converter.
2. Connect the GND pin of GPS Module to GND pin of USB to TTL converter.
3. Connect the Tx pin of GPS Module to Rx pin of USB to TTL converter.
4. Connect the Rx pin of GPS Module to Tx pin of USB to TTL converter.
5. Lastly, connect the USB to TTL converter to USB port of Raspberry Pi.



Software Guide:

Open Terminal Window and type the following command to know to which USB port the GPS module is attached:

```
ls /dev/ttyUSB*
```

We can find whether our GPS module is working properly and the connections are correct by typing the following command

```
sudo cat /dev/ttyUSB*
```

Use 'gpsd'

You can always just read that raw data, but it's much better if you can have some Linux software prettify it. We'll try gpsd which gpsd-handling Daemon (background-helper)

Installing GPS Daemon (gpsd)

The first step is installing some software on Raspberry Pi that understands the serial data that your GPS module is providing via /dev/ttyUSB0.

To install gpsd, make sure your pi has an internet connection and run the following command from the console

```
sudo apt-get update
sudo apt-get install gpsd gpsd-clients python-gps
```

And install the software as it prompts you to do.

Raspbian systemd service fix:

If you are using the Raspbian Jessie or later release you'll need to disable the system services that gpsd installs. This service has system listen on a local socket and run gpsd when client connects to it, however it will also interfere with other gpsd instances that are manually run. You'll need to disable the gpsd systemd by running the following commands.

```
sudo systemctl stop gpsd.socket
sudo systemctl disable gpsd.socket
```

If you want to enable the default gpsd system service you can run this command to restore it.

```
sudo systemctl enable gpsd.socket
sudo systemctl start gpsd.socket
```

Try out ‘gpsd’

After installing and disabling the gpsd systemd service as mentioned above you are ready to start using gpsd yourself.

Start gpsd and direct it to use USB simply entering the command

```
sudo gpsd /dev/ttyUSB0 –F /var/run/gpsd.sock
```

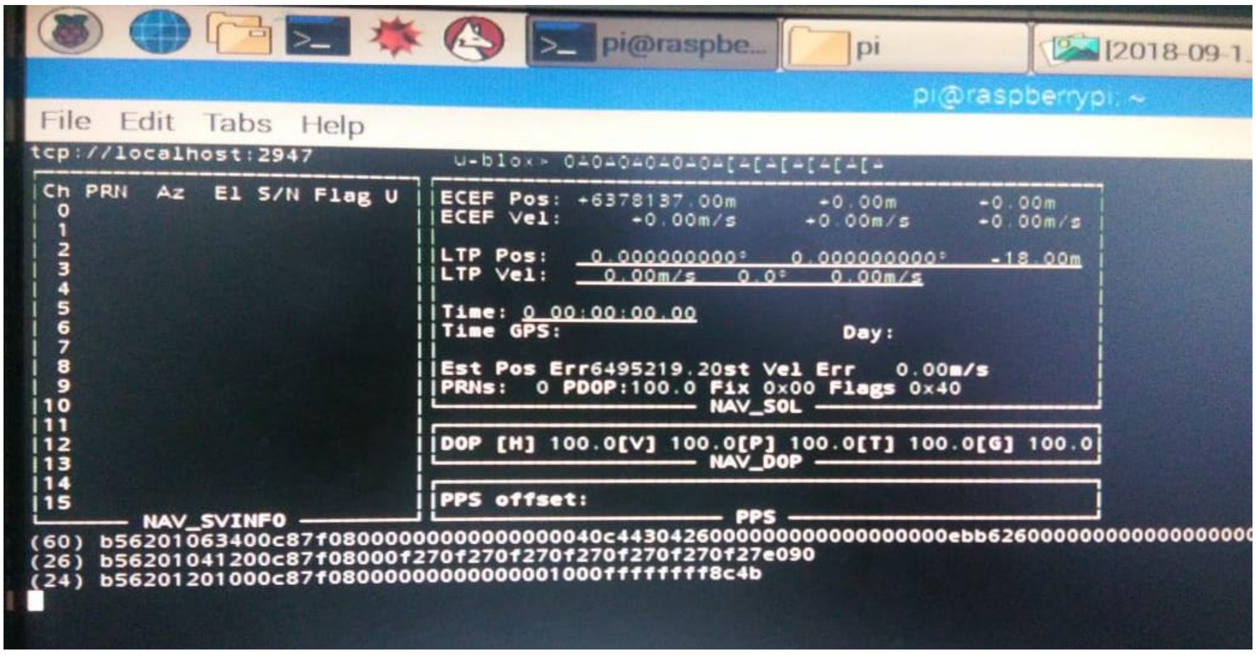
which willpoint the gps daemon to our gps device on the /dev/ttyAMA0 console

Try running gpsmon to get a live streaming update of GPS data! Or cgps which gives a less detailed but still quite nice output

