PRACTICAL NO.	TOPIC	DATE	SIGN
1.	Display Single led-blinking pattern using Raspberry Pi.	25/7/22	
2.	Display Three led-blinking pattern using Raspberry Pi.	1/8/22	
3.	Display Four led-blinking pattern using Raspberry Pi	8/8/22	
4.	Display Time over 4-digit 7 Segment using Raspberry Pi.	22/8/22	
5.	Raspberry Pi based Oscilloscope.	29/8/22	
6.	Controlling Raspberry Pi with Telegram App.	5/9/22	
7.	Fingerprint Sensor interfacing with Raspberry Pi.	12/9/22	
8.	Interfacing GPS module with Raspberry Pi.	19/9/22	
9.	Interface Raspberry Pi with Pi Camera.	26/9/22	
10.	Interfacing Raspberry Pi with RFID.	3/10/22	

Prepare and Setup RASPBERRY PI

Hardware preparation and Installation

Hardware Requirements

- A Raspberry Pi Model A/B/B+
- A good quality, micro USB power supply that can provide at least 700mA at 5V
- 8 GB Class 10 SD card (or better) and a Card Reader
- An Ethernet cable / Wi-Fi
- USB Keyboard and Mouse
- HDMI video cable to connect with HDMI-capable monitor/television Screen OR
- HDMI to VGA connector to Connect with VGA-capable monitor

Software Requirements

- Raspbian image
- SD Card formatter
- Windows32 Disk Imager

Pre-requisites

- Download SD card formatter Application

  https://sd-card-formatter.en.uptodown.com/windows
- Download Windows32 Disk Imager Application

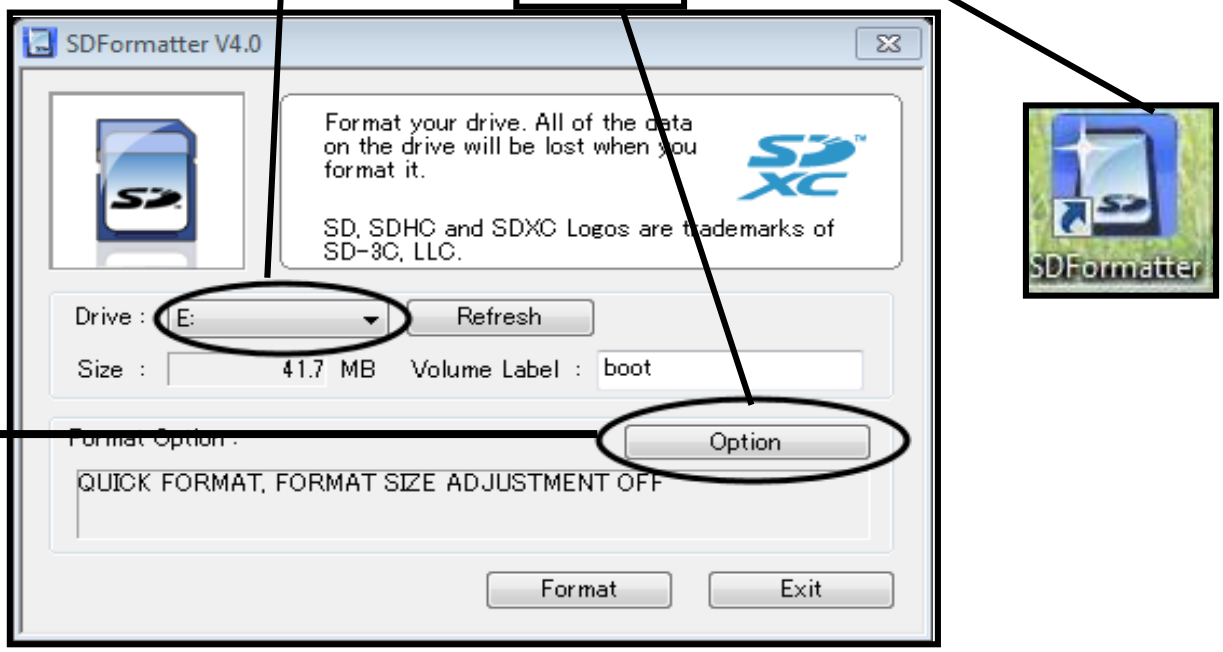
  https://sourceforge.net/projects/win32diskimager/

Step 1: Prepare Your SD Card

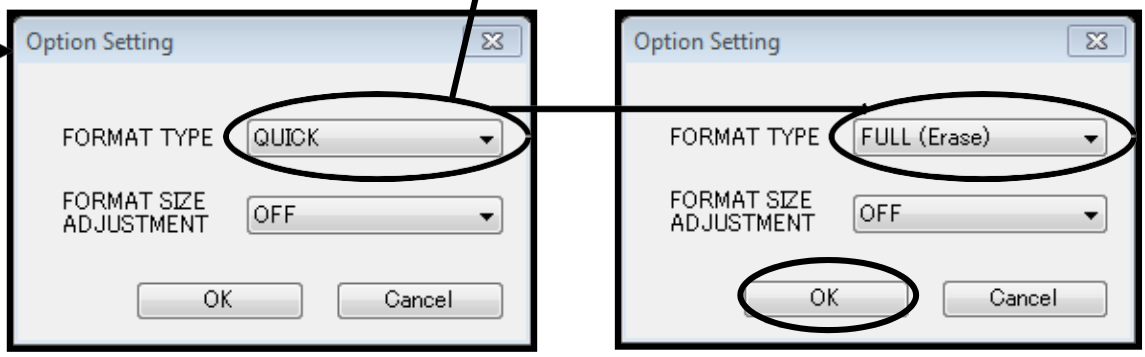
1. Format SD card using SD card formatter software
2. Download the latest version of Raspbian and unzip the .img file
3. Make bootable image for Raspbian OS using Win32DiskImager software

1) Format your SD card using **SDFormatter**.

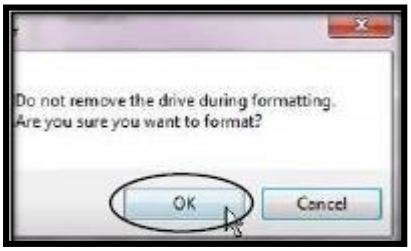
2) Select the **drive** & click on **Option**.



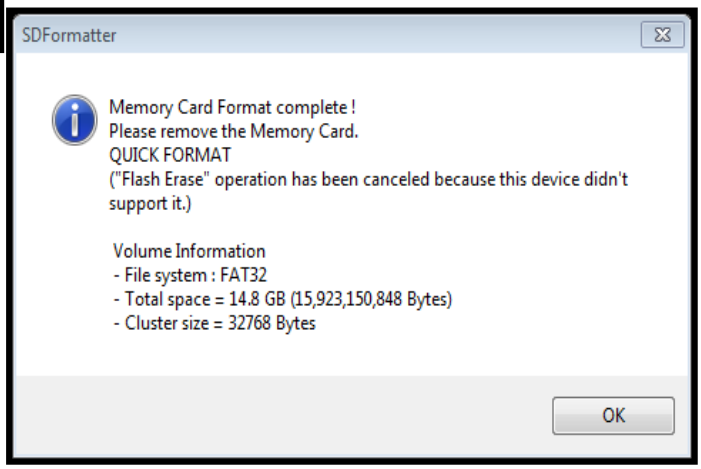
3) Choose the format type **Quick to Full (Erase)** & Click on **Ok**:



4) Formatting will erase all your data from SD Card, so make sure and then click on **Format** and click on **Ok**.



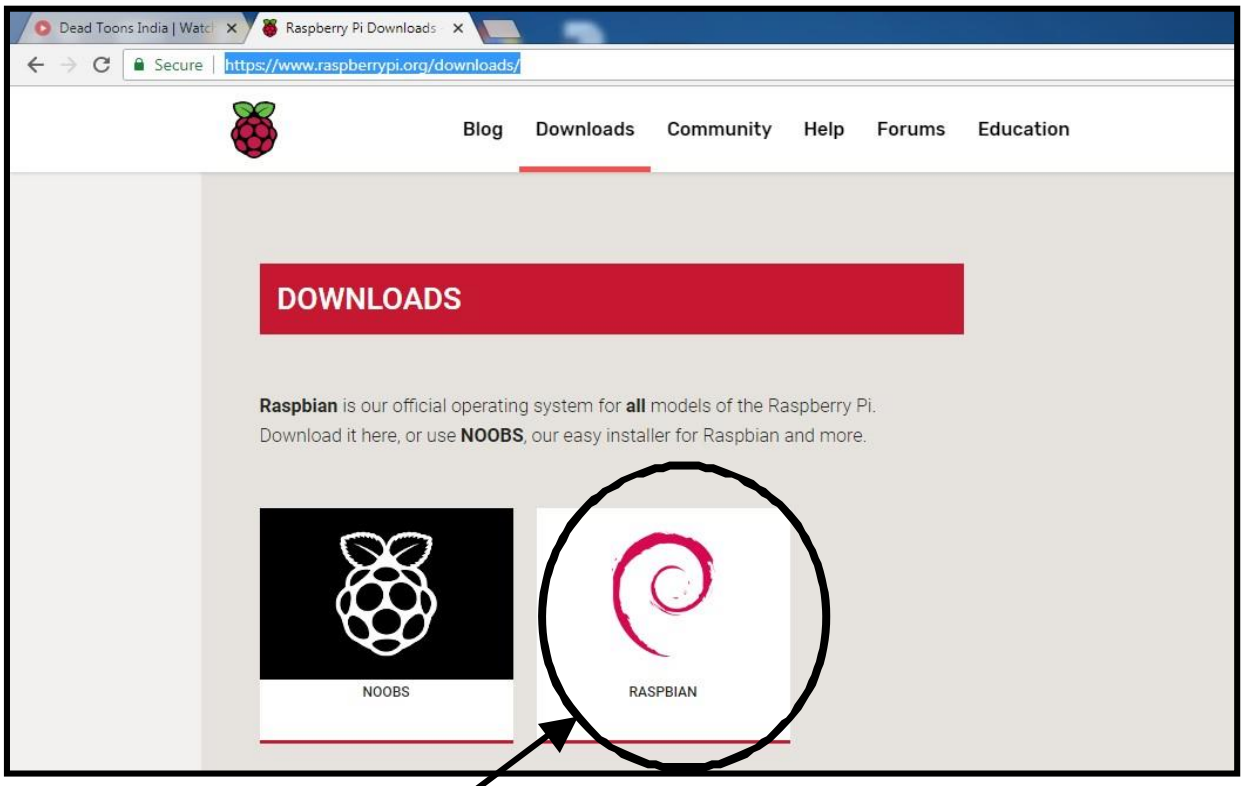
5) Now SD Card formatting is successfully completed.



