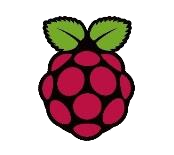
INTERNET OF THINGS

**Index**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Practical No** | **Topic** | **Page No** | **Date** | **Sign** |
| 1. | Starting Raspbian OS, Familiarizing with Raspberry Pi Components and interface, Connecting to ethernet, Monitor, USB. |  |  |  |
| 2. | Displaying different LED patterns with Raspberry Pi. |  |  |  |
| 3. | Displaying Time over 4-Digit 7-Segment Display using Raspberry Pi |  |  |  |
| 4. | GPS Module interfacing with Raspberry Pi |  |  |  |
| 5. | Interfacing Raspberry Pi with RFID. |  |  |  |
| 6. | Visitor Monitoring with Raspberry Pi and Pi Camera |  |  |  |
| 7. | Fingerprint Sensor interfacing with  Raspberry Pi |  |  |  |
| 8. | Controlling Raspberry Pi with Telegram. |  |  |  |

**Raspberry Pi Hardware Preparation and Installation**

**Hardware Guide:**

For getting started with raspberry pi for the first time you will require the following hardware

1. Raspberry Pi (latest Model)
2. Monitor or TV
3. HDMI cable
4. Ethernet cable
5. USB keyboard
6. USB mouse
7. Micro USB power supply
8. 8GB or larger microSD card
9. SD Card Reader

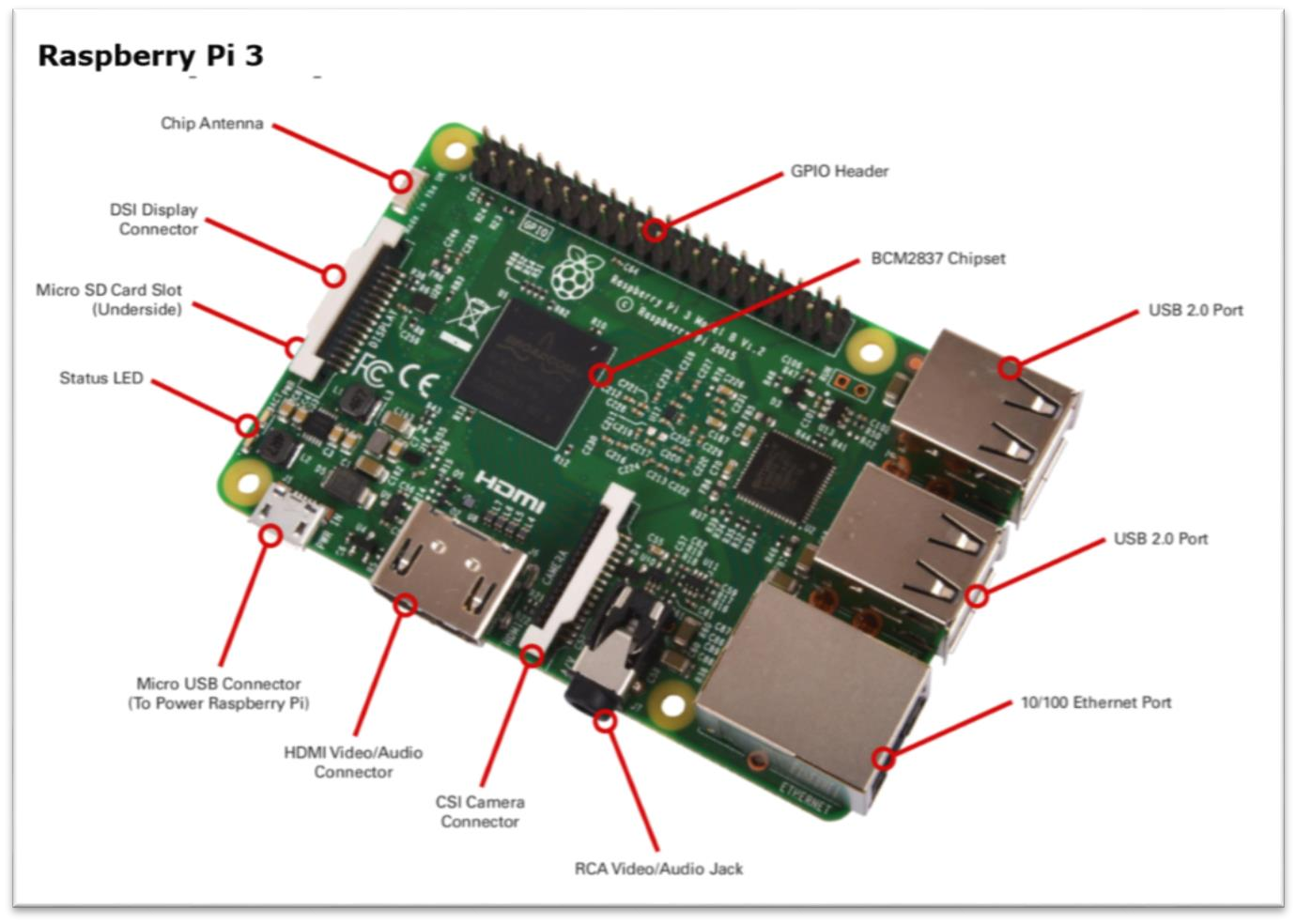
**Raspberry Pi 3 Model B:**

The Raspberry Pi 3 is the third generation Raspberry Pi. It replaced the Raspberry Pi 2 Model B in February 2016. Compared to the Raspberry Pi 2 it has:

* A 1.2GHz 64-bit quad-core ARMv8 CPU
* 802.11n Wireless LAN
* Bluetooth 4.1
* Bluetooth Low Energy (BLE)

Like the Pi 2, it also has:

* 4 USB ports
* 40 GPIO pins
* Full HDMI port
* Ethernet port
* Combined 3.5mm audio jack and composite video
* Camera interface (CSI)
* Display interface (DSI)
* Micro SD card slot (now push-pull rather than push-push)
* Video Core IV 3D graphics core
* The Raspberry Pi 3 has an identical form factor to the previous Pi 2 (and Pi 1 Model B+) and has complete compatibility with Raspberry Pi 1 and 2.