1. cgps –s

You can abort gpsd by the following command

1. sudo killall gpsd



```
tcp://localhost:2947                u-blox> 0A0A0A0A0A0A[A[A[A[A[A[A
|Ch PRN  Az  El S/N Flag U ||ECEF Pos: +6378137.00m      +0.00m       +0.00m
  0                        ||ECEF Vel:      +0.00m/s     +0.00m/s     +0.00m/s
  1                        ||
  2                        ||LTP Pos:  0.000000000°   0.000000000°    -18.00m
  3                        ||LTP Vel:   0.00m/s   0.0°   0.00m/s
  4                        ||
  5                        ||Time: 0 00:00:00.00
  6                        ||Time GPS:                       Day:
  7                        ||
  8                        ||Est Pos Err6495219.20st Vel Err   0.00m/s
  9                        ||PRNs:  0 PDOP:100.0 Fix 0x00 Flags 0x40
|10                        |------------------ NAV_SOL ---------
|11                        |
|12                        ||DOP [H] 100.0[V] 100.0[P] 100.0[T] 100.0[G] 100.0|
|13                        |------------------ NAV_DOP ---------
|14                        |
|15                        ||PPS offset:
------- NAV_SVINFO --------------------------- PPS ------------
(60) b56201063400c87f08000000000000000040c44304260000000000000000000ebb6260000000000000000000
(26) b56201041200c87f08000f270f270f270f270f270f270f27e090
(24) b56201201000c87f080000000000000001000ffffffff8c4b
█
```

Download or copy the code from the following link:

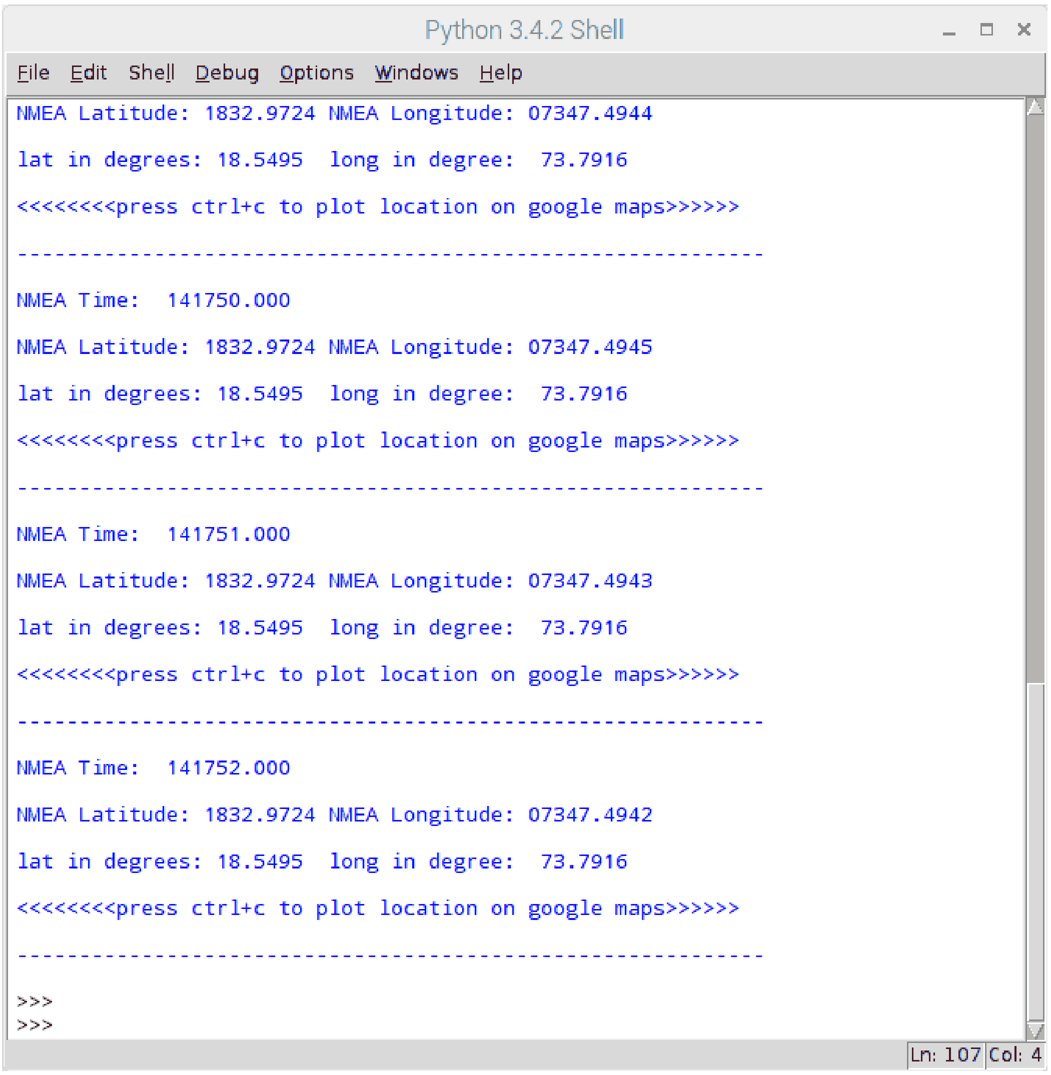
```
http://www.electronicwings.com/raspberry-pi/gps-module-interfacing-with-raspberry-pi
```