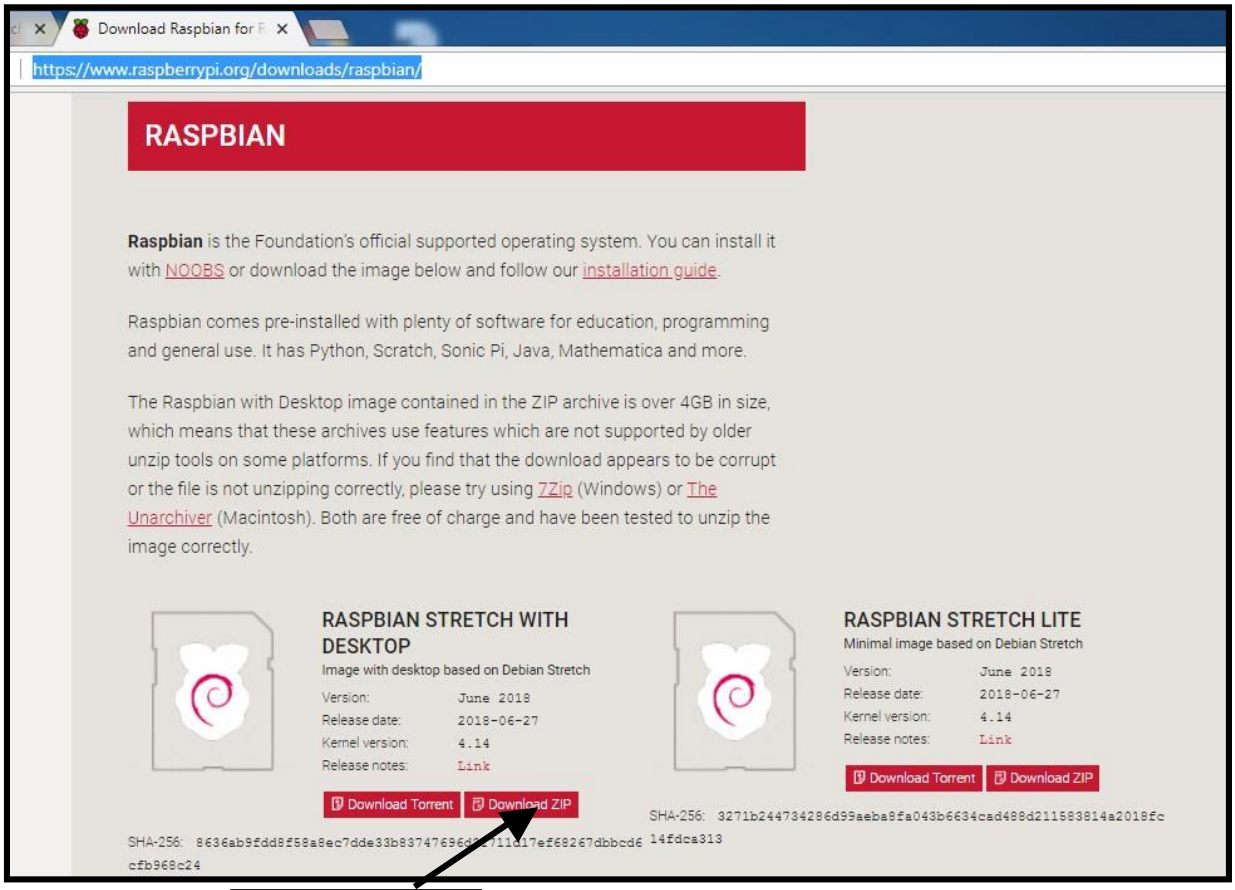
6) Download the latest version of Raspbian OS and unzip the .img file

- Download the OS from Raspberry Pi's official website using following link

https://www.raspberrypi.org/downloads/

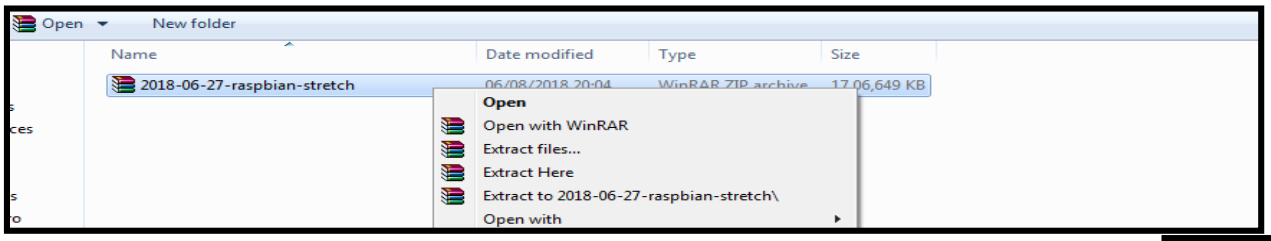


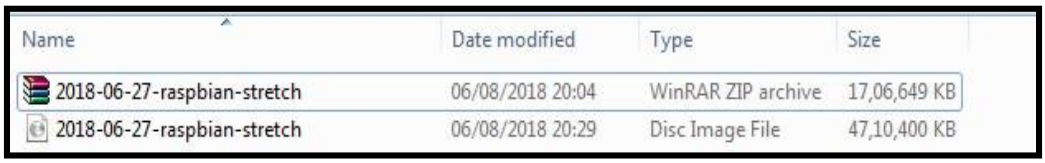
- After clicking on **Raspbian OS** option following page will be open



- Now click on Download ZIP .After the download unzip the file

7) **Extract (Raspbian) Zip** File –

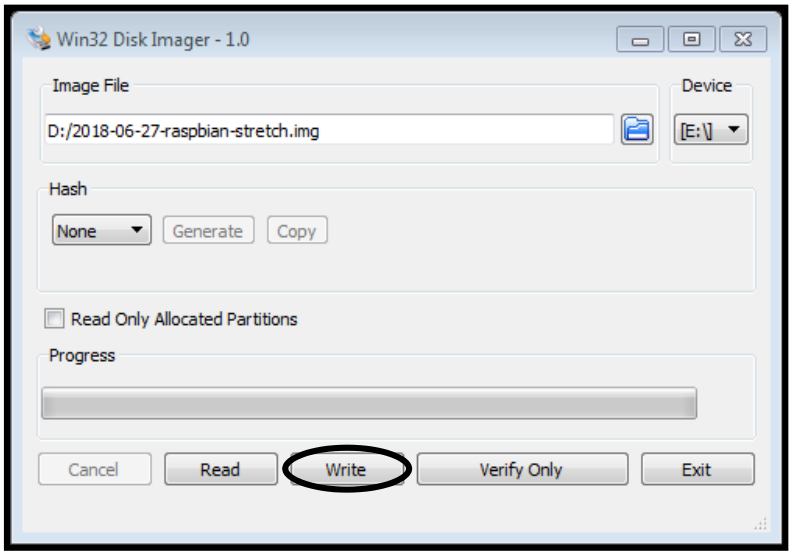




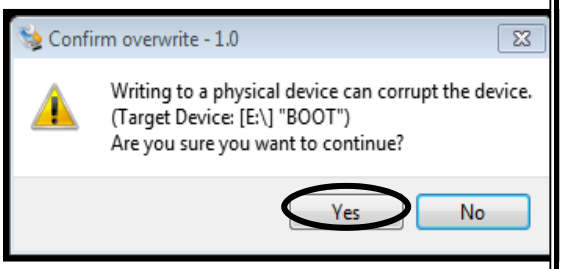
8) Make bootable image for Raspbian OS using **Win32DiskImager** software & open it.

9) Now Select Device and select path of image where **Raspbian image** file is stored using **browse** option

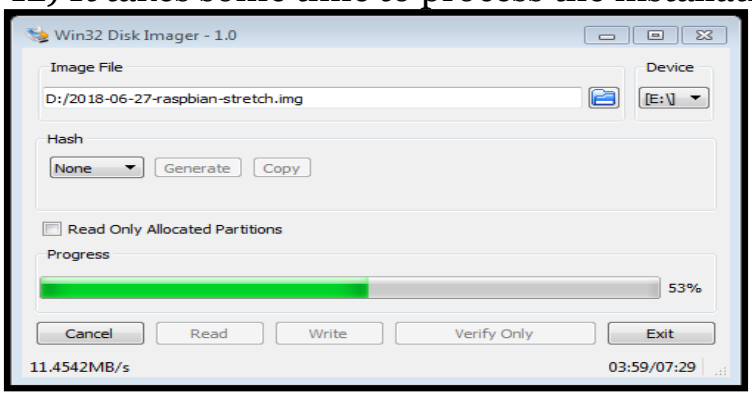
10) Now Click on the **Write** to start installation



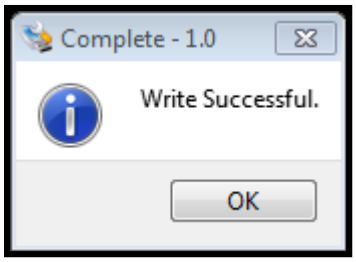
11) Now click on **Yes**.



12) It takes some time to process the installation



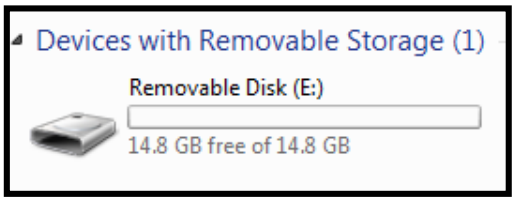
13) Installation is done



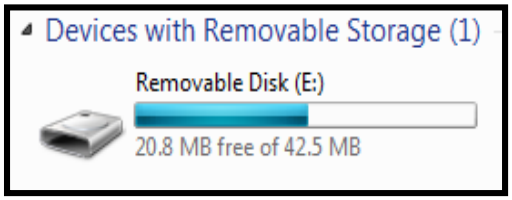
Now your SD Card is ready to use.

Open **My Computer** and check the status of your **SD Card**

Before creating bootable SD card (using Raspbian OS)



After creating bootable SD card (using Raspbian OS)

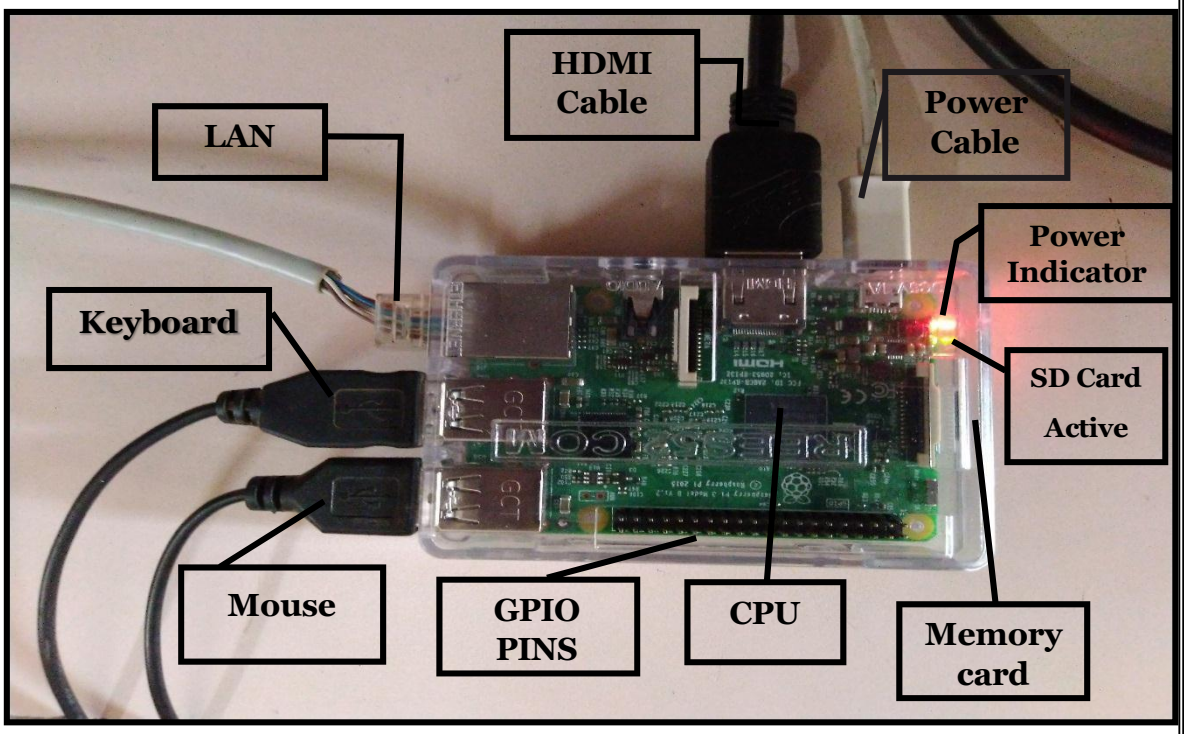


14) Right click on SD Card and Click on **Eject.**

Step 2: Time to Power Up Raspberry Pi

1) Now Remove SD card from Card Reader and insert it into Raspberry Pi.
2) Connect all devices
    a. USB Keyboard and Mouse
    b. Ethernet Cable
    c. HDMI Cable **or** HDMI to VGA Connector
    d. Power Supply

Connection of raspberry pi



Practical No:1

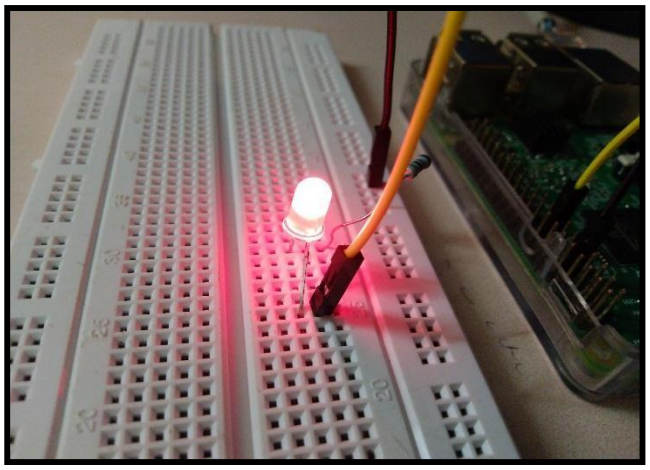
*3,2: /LJKW WKH /(' SDWWHUQ ZLWK 5DVSEHUU\ 3L

1) Using One LED

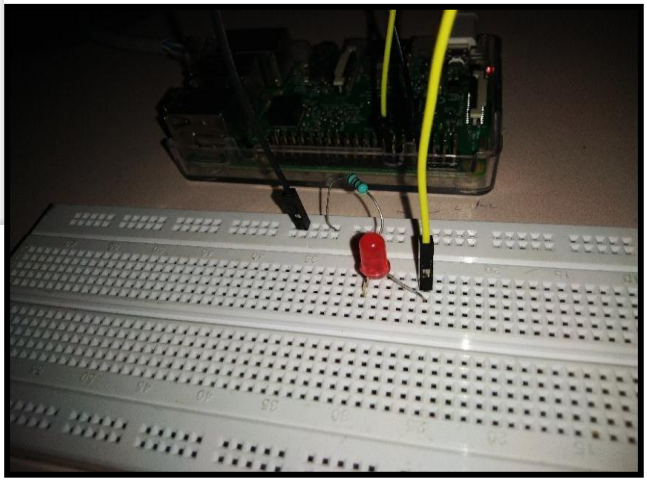
Code:

```
import RPI.GPIO as GPIO
import time

GPIO.setmode (GPIO.BOARD)
GPIO.setup(15,GPIO.OUT)
for i in range(5):
    GPIO.output(15,True)
    print("ON")
    time.sleep(1)
    GPIO.output(15,False)
    print("OFF")
    time.sleep(1)
print("Done")
GPIO.cleanup()
```

Output:



LED ON



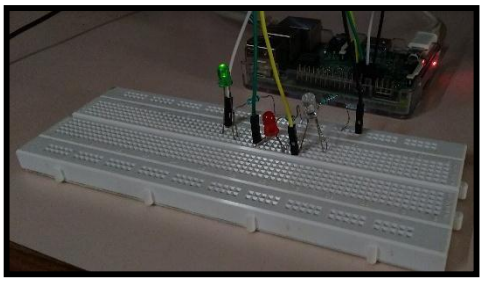
LED OFF

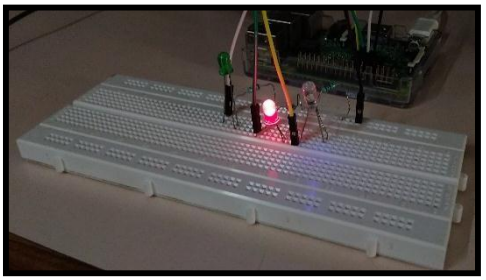
2) 3 Led blinking Code :

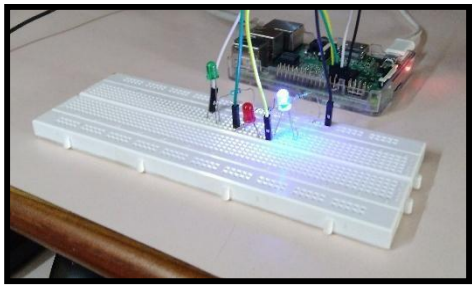
```
import RPI.GPIO as GPIO
import time

GPIO.setmode (GPIO.BOARD)
GPIO.setup(11,GPIO.OUT)
GPIO.setup(13,GPIO.OUT)
GPIO.setup(15,GPIO.OUT)

for i in range(5):
    GPIO.output(11,True)
    print("ON")
    GPIO.output(11,False)
    print("OFF")

    GPIO.output(13,True)
    print("ON")
    GPIO.output(13,False)
    print("OFF")

    GPIO.output(15,True)
    print("ON")
    GPIO.output(15,False)
    print("OFF")

print("Done")
GPIO.cleanup()
```

2) Using Three LED

Output:



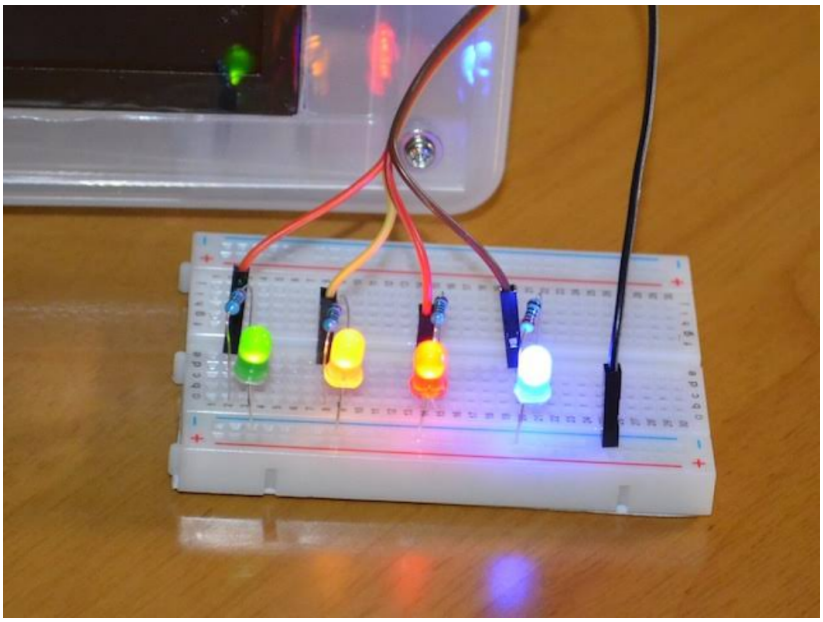




3) 4 Led Blinking Code

```
import RPI.GPIO as GPIO
import time

GPIO.setmode (GPIO.BOARD)
GPIO.setup(11,GPIO.OUT)
GPIO.setup(13,GPIO.OUT)
GPIO.setup(15,GPIO.OUT)
GPIO.setup(16,GPIO.OUT)

for i in range(5):
    GPIO.output(11,True)
    print("ON")
    GPIO.output(11,False)
    print("OFF")

    GPIO.output(13,True)
    print("ON")
    GPIO.output(13,False)
    print("OFF")

    GPIO.output(15,True)
    print("ON")
    GPIO.output(15,False)
    print("OFF")

    GPIO.output(16,True)
    print("ON")
    GPIO.output(16,False)
    print("OFF")

print("Done")
GPIO.cleanup()
```

Output :



Displaying Time over 4-Digit 7-Segment Display Using Raspberry Pi.

Installation Manual

To display small amount of data with Raspberry Pi, we can use 4 digit 7-segment Display.
7 Segment Display has seven segments in it and each segment has one LED inside it to display the numbers by lighting up the corresponding segments.

Hardware Requirements

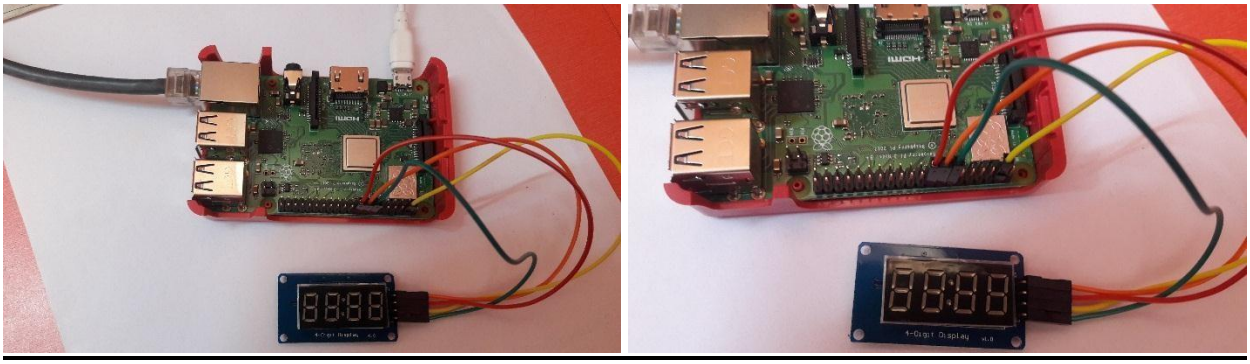
1. Raspberry Pi Model A/B/B+
2. 4 digit 7 Segment Display
3. Jumper wires (Female to Female)

Here, I am using **4 digits-7 segments LED** display with TM1637 controller

- **Software Requirements**
  1. Raspbian Stretch OS

1. **Connect your 4 digit 7 segment display with Raspberry Pi's GPIO Pins.**

TM1637 Board Pin	Function	RPI Physical Pin	Raspberry Function
GND	Ground	14	GND
VCC	+ 5V Power	4	5V
DI0	Data In	18	GPIO 24
CLK	Clock	16	GPIO 23



Step 1: Download Python Script

In order to control the LED, using a special script with pre-defined functions. Various functions are available in the script, for example, to display numbers and adjust the intensity of the LEDs.

Create a folder 4digitTime under /home/pi.

Download the script using wget command.

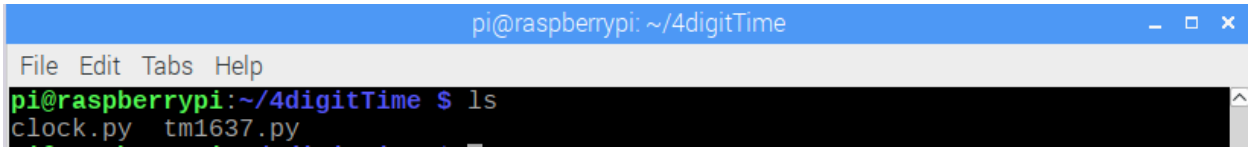
Note: This Script file contains some of the important functions, which are required to add in our Python script.

Step 2: Write Python Script to display Time (e.g clock.py)

```
import sys
import time
import datetime
import RPi.GPIO as GPIO
import tm1637

#CLK -> GPIO23 (Pin 16)
#Di0 -> GPIO24 (Pin 18)
```

```
Display = tm1637.TM1637(23,24,tm1637.BRIGHT_TYPICAL)

Display.Clear()
Display.SetBrightnes(1)
while(True):
   now = datetime.datetime.now()
   hour = now.hour
   minute = now.minute
   second = now.second
   currenttime = [ int(hour / 10), hour % 10, int(minute / 10), minute % 10 ]

   Display.Show(currenttime)
   Display.ShowDoublepoint(second % 2)

   time.sleep(1)
```

The above script needs the **tm1637.py** script to work, so place both files in the same folder.



Script functions

The clock script uses the following functions, defined in tm1637.py:

Display. Clear () - Clears the display if individual LEDs are still active.

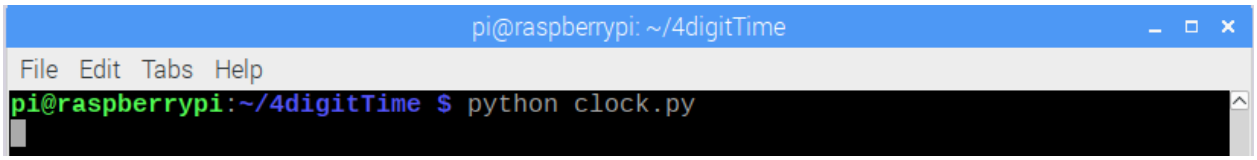
Display.SetBrightnes(x) - After this you can adjust the brightness of the display, at least 0 and maximum 7.

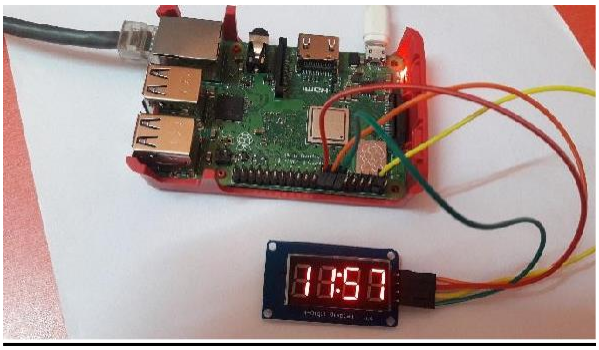
Display. Show(x,x,x,x) - Show the actual 4 digits (digits), x can be 0 to 9.