Python Code:

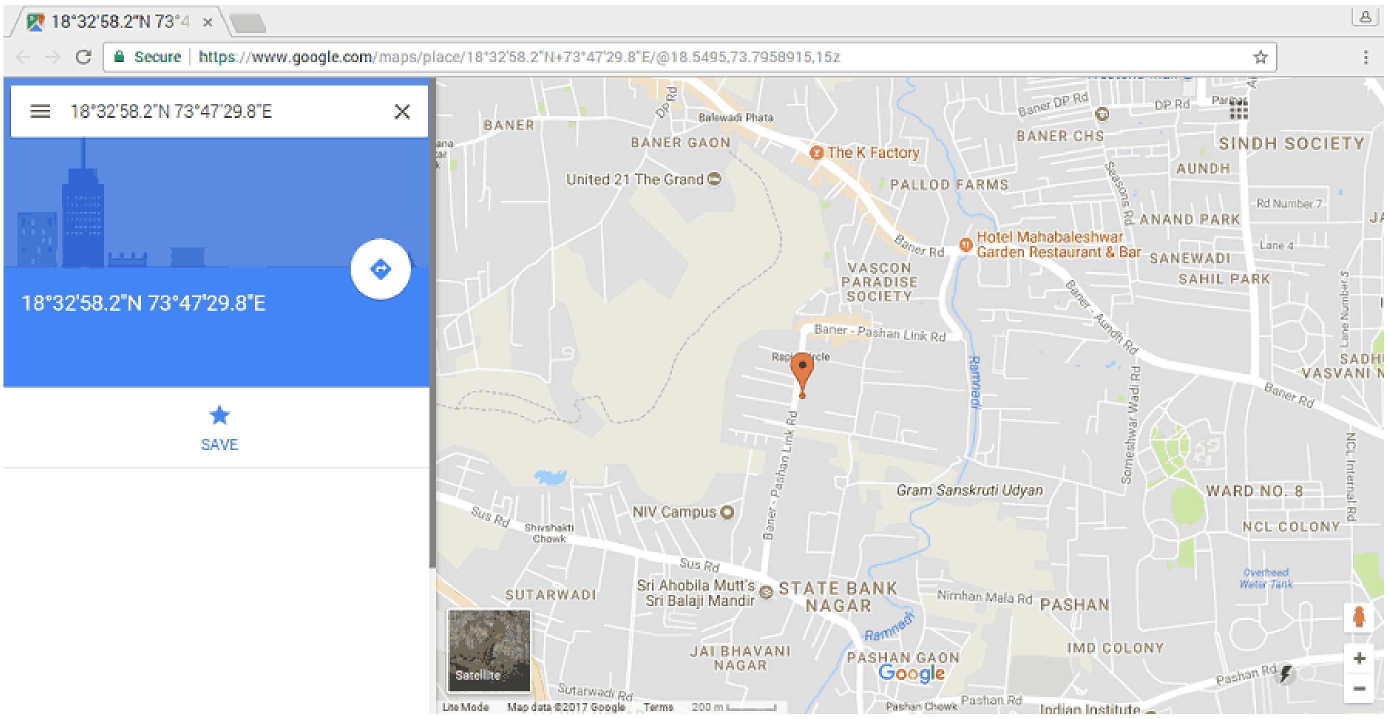
```
import serial               #import serial pacakge
from time import sleep
import webbrowser           #import package for opening link in browser
import sys                  #import system package

def GPS_Info():
   global NMEA_buff
   global lat_in_degrees
   global long_in_degrees
   nmea_time = []
   nmea_latitude = []
   nmea_longitude = []
   nmea_time = NMEA_buff[0]                #extract time from GPGGA string
   nmea_latitude = NMEA_buff[1]            #extract latitude from GPGGA string
   nmea_longitude = NMEA_buff[3]           #extract longitude from GPGGA string

   print("NMEA Time: ", nmea_time,'\n')
   print ("NMEA Latitude:", nmea_latitude,"NMEA Longitude:", nmea_longitude,'\n')

   lat = float(nmea_latitude)              #convert string into float for calculation
   longi = float(nmea_longitude)           #convertr string into float for calculation

   lat_in_degrees = convert_to_degrees(lat)    #get latitude in degree decimal format
   long_in_degrees = convert_to_degrees(longi) #get longitude in degree decimal format

#convert raw NMEA string into degree decimal format
def convert_to_degrees(raw_value):
   decimal_value = raw_value/100.00
   degrees = int(decimal_value)
   mm_mmmm = (decimal_value - int(decimal_value))/0.6
   position = degrees + mm_mmmm
   position = "%.4f" %(position)
   return position


gpgga_info = "$GPGGA,"
ser = serial.Serial ("/dev/ttyS0")    #Open port with baud rate
GPGGA_buffer = 0
NMEA_buff = 0
lat_in_degrees = 0
long_in_degrees = 0

try:
   while True:
      received_data = (str)(ser.readline())             #read NMEA string received
```

```
        GPGGA_data_available = received_data.find(gpgga_info)   #check for NMEA GPGGA string
        if (GPGGA_data_available>0):
            GPGGA_buffer = received_data.split("$GPGGA,",1)[1]  #store data coming after "$GPGGA," string
            NMEA_buff = (GPGGA_buffer.split(','))               #store comma separated data in buffer
            GPS_Info()                                          #get time, latitude, longitude
            print("lat in degrees:", lat_in_degrees," long in degree: ", long_in_degrees, '\n')
            map_link = 'http://maps.google.com/?q=' + lat_in_degrees + ',' + long_in_degrees    #create link to plot location on Google map
            print("<<<<<<<<press ctrl+c to plot location on google maps>>>>>>\n")               #press ctrl+c to plot on map and exit
            print("------------------------------------------------------------\n")

except KeyboardInterrupt:
    webbrowser.open(map_link)        #open current position information in google map
    sys.exit(0)
```