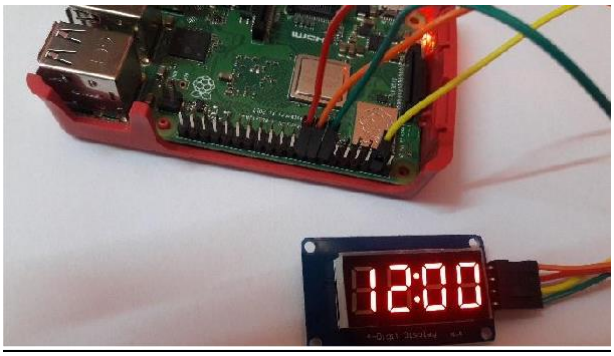
Display.ShowDoublepoint (status) - Controlling the ':' between the second and third digit, true (1) = on / false (0) = off.

To know more about TM1637 controller, check
http://www.microcontroller.it/english/Tutorials/Elettronica/componenti/TM1637.htm

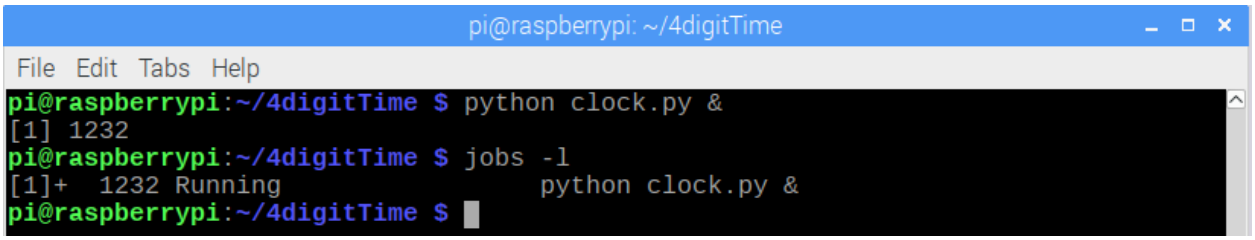
Step 3: Start the script with following command







To run the script in background you can use following command:



That's all !!!

Thank you….

Raspberry Pi based Oscilloscope

Installation Manual

One of the most important tools in Electrical/Electronic engineering is **The Oscilloscope**.

An oscilloscope is a laboratory instrument commonly used to display and analyze the waveform of electronic signals. In effect, the device draws a graph of the instantaneous signal voltage as a function of time.

In this project we will seek to replicate the signal visualization capabilities of the oscilloscope using the Raspberry Pi and an analog to digital converter module.

Replicating the signal visualization of the oscilloscope using the Raspberry Pi will require the following steps;

1. Perform Digital to analog conversion of the Input signal
2. Prepare the resulting data for representation
3. Plot the data on a live time graph

Analog Signals ⇨ **Signal Processing** ⇨ **Visualize using the Graph**

- **Hardware Requirements**

1. Raspberry Pi Model A/B/B+
2. ADS1115 ADC
3. Breadboard
4. Jumper Wires

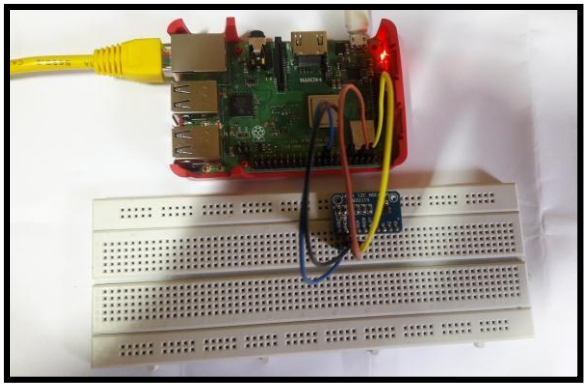
ADS1115 ADC chip is used to convert the analog input signals to digital signals which can be visualized with the Raspberry Pi. This chip is important because the Raspberry Pi does not have an on-board analog to digital converter (ADC).

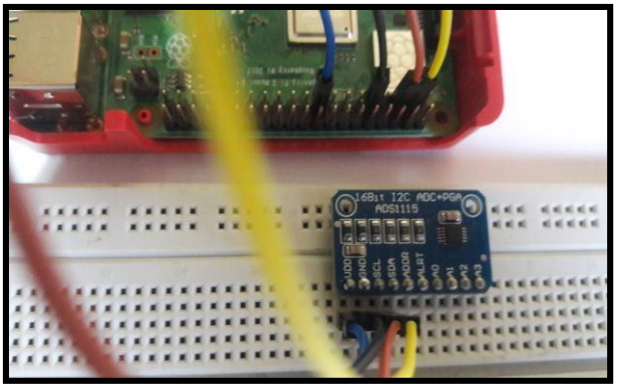
- **Software Requirements**

1. Raspbian Stretch OS
2. Adafruit module for interfacing with the ADS1115 ADC chip
3. Python Module **matplotlib** used for data visualization

1. **Connect your ADC with Raspberry Pi's GPIO Pins.**

ADS1115 ADC	Pin Number	GPIO Number
VDD	Pin 17	3.3v
GND	Pin 9	GND
SCL	Pin 5	GPIO 3
SDA	Pin 3	GPIO 2





//First visit to the link of Adafruit ADC Oscilloscope given below
https://learn.adafruit.com/adafruit-4-channel-adc-breakouts/python-circuitpython

---

//Need to install the ADC packages from Adafruit by using pip3
$ sudo pip3 install adafruit-circuitpython-ads1x15
$ sudo pip3 install adafruit-blinka

$ git clone https://github.com/adafruit/Adafruit_CircuitPython_ADS1x15

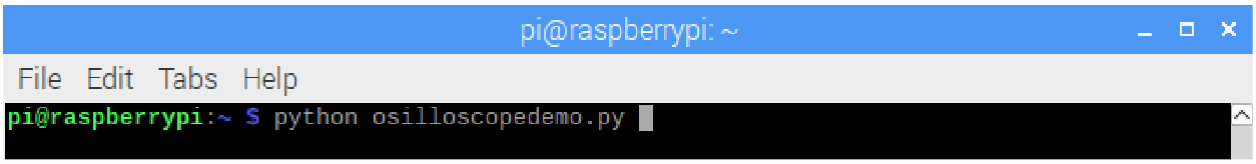
$ cd Adafruit_CircuitPython_ADS1x15
$ cd examples

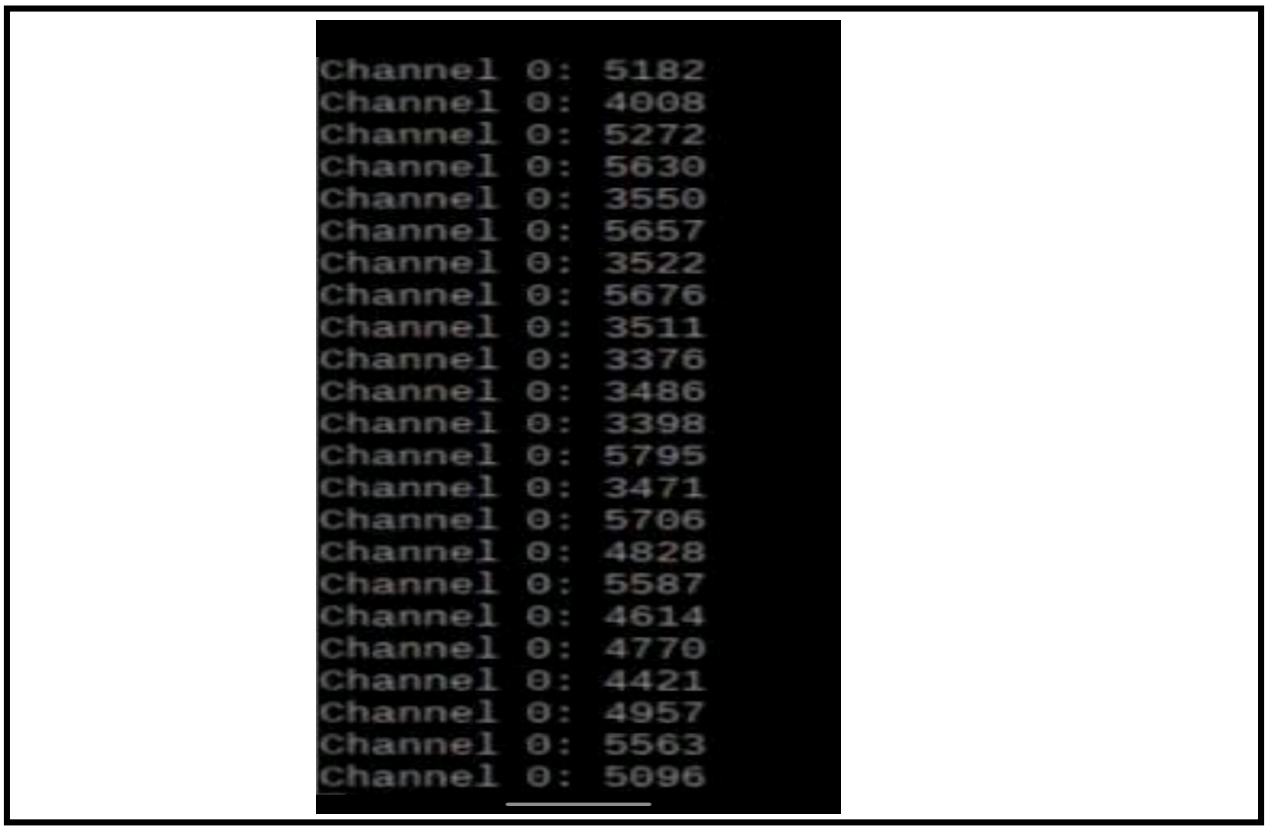
//There will be a program name ads1x15_simpletest.py and run it by using python3
$ python3 ads1x15_simpletest.py

//Or you can create the program by your own and the source code is given below
$ nano ads1x15_simpletest.py

 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
 # SPDX-License-Identifier: MIT

```
import time
import board
import busio
import adafruit_ads1x15.ads1015 as ADS
from adafruit_ads1x15.analog_in import AnalogIn

# Create the I2C bus
i2c = busio.I2C(board.SCL, board.SDA)

# Create the ADC object using the I2C bus
ads = ADS.ADS1015(i2c)

# Create single-ended input on channel 0
chan = AnalogIn(ads, ADS.P0)

# Create differential input between channel 0 and 1
# chan = AnalogIn(ads, ADS.P0, ADS.P1)

print("{:>5}\t{:>5}".format("raw", "v"))

while True:
    print("{:>5}\t{:>5.3f}".format(chan.value, chan.
voltage))
    time.sleep(0.5)
```


pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ python osilloscopedemo.py


Channel 0: 5182
Channel 0: 4008
Channel 0: 5272
Channel 0: 5630
Channel 0: 3550
Channel 0: 5657
Channel 0: 3522
Channel 0: 5676
Channel 0: 3511
Channel 0: 3376
Channel 0: 3486
Channel 0: 3398
Channel 0: 5795
Channel 0: 3471
Channel 0: 5706
Channel 0: 4828
Channel 0: 5587
Channel 0: 4614
Channel 0: 4770
Channel 0: 4421
Channel 0: 4957
Channel 0: 5563
Channel 0: 5096

Thank you….