Output:



```
NMEA Latitude: 1832.9724 NMEA Longitude: 07347.4944

lat in degrees: 18.5495  long in degree:  73.7916

<<<<<<<<press ctrl+c to plot location on google maps>>>>>>

------------------------------------------------------------

NMEA Time:  141750.000

NMEA Latitude: 1832.9724 NMEA Longitude: 07347.4945

lat in degrees: 18.5495  long in degree:  73.7916

<<<<<<<<press ctrl+c to plot location on google maps>>>>>>

------------------------------------------------------------

NMEA Time:  141751.000

NMEA Latitude: 1832.9724 NMEA Longitude: 07347.4943

lat in degrees: 18.5495  long in degree:  73.7916

<<<<<<<<press ctrl+c to plot location on google maps>>>>>>

------------------------------------------------------------

NMEA Time:  141752.000

NMEA Latitude: 1832.9724 NMEA Longitude: 07347.4942

lat in degrees: 18.5495  long in degree:  73.7916

<<<<<<<<press ctrl+c to plot location on google maps>>>>>>

------------------------------------------------------------

>>>
>>>
```

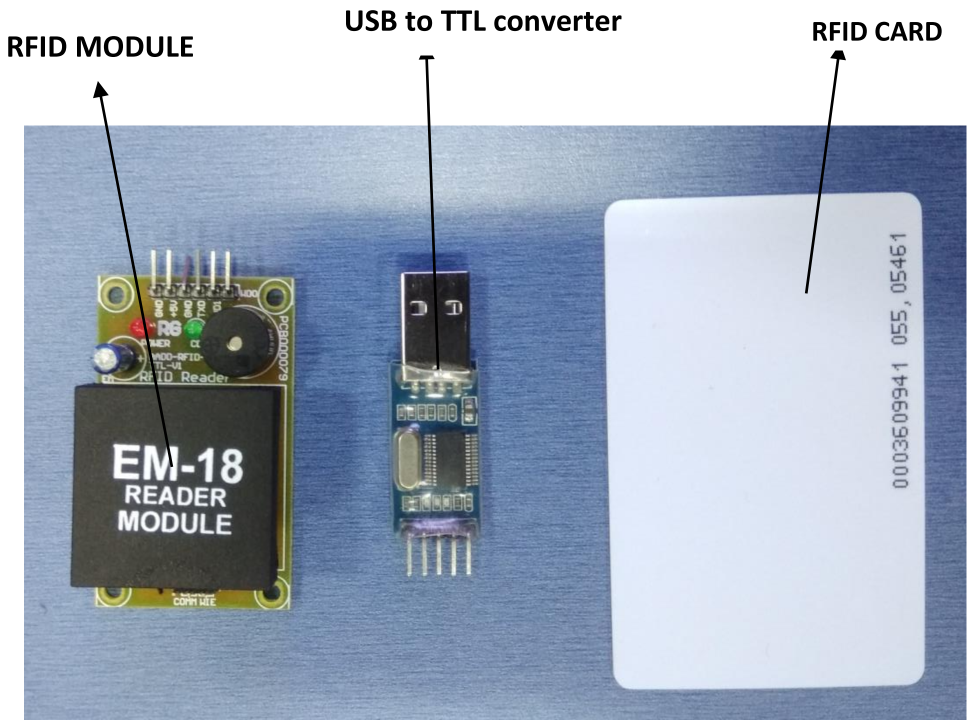

18°32'58.2"N 73°4
Secure
https://www.google.com/maps/place/18°32'58.2"N+73°47'29.8"E/@18.5495,73.7958915,15z
18°32'58.2"N 73°47'29.8"E
18°32'58.2"N 73°47'29.8"E
SAVE
BANER
BANER GAON
The K Factory
United 21 The Grand
PALLOD FARMS
BANER CHS
SINDH SOCIETY
AUNDH
Hotel Mahabaleshwar Garden Restaurant & Bar
ANAND PARK
SANEWADI
SAHIL PARK
VASCON PARADISE SOCIETY
Baner - Pashan Link Rd
Gram Sanskruti Udyan
WARD NO. 8
NCL COLONY
NIV Campus
Sus Rd
SUTARWADI
Sri Ahobila Mutt's Sri Balaji Mandir
STATE BANK NAGAR
Nimhan Mala Rd
PASHAN
JAI BHAVANI NAGAR
PASHAN GAON
IMD COLONY
Pashan Rd
Satellite
Map data ©2017 Google