Controlling Raspberry Pi using Telegram

Installation Manual

- **Hardware Requirements**

1. Raspberry Pi Model A/B/B+
2. LED
3. Breadboard
4. Jumper Wires

1. Connect 4 LED with Raspberry Pi's GPIO Pins.

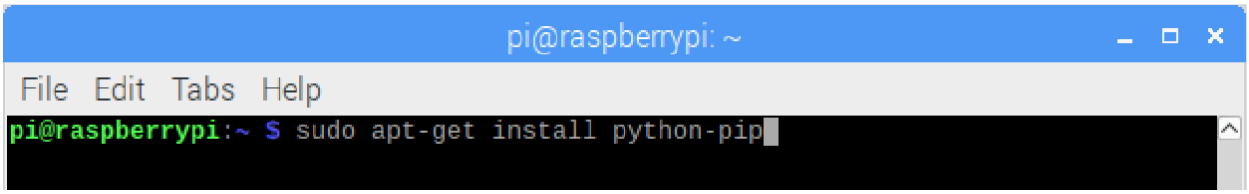
Led Terminal	Pin Number	GPIO Number
Led Positive	Pin 11	GPIO 17
Led Negative	Pin 06	Ground

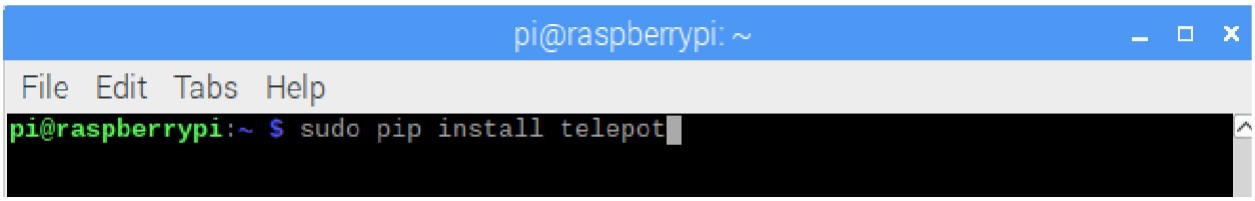
2. Install Telegram App in Mobile . Follow process to obtain access token

Raspberry Pi Telegram Bot

- Install Telegram app on your Smart Phone from Playstore.
- Open Telegram.
- Request the Bot Father to create a new Bot.
- Search "BotFather" and Click on Start
- Create new bot using /newbot
- Provide a Name for your Bot (e.g.msdtyit)
- Then ,Provide username for your Bot Must be end in "bot" (e.g. msdtyit_bot)
- After this process the BotFather will give you a Token for access.

3. **Install Telegram Bot on Raspberry Pi**





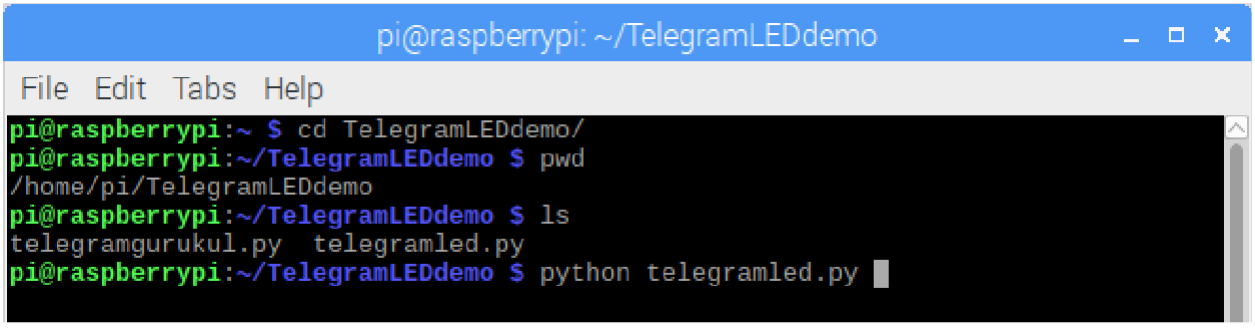
4. **Write Python Script to blink LED with Telegram Bot**

#Sample program for Light LED on/off using Telegram

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

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

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
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('5329577491:AAHhKMS0XTk-H2DMAVMwI69hoN9vWXVGKfk')
bot.message_loop(handle)
print("I am Listening....")
while 1:
    try:
        time.sleep(10)
    except keyboardInterrupt:
        print('/ program interrupt')
        GPIO.cleanup()
        exit()
    except:
        print('other error or exception occured')
        GPIO.cleanup()
```

5.



Thank you….

Fingerprint Sensor interfacing with Raspberry Pi

Installation Manual

Finger Print Sensor, which we used to verify the identity of a person for various purposes.

Nowadays we can see fingerprint-based systems everywhere in our daily life.

One of the advantage of fingerprint-based systems is that passwords and / or number codes can be completely omitted.

Hardware Requirements

1. Raspberry Pi Model A/B/B+
2. Fingerprint Module
3. Serial USB Converter
4. Jumper Wires

Here, I am using **R307 fingerprint module** with **Serial USB Converter.**

R307 Fingerprint Module consists of optical fingerprint sensor, Supply voltage: DC 4.2 ~ 6.0V

Pin No	Pin Name	Details
1	5V	Regulated 5V DC
2	GND	Common Ground
3	TXD	Data output - Connect to MCU RX
4	RXD	Data Input - Connect to MCU TX
5	TOUCH	Active Low output when there is touch on sensor by finger
6	3.3V	Use this wire to give 3.3V to sensor instead of 5V

Note: So I connected first 4 pins of Fingerprint module to Serial USB Converter.

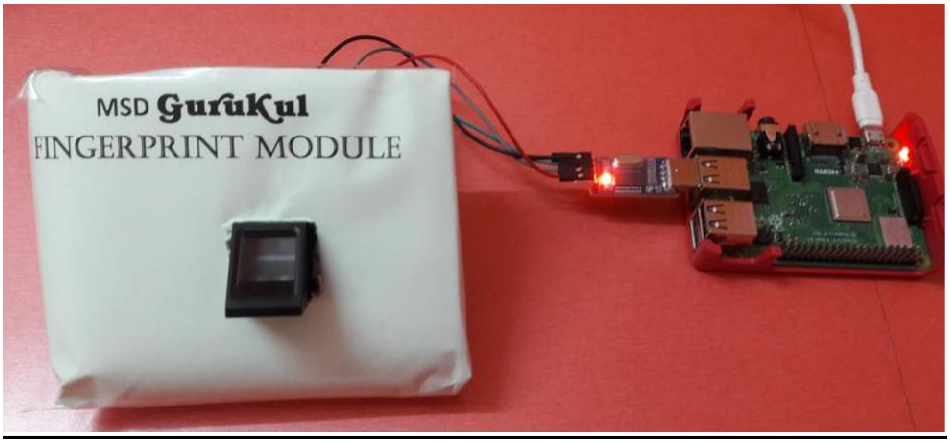
Connect fingerprint module to USB Serial converter.

Fingerprint Module	USB Serial Converter
5V	5.0 V
GND	GND
TXD	TXD
RXD	RXD
TOUCH	-
3.3V	-

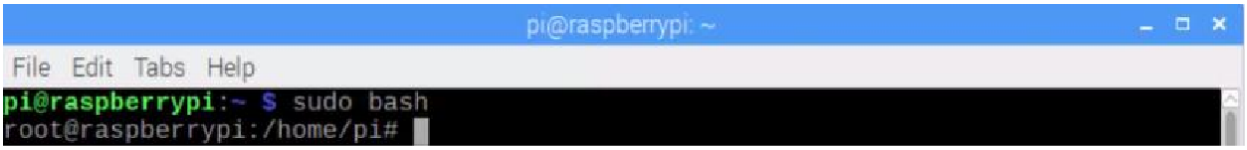
Note: Check USB to Serial Converter pins according to your model. And connect accordingly.

- **Software Requirements**
  1. Raspbian Stretch OS

1. **Now, just connect fingerprint module to Raspberry Pi USB port by using USB to Serial converter.**



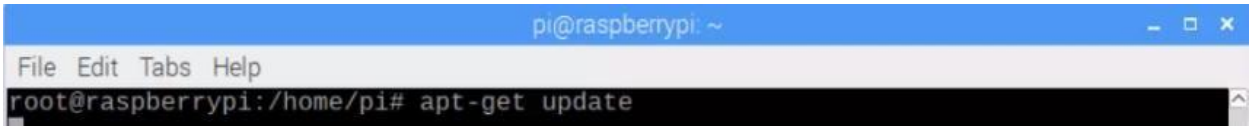
Step 1: To install this library, root privileges are required. So login with root user.



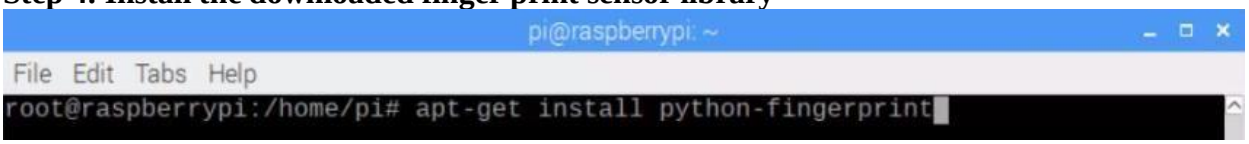
Step 2: Download some required packages using wget command

```
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/
```

Step 3: Update the Raspberry Pi

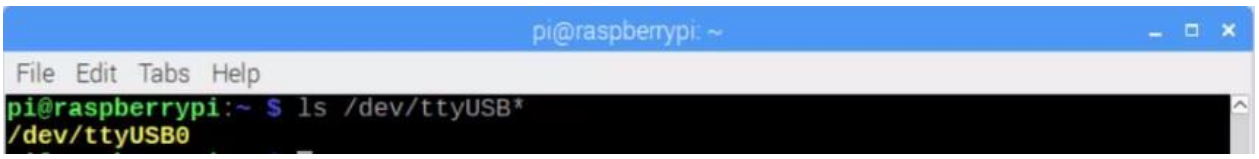


Step 4: Install the downloaded finger print sensor library

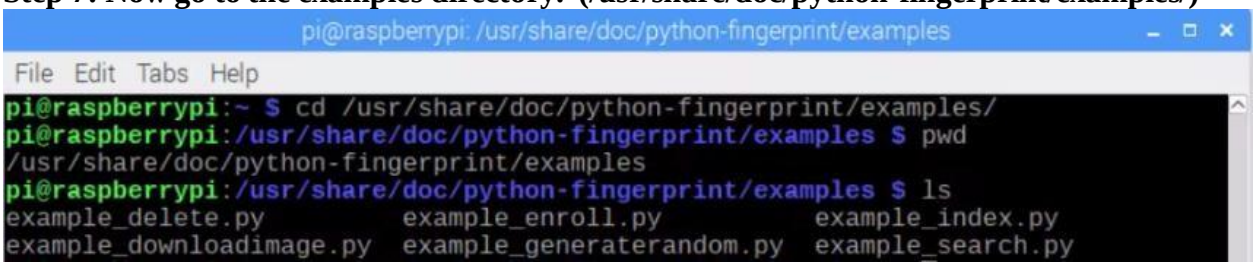


Step 5: To return to the normal shell (under the Pi user**), type** exit

Step 6: Now check USB port **on which your** finger print sensor **is connected.** Use this **USB port** in our **Python script.**



Step 7: Now go to the examples directory. (/usr/share/doc/python-fingerprint/examples/)



Step 8: Run sample file, to test to see if the sensor is detected and ready for access

```
pi@raspberrypi:/usr/share/doc/python-fingerprint/examples $ sudo python example_
index.py
Currently used templates: 4/1000
Please enter the index page (0, 1, 2, 3) you want to see: 1
```

The above data should appear, which allows you to display the positions under which an imprint is stored by selecting a page (0-3).
If here you get **Exception message**, then something is wrong with the cabling or the sensor. Check it again.

Step 9: Now execute other scripts, to make sure Fingerprint module is working.

Script	Usage
example_index.py	**Shows template index table.**
example_enroll.py	**Stores new fingerprint**
example_delete.py	**Deletes a fingerprint from sensor**
example_search.py	**Search for recorded fingerprint**
example_downloadimage.py	**Read fingerprint and download it.**
example_generaterandom.py	**Generates 32 bit random number.**

Run **example_enroll.py** script to store new fingerprint

```
pi@raspberrypi:/usr/share/doc/python-fingerprint/examples $ sudo python example_
enroll.py
```

Put your finger on the glass surface, wait for the instruction in the terminal and remove your finger as soon as it is written there. Afterwards you have to put your finger a second time for the verification and the imprint is stored in the next number.

Now, Run **example_search.py** script to see whether our finger is recognized.