Practical No. 8

Aim: Interfacing Raspberry Pi with RFID

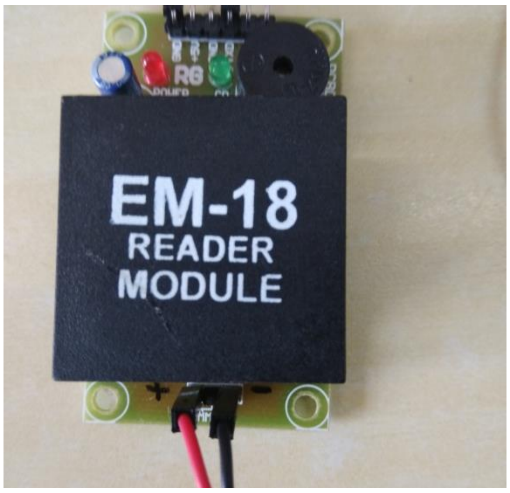
Components required:

1. RFID MODULE
2. USB to TTL converter
3. Connecting wires



Connections:

1. Connect TX pin of Module to Rx Pin of USB to TTL converter.
2. Connect the GND Pin of Module to GND Pin of USB to TTL converter.
3. Connect the positive of 5V external supply to VCC pin of module.

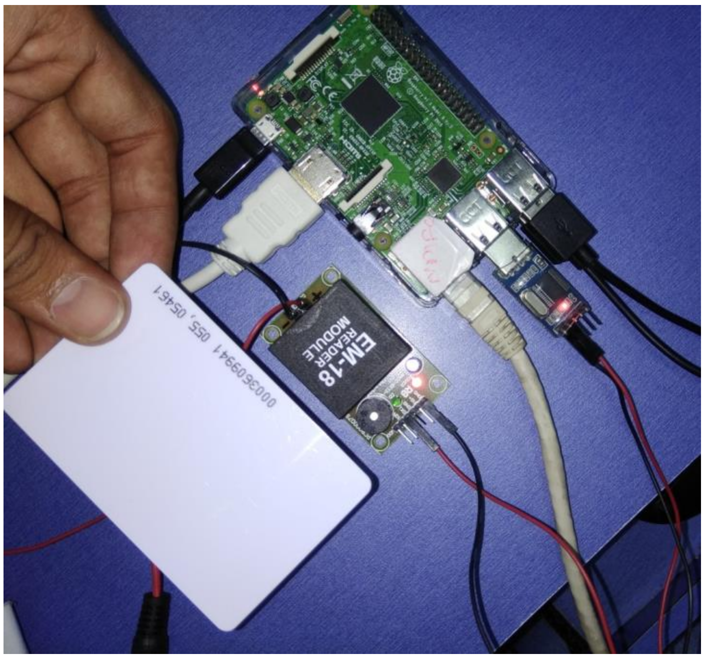


Python Code:

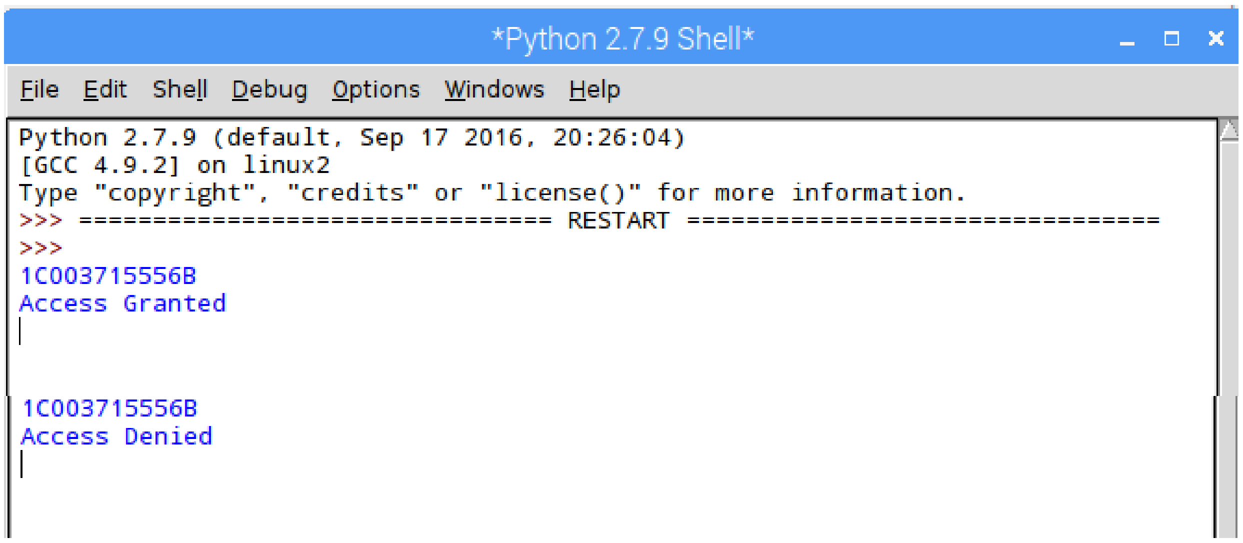
```
import RPi.GPIO as GPIO
import time
import serial

GPIO.setmode(GPIO.BOARD)
greenLED=37
redLED=35
buzzer=33

GPIO.setup(greenLED,GPIO.OUT)
GPIO.setup(redLED,GPIO.OUT)
GPIO.setup(buzzer,GPIO.OUT)

GPIO.output(greenLED,False)
GPIO.output(redLED,False)
GPIO.output(buzzer,True)

time.sleep(0.1)
GPIO.output(buzzer,False)
time.sleep(0.1)

GPIO.output(buzzer,True)
time.sleep(0.1)

GPIO.output(buzzer,False)
time.sleep(0.1)

def read_rfid():
    ser=serial.Serial("/dev/ttyUSB0")
    ser.baudrate=9600
    data=ser.read(12)
    ser.close()
    return data
```

```
try:
    while True:
        id=read_rfid()
        print(id)

        if id=="400034E165F0":
            print("Access Granted")
            GPIO.output(greenLED,True)
            GPIO.output(redLED,False)
            GPIO.output(buzzer,False)
            time.sleep(2)
        else:
            print("Access Denied")
            GPIO.output(greenLED,False)
            GPIO.output(redLED,True)
            GPIO.output(buzzer,True)
            time.sleep(2)
        GPIO.output(greenLED,False)
        GPIO.output(redLED,False)
        GPIO.output(buzzer,False)
finally:
    GPIO.cleanup()
```