```
pi@raspberrypi:/usr/share/doc/python-fingerprint/examples $ sudo python example_
search.py
```

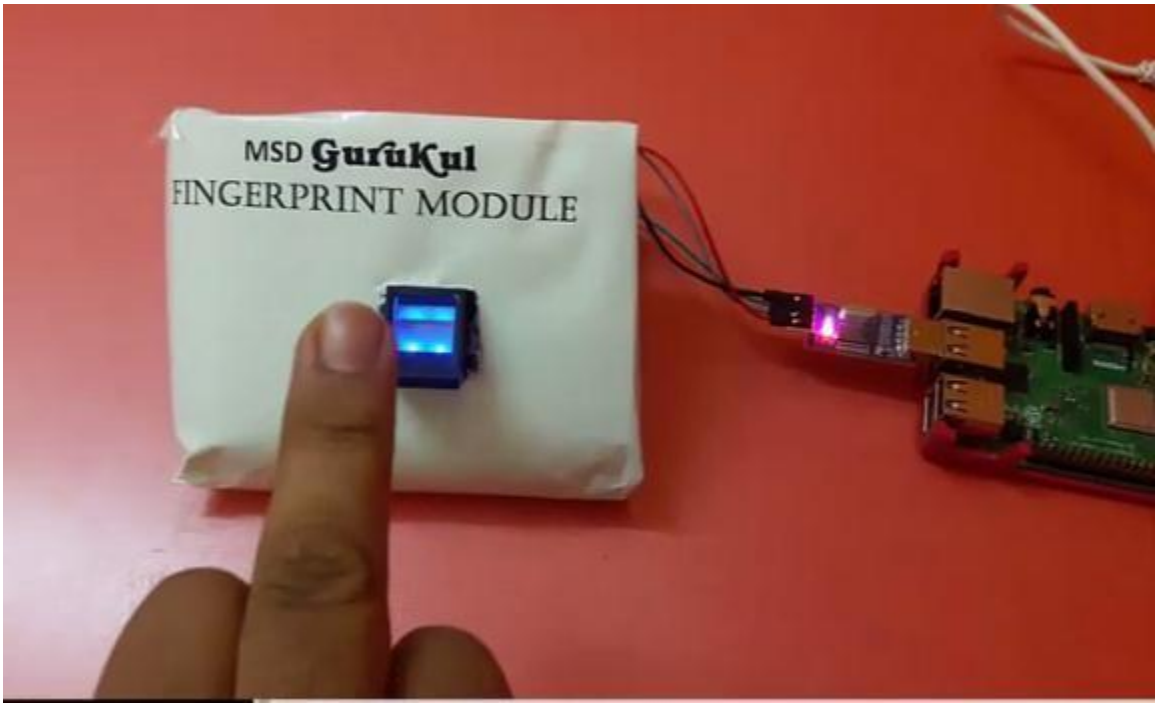
Put the **same finger** on glass surface.
If the fingerprint is detected, it displays below message.

```
pi@raspberrypi:/usr/share/doc/python-fingerprint/examples $ sudo python example_
search.py
Currently used templates: 10/1000
Waiting for finger...
Found template at position #9
The accuracy score is: 109
SHA-2 hash of template: 37e93008eaa36b21907a58839ad01aa5b7103057ebbc471590d21c38
d0119c09
```

Execute same script again, this time use other finger which is not previously stored.
If fingerprint is not detected, then gives **"No match Found"** message

```
pi@raspberrypi:/usr/share/doc/python-fingerprint/examples $ sudo python example_
search.py
Currently used templates: 10/1000
Waiting for finger...
No match found!
```

So, our fingerprint module is working properly.



That's all!!!

Thank you….

Interfacing GPS module with Raspberry Pi

Installation Manual

- **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).
- GPS module is the main component in our vehicle tracking system.
- 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.
- NMEA format consists of sentence starts from **$GPGGA,** the coordinates, time and other useful information.
- **GPGGA** is referred to **Global Positioning System Fix Data**.
- GPGGA string contains following co-ordinates separated by commas.

Sr No	Identifier	Description	Sr No	Identifier	Description
1	**$GPGGA**	**Global Positioning system fix data**	**7**	**FQ**	**Fix Quality Data**
2	**HHMMSS.SSS**	**Time in hour minute seconds and milliseconds format.**	**8**	**NOS**	**No. of Satellites being Used**
3	**Latitude**	**Latitude (Coordinate)**	**9**	**HPD**	**Horizontal Dilution of Precision**
4	**N**	**Direction N=North, S=South**	**10**	**Altitude**	**Altitude from sea level**
5	**Longitude**	**Longitude(Coordinate)**	**11**	**M**	**Meter**
6	**E**	**Direction E= East, W=West**	**12**	**Height**	**Height**
			13	**Checksum**	**Checksum Data**

Hardware Requirements	Software Requirements
• Raspberry Pi Model B/B+ , SD Card • Ethernet Cable / Wi-Fi • Power Supply to Pi • Neo 6m v2 GPS Module • Jumper wires • Breadboard • 2x16 LCD Display • Potentiometer (to adjust Contrast)	• Raspbian Stretch OS • Adafruit Python Char LCD Library

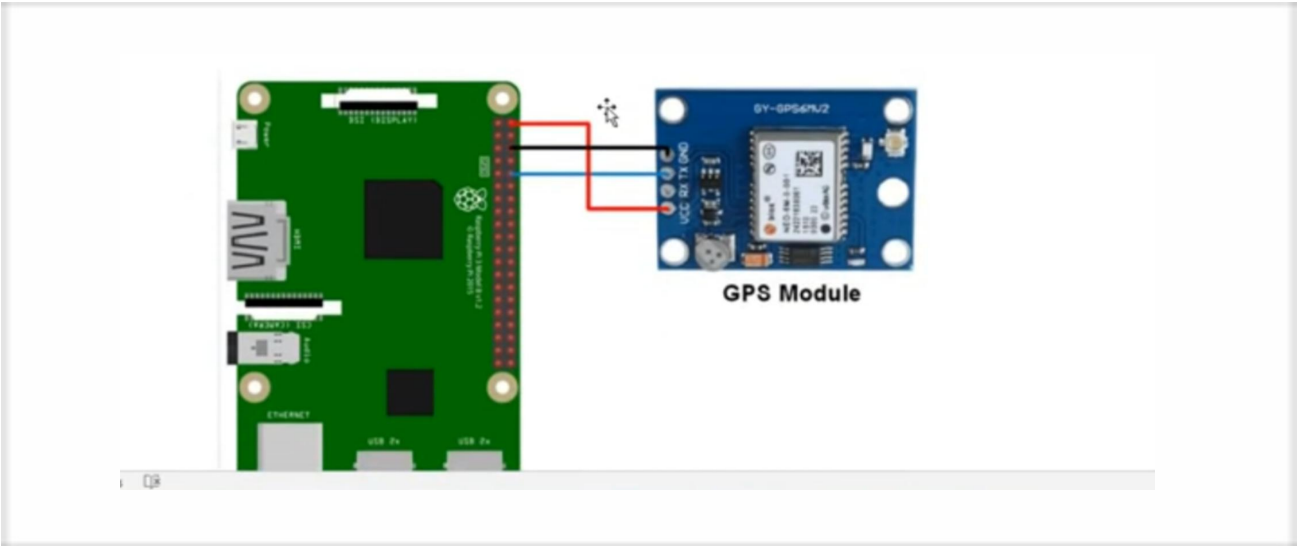
Neo 6m v2 GPS Module

- This board features the u-blox NEO-6M GPS module with antenna and built-in EEPROM. This is compatible with various flight controller boards designed to work with a GPS module.
- EEPROM is used for saving the configuration data when powered off.
- Power Supply Range: 3 V to 5 V
- Default Baud Rate: 9600 bps

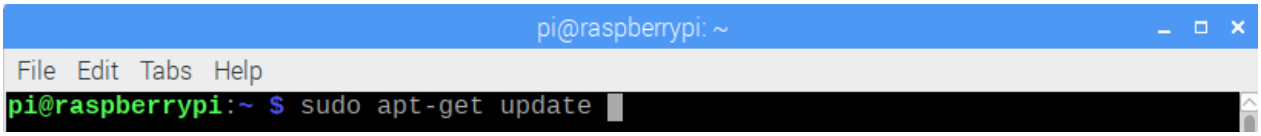
Connect GPS Module to RPi

Connection is very simple. Requires 4 Female to Female Jumper Wires

Neo 6m V2 GPS Board Pin	Details	Raspberry Pi Physical Pin	Raspberry Pi Function
VCC	Power	Pin 2	3.3V Power
GND	Common Ground	Pin 6	GND
TXD	Data Output	Pin 10	(UART_RXD0) GPIO15
RXD	Data Input	Not required	(UART_TXD0) GPIO14



Step 1: Update Raspberry Pi

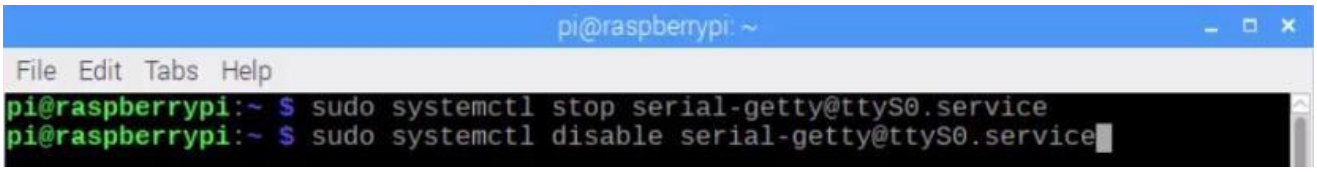


Step 2: edit the /boot/config.txt file


```
#config starts- do not modify other configuration lines
dtparam=spi=on
dtoverlay=pi3-disable-bt
core freq=250
enable_uart=1
force_turbo-1
#config ends To save and
#exit editor press Ctrl+O> Enter -> ctrl +X
```

Step 3 : Reboot Raspberry Pi using the command *sudo reboot*

Step 4 : Stop and disable the Pi's serial ttyS0 service

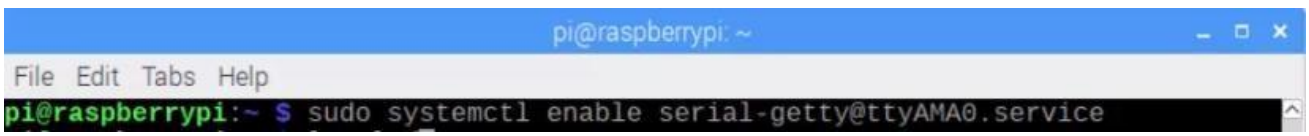


The following commands can be used to enable it again if needed

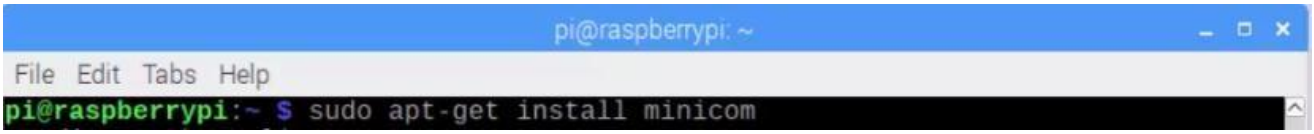
```
sudo systemctl enable serial-getty@ttyS0.service

sudo systemctl start serial-getty@ttyS0.service
```

Step 5 : Reboot Raspberry Pi using the command *sudo reboot*

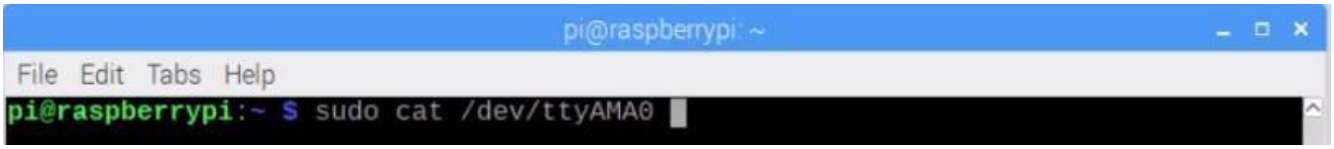
Step 6 : Now, Enable the ttyAMA0 service

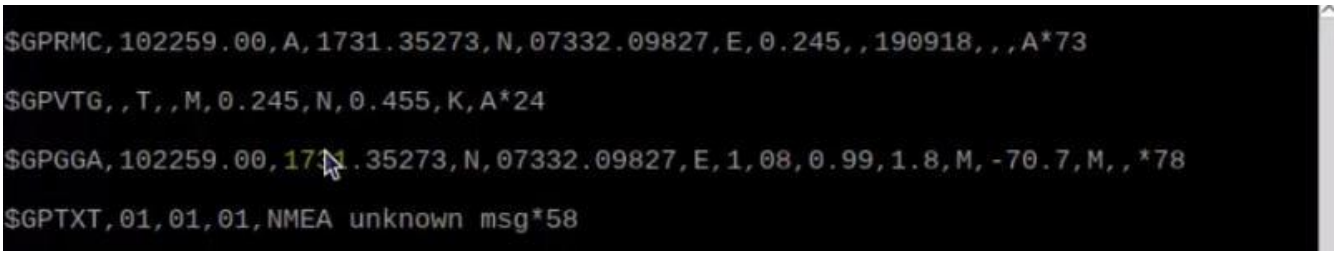


Step 7:Install minicom

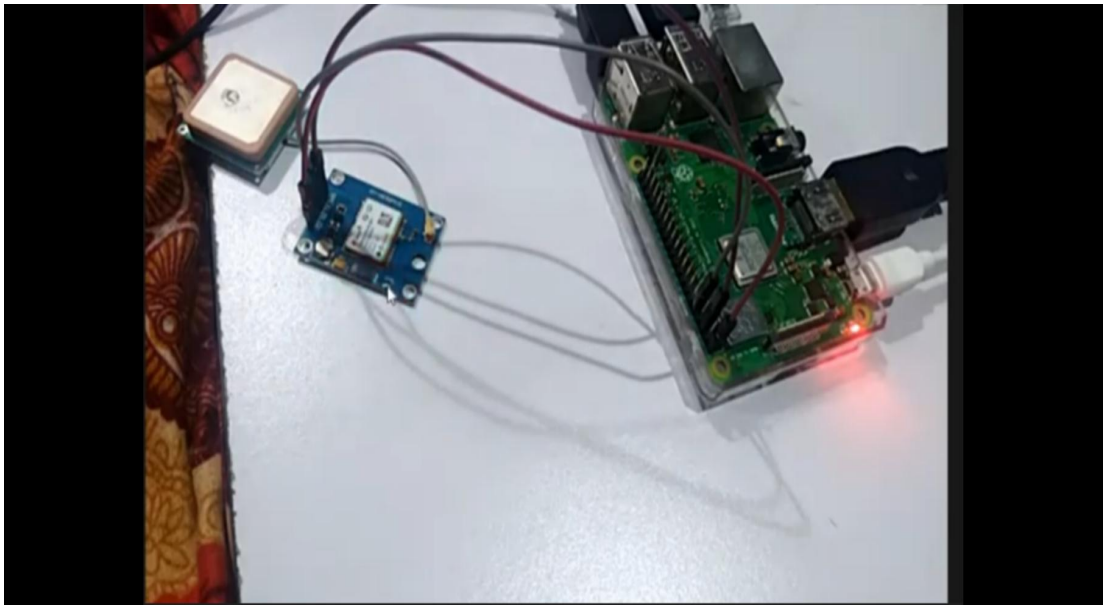


Now Run file by below command





This sentence gives you Latitude found after two commas and Longitude found after four commas.



That's all!!! Thankyou….

Interface Raspberry Pi with Pi Camera

Installation Manual

In this tutorial, we are demonstrating Visitors Monitoring System with Image capture and recording functionality. This Monitoring system will digitize and automate the whole visitor entries, and there will be no need to maintain them manually.

We are interfacing Pi camera with Raspberry Pi to capture the image and record videos of every visitor which has entered through the Gate or door. This captured image is saved in the system with the Date and time of the entry.

This system is very useful in offices, factories where visitor entry record is maintained for visitors, employees. So this can be very useful for security purpose.

Hardware Requirements

- Raspberry Pi , SD Card
- Pi camera
- Buzzer
- 2 Push Buttons
- LED
- Bread Board
- Jumper Wires
- Power supply
- Ethernet Cable / Wi Fi
- Monitor , Keyboard, Mouse (**Optional** ,For without headless connection)

Software Requirements

1. Raspbian Stretch OS
2. Python Script

System Explanation

- **Pi camera** is used to capture the images and record the videos of visitors, when a **push button** is pressed or triggered.
- After **pushing the one button**, Raspberry Pi sends command to Pi Camera to **click the picture** and save it.
- Similarly, after **pushing another button**, Raspberry Pi sends command to Pi Camera to **record the video** and save it.
- The **buzzer** is used to generate sound when button pressed.
- **LED** is used for indicating that Raspberry Pi is ready to accept Push Button press, means when LED is ON, and system is ready for operation.

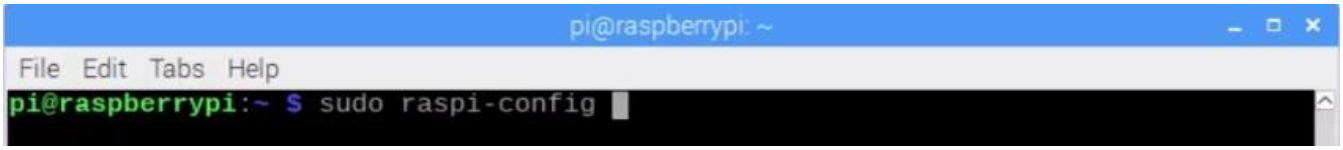
Circuit Explanation

❖ Connect Camera to Raspberry Pi

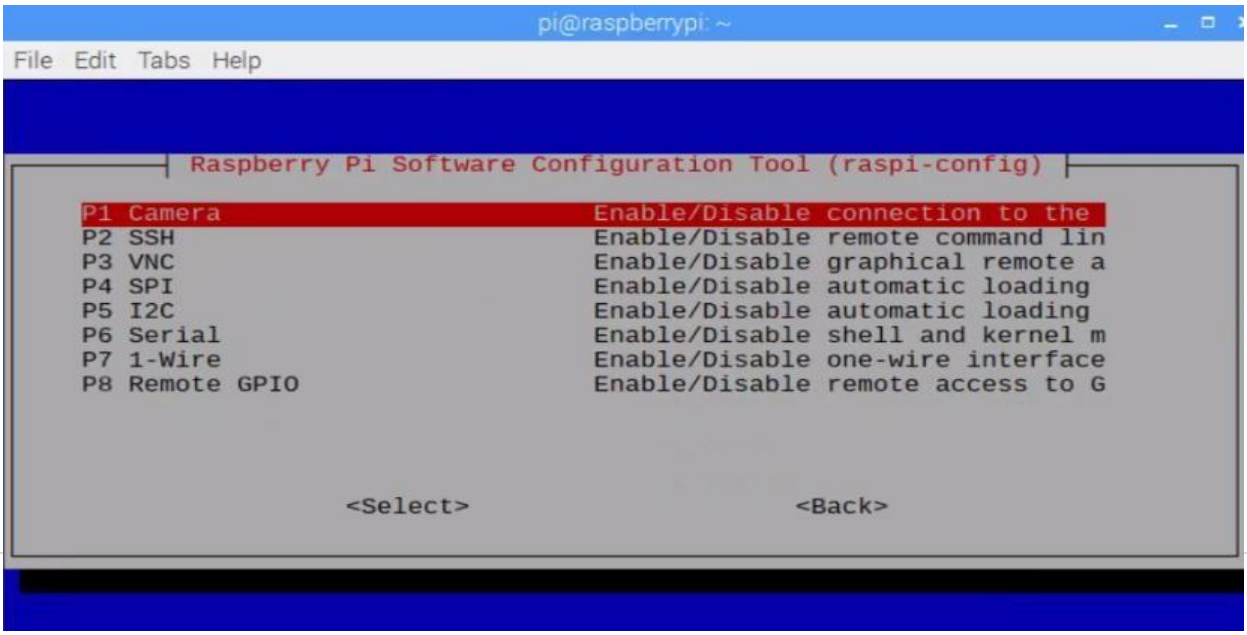
The Camera Module is a great accessory for the Raspberry Pi, allowing users to take still pictures and record video in full HD. Connect the **Camera Module** to the Raspberry Pi's camera port (Blue side of cable to face Ethernet connection).

Note that the camera preview only works when a monitor is connected to the Pi, so remote access (such as SSH and VNC) will not allow you to see the camera preview.

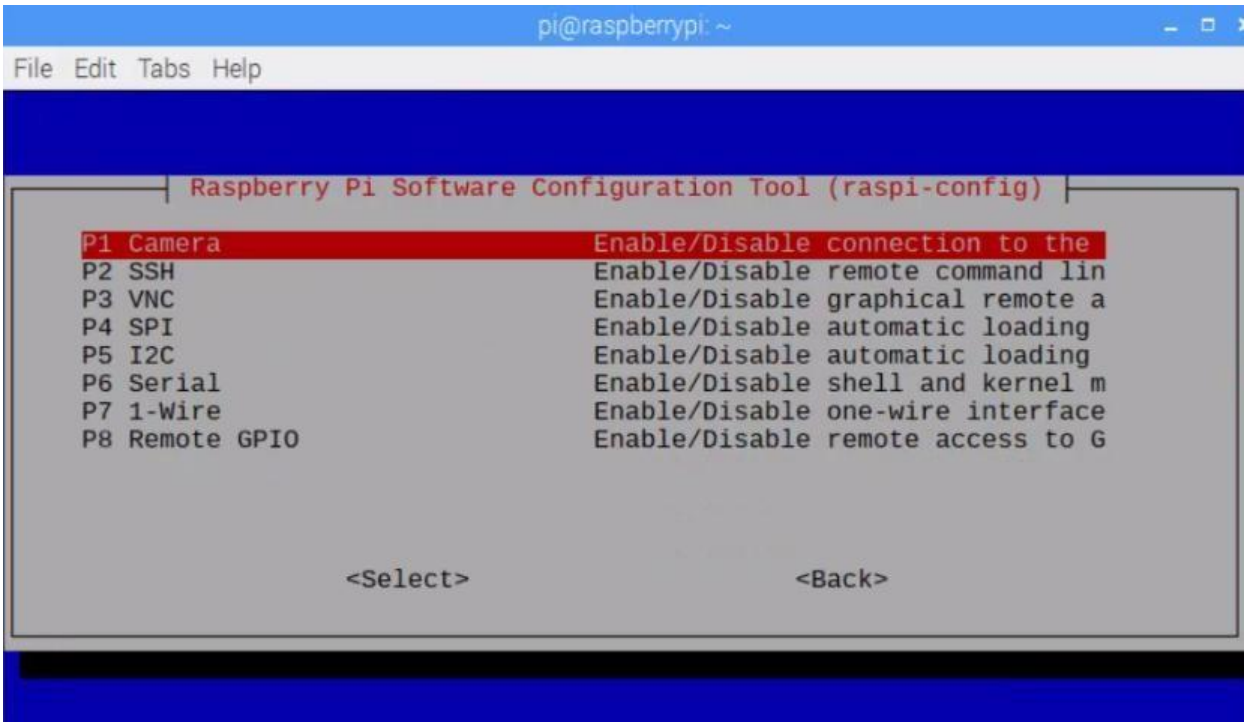
Enable Raspberry Pi Camera by using Raspberry Pi Software Configuration Tool (raspi-config)



Select Interfacing Options → Camera → Enable



Select Interfacing Options → Camera → Enable



After this reboot the system, using **sudo reboot** command.

Step 4: After Reboot, Create a folder which is used to store pictures captured by Pi Camera. (E.g. /home/pi/Desktop/Visitors)

Step 5: Write a Python Script to capture images and save them in folder. (capture_record_picamera.py)

Install Pi-camera packages. Run following command on terminal.

sudo apt-get install python-picamera

sudo apt-get install python3-picamera

sudo pip install picamera

Enable camera

Note: Use IDLE to write and execute this program