Interfacing:



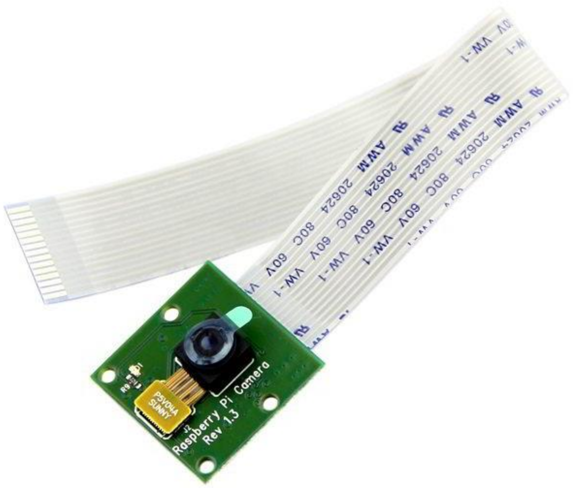
OUTPUT:



```
Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
1C003715556B
Access Granted


1C003715556B
Access Denied
```

Practical No: 10

Aim: Capturing Images with Raspberry Pi and Pi Camera



Pi Camera module is a camera which can be used to take pictures and high definition video.
Raspberry Pi Board has CSI (Camera Serial Interface) interface to which we can attach PiCamera module directly.
This Pi Camera module can attach to the Raspberry Pi's CSI port using 15-pin ribbon cable.

Features of Pi Camera

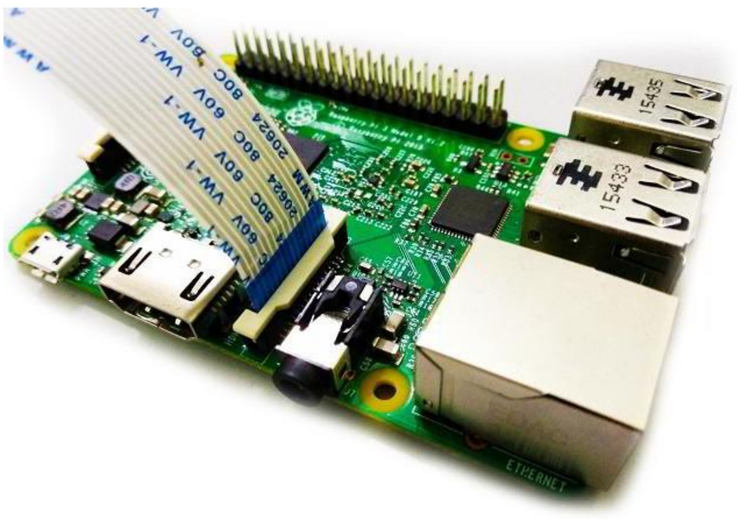
Pi camera v1.3 features are listed below,

- Resolution – 5 MP
- HD Video recording – 1080p @30fps, 720p @60fps, 960p @45fps and so on.
- It can capture wide, still (motionless) images of resolution 2592x1944 pixels
- CSI Interface enabled.

Connect the Camera Module to Raspberry Pi

With the Pi switched off, connect the Camera module to the Raspberry Pi's camera port, then start up the Pi and ensure the software is enabled.

Connect Pi Camera to CSI interface of Raspberry Pi board as shown below,



Now, we can use Pi Camera for capturing images and videos using Raspberry Pi.