```
#program to capture image starts
import time
import picamera
camera =picamera.PiCamera()
camera.resolution =(1024,768)
camera.start_preview()
time.sleep(2)
camera.capture('test.jpg')
camera.stop_preview()
#program to capture image ends
```

That's all!!!

Thankyou…

Interfacing Raspberry Pi with RFID.

Installation Manual

RFID stands for **Radio Frequency Identification** uses radio frequency to read information stored in a RFID card or tag. Each card has a unique ID and this makes it a perfect choice for many authentication applications. The RFID authentication systems are easy to design and are cheap in cost. Interfacing RFID Reader with Raspberry Pi can be very useful as you can implement a wide range of applications like:

- Access Control
- Authentication
- e-Ticket ,e-Payment ,e-Toll
- Attendance System

Hardware Requirements

1. Raspberry Pi Model 3 B/B+
2. RFID Reader (RC 522)
3. RFID Tags or Cards
4. Jumper wires (Female to Male)
5. Breadboard

Here, I am using **RFID Reader RC 522**

This module came with two different styles of header pins, one of which needed to be soldered onto the PCB.

The RFID RC522 is a very low-cost RFID (Radio-frequency identification) reader and writer that is based on the MFRC522 microcontroller. This microcontroller provides its data through the SPI protocol and works by creating a 13.56MHz electromagnetic field that it uses to communicate with the RFID tags.

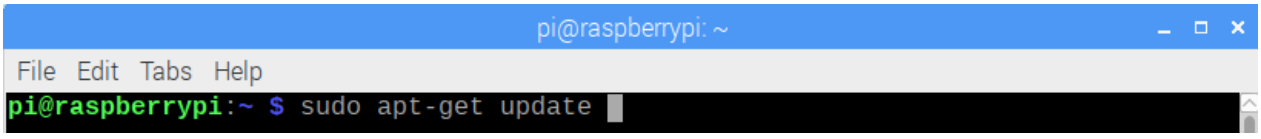
- **Software Requirements**

1. Raspbian Stretch OS
2. SPI Supporting Libraries
3. RC522 Python Library

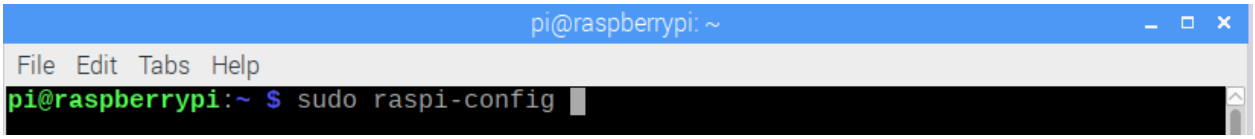
Connect your RFID reader with Raspberry Pi's GPIO Pins.

RFID Reader Board Pin	RPI Physical Pin	Raspberry Function
SDA	**24**	**GPIO8 (SPI_CE0_N)**
SCK	**23**	**GPIO11 (SPI0-_CLK)**
MOSI	**19**	**GPIO10 (SPI0_MOSI)**
MISO	**21**	**GPIO9 (SPI0_MISO)**
IRQ	**UNUSED**	
GND	**6**	**GND**
RST	**22**	**GPIO25 (GPIO_GEN6)**
3.3V	**1**	**3.3V PWR**

Step 1: Update Raspberry Pi



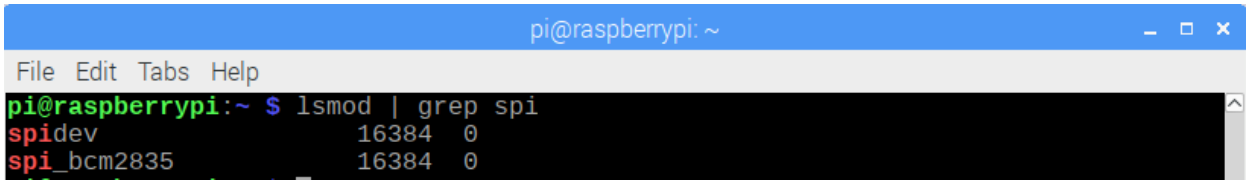
Step 2: Enable SPI Interface using sudo raspi-config tool



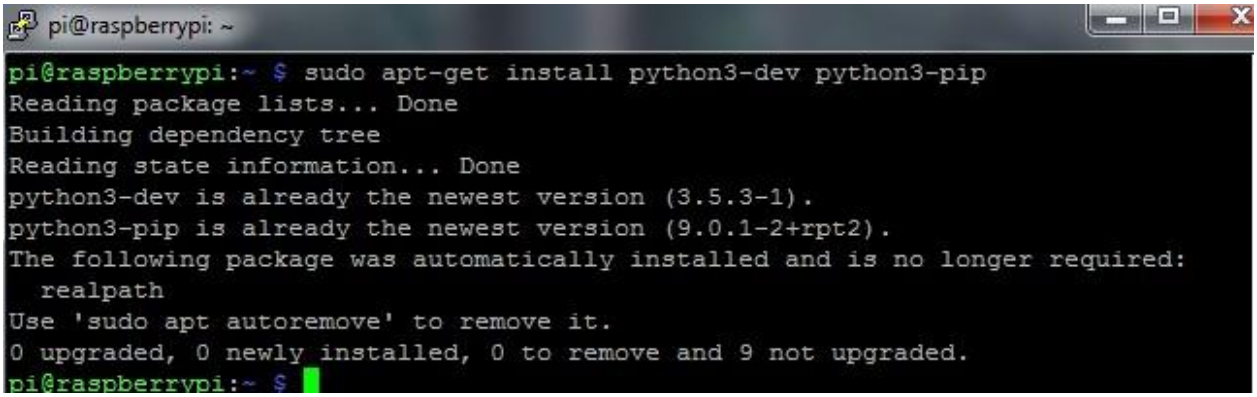
Select using Arrow Keys → Interfacing Options → SPI → Enable

Step 3: Reboot Raspberry Pi

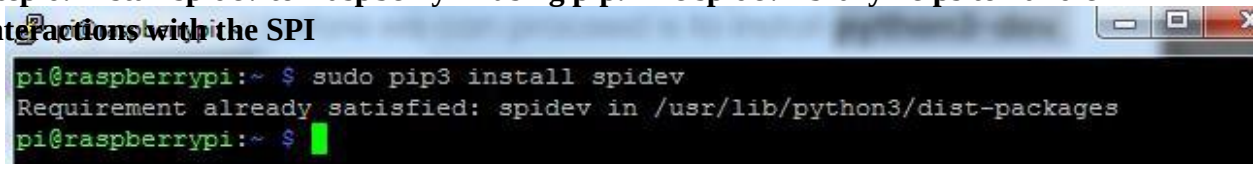
Step 4: Check to make sure that SPI has been enabled.



Step 5: Install python3-dev, python3-pip packages to setting up RFID reader



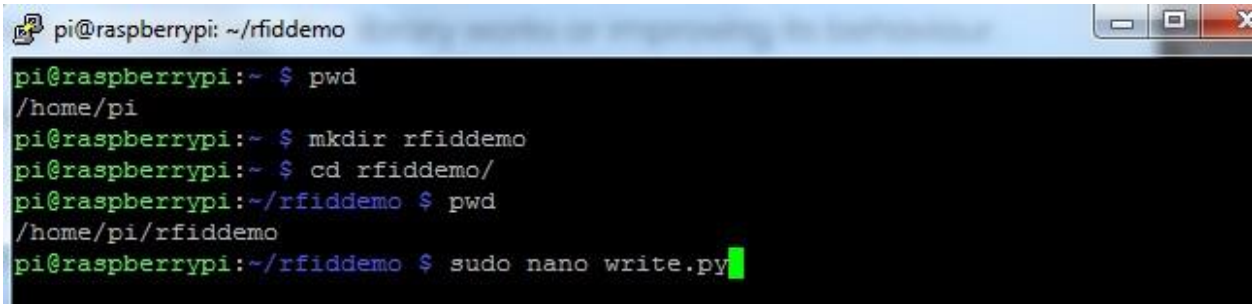
Step 6: Install spidev to Raspberry Pi using pip. The spidev library helps to handle interactions with the SPI



Step 7: Install the MFRC522 library using pip that helps talk to the RC522 module over the SPI interface

There are two files included in this repository:
MFRC522.py **which is an implementation of the RFID RC522 circuit.**
SimpleMFRC522.py **that takes the** MFRC522.py **file and greatly simplifies it.**

Step 8: Write Python script which is used to write data from the RC522 to your RFID tags.

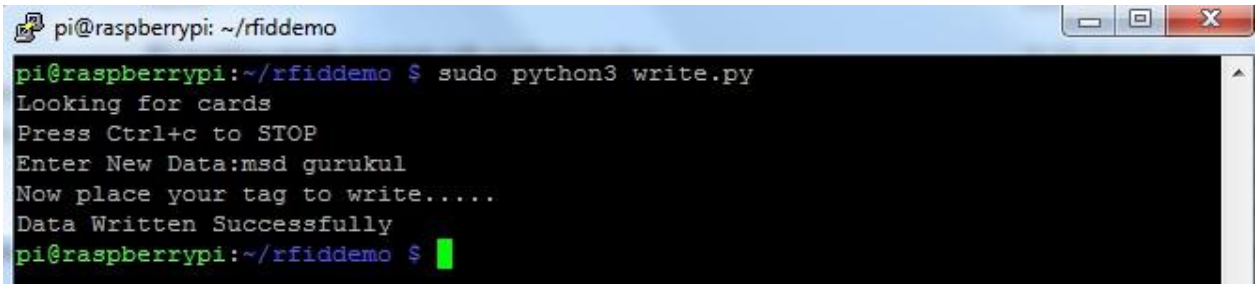



```
import RP1.GPIO as GPIO
from mfrc522 import SimpleMFRC522

reader = SimpleMFRC522()

#welcome message
print("Looking for cards")
print("Press Ctrl+c to STOP")

try:
    text=input('Enter New Data:')
    print("Now place your tag to write....")
    reader.write(text)
    print("Data Written Successfully...")
finally:
    GPIO.cleanup()
```

#Write this code in write.py

And Run this file

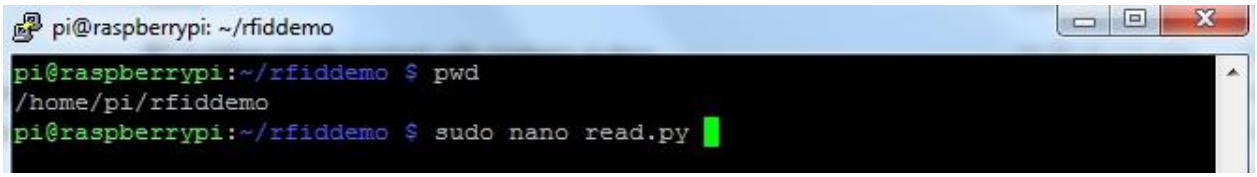
Run above script



When you run script, it asked to write in the new data, in my case I am going to just type in msd gurukul. **Press** Enter **when you are happy with what you have written.**

With that done, simply place your RFID Tag on top of your RFID RC522 circuit. As soon as it detects it, it will immediately write the new data to the tag. You should see "Data Written successfully**" appear in your command line if it was successful.**

Step 8: Write Python script which is used to read this data back off the RFID tag.

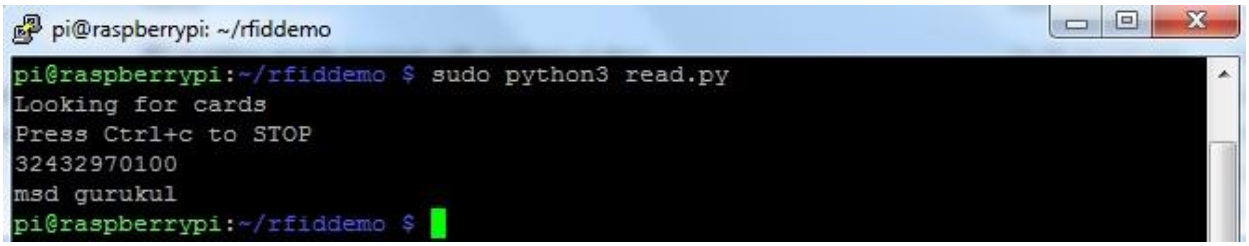



```
import RPi.GPIO as GPIO
from mfrc522 import
SimpleMFRC522

reader = SimpleMFRC522()

#welcome message
print ("Looking for cards")
print ("Press Ctrl+c to STOP")

try:
    id, text = reader.read()
    print(id)
    print(text)
finally:
    GPIO.cleanup ()
```

#Write this code in read.py

And Run this file

Run above script



With the script now running, all you need to do is place your RFID Tag on top of your RFID RC522 circuit. As soon as the Python script detects the RFID tag being placed on top, it will immediately read the data and print it back out to you.

If you successfully receive data back from your **read.py** script with the text that you pushed to the card using your **write.py** script then you have successfully set up your Raspberry Pi to connect with your RFID RC522 Circuit.

That's all !!!

Thank you…