How to Enable Camera functionality on Raspberry Pi

For enabling camera in Raspberry Pi, open raspberry pi configuration using following command,

```
sudo raspi-config
```

then select **Interfacing options** in which select **camera** option to enable its functionality. Reboot Raspberry Pi.

You can capture an image by just typing a single line command. Open terminal window and type the command as follows:

```
$ sudo raspistill –o /home/pi/Desktop/image.jpg
```

Capture images and save it to the specified directory

Python Program for Image Capture

```
#import time and picamera library
import picamera
from time import sleep

#create object for PiCamera class
camera = picamera.PiCamera()

#set resolution
camera.resolution = (1280, 720)
camera.brightness = 60
camera.start_preview()

#Camera warmup time
sleep(2)

#store image
camera.capture('/home/pi/Pictures/image1.jpeg')
camera.stop_preview()
```

Python Program for Video Recording

```
import picamera
from time import sleep

camera = picamera.PiCamera()
camera.resolution = (1280, 720)
camera.start_preview()

#start recording using pi camera
camera.start_recording("/home/pi/demo.h264")
sleep(20)
```

```
#stop recording
camera.stop_recording()
camera.stop_preview()
```

Play Recorded Video

To open video, we can use omxplayer by using following command,

```
omxplayer video_name
```

Practical No. 10

Aim: Installing Windows 10 IOT Core on Raspberry Pi

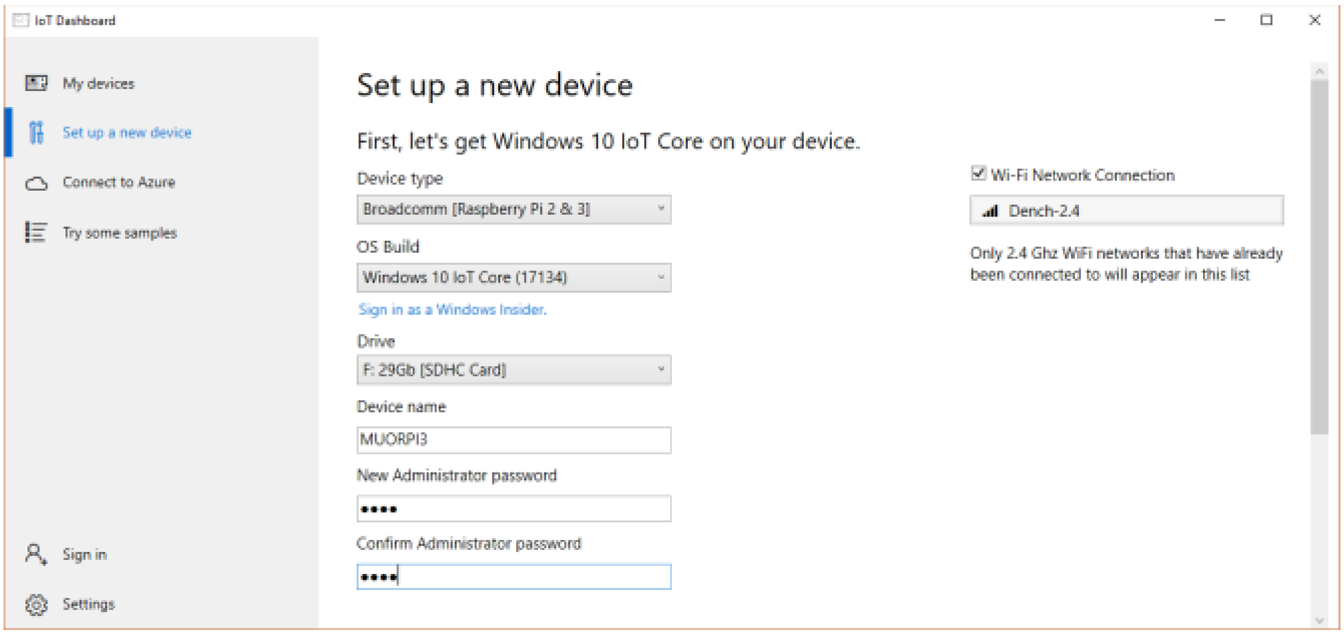
Windows 10 IoT is a member of the Windows 10 family that brings enterprise-class power, security and manageability to the Internet of Things. It leverages Windows embedded experience, ecosystem and cloud connectivity, allowing organizations to create their Internet of Things with secure devices that be quickly provisioned, easily managed, and seamlessly connected to an overall cloud strategy.

Components Used:

1. Raspberry Pi 3
2. 16 GB Micro SD Card-class 10
3. Display
4. Keyboard
5. Mouse
6. Windows 10 PC
7. Card Reader

Procedure:

1. Download the Windows 10 IoT Core Dashboard from https://developer.microsoft.com /enus/windows/iot/Downloads
2. Once downloaded, open the Dashboard and click on set up a new device and insert a SD card into your computer.
3. Fill out all of the fields as indicated.
4. Accept the software license terms and click Download and install. You'll see that windows 10 IoT Core is now flashing onto your device.



Set up a new device

First, let's get Windows 10 IoT Core on your device.

Device type: Broadcomm [Raspberry Pi 2 & 3]

OS Build: Windows 10 IoT Core (17134)

Sign in as a Windows Insider.

Drive: F: 29Gb [SDHC Card]

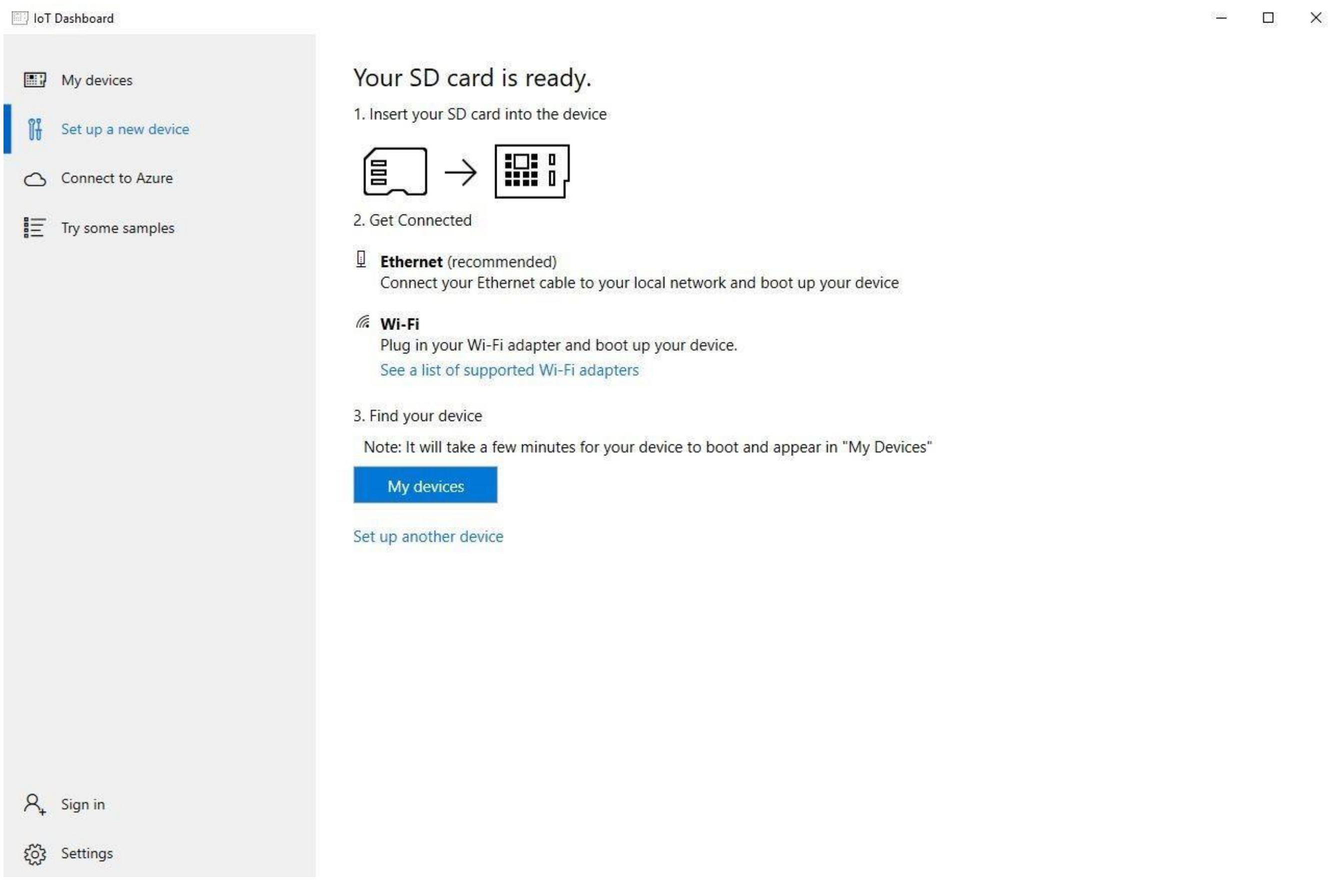
Device name: MUORPI3

New Administrator password: ••••

Confirm Administrator password: ••••

☑ Wi-Fi Network Connection: Dench-2.4

Only 2.4 Ghz WiFi networks that have already been connected to will appear in this list



Your SD card is ready.

1. Insert your SD card into the device

2. Get Connected

Ethernet (recommended)
Connect your Ethernet cable to your local network and boot up your device

Wi-Fi
Plug in your Wi-Fi adapter and boot up your device.
See a list of supported Wi-Fi adapters

3. Find your device

Note: It will take a few minutes for your device to boot and appear in "My Devices"

My devices

Set up another device

Once the image has been installed on the microSD card, it's time to eject it from your PC and go over to the Raspberry Pi. First connect up the micro USB cable and power supply, HDMI cable and USB WiFi adapter or Ethernet cable. Connect the HDMI cable to your chosen display, insert the microSD card into the Raspberry Pi and power it up.