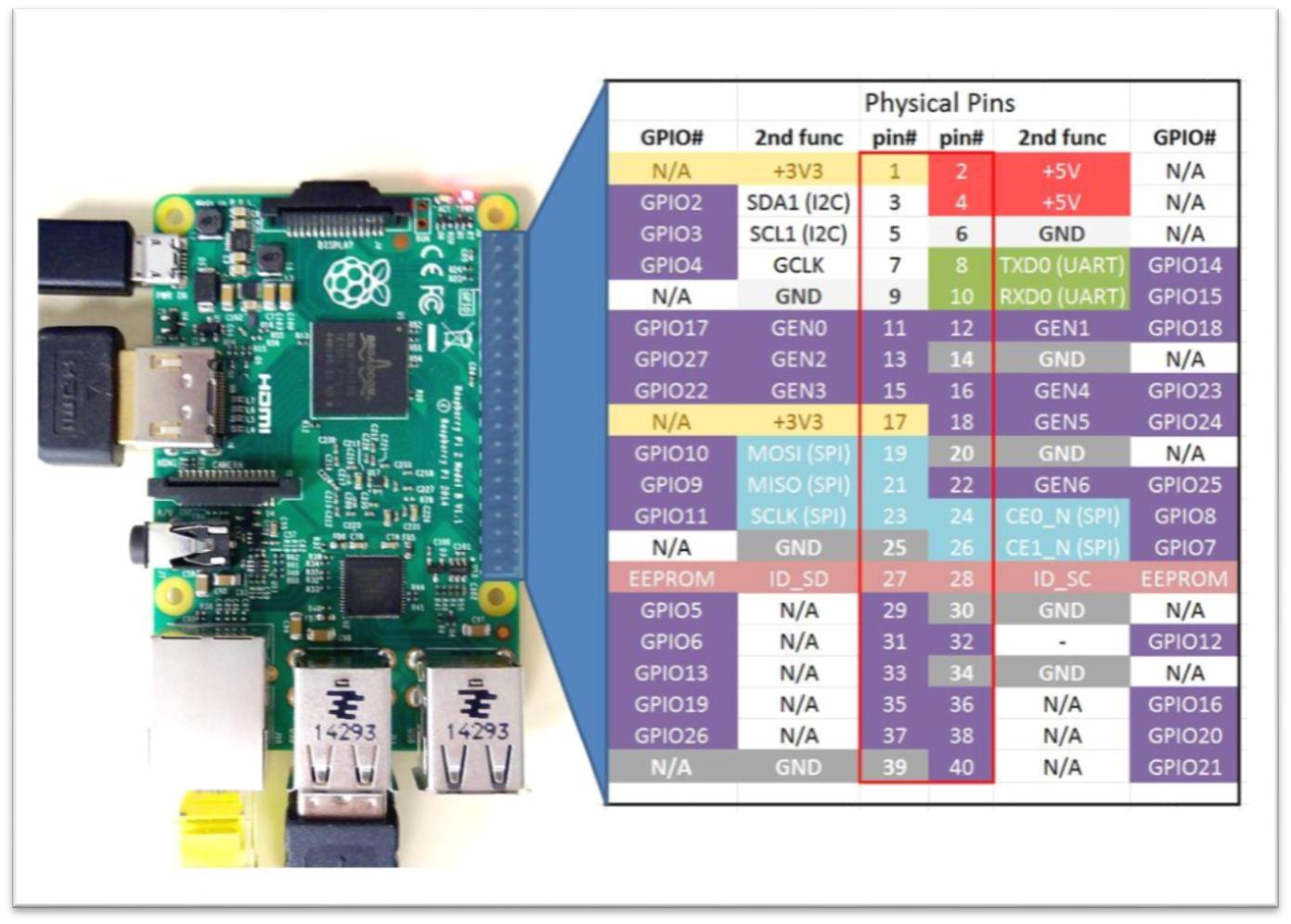


**A note on pin numbering:**

When programming the GPIO pins there are two different ways to refer to them: GPIO numbering and physical numbering.

GPIO numbering: These are the GPIO pins as the computer sees them. The numbers don't make any sense to humans, they jump about all over the place, so there is no easy way to remember them. You will need a printed reference or a reference board that fits over the pins.

Physical numbering: The other way to refer to the pins is by simply counting across and down from pin 1 at the top left (nearest to the SD card).



|  |  |
| --- | --- |
| **Bread Board** |  |
| **LED** |  |
| **Resistors: 220 K, 10 K and 1K** |  |
| **Jumper wires: M to M**  **F to M** |  |
| **8 LED Pattern** |  |
| **4-digit 7-segment display module** |  |
| **GPS Module** |  |
| **RFID Sensor and Card** |  |
| **Camera Module** |  |
| **Finger-print sensor** |  |
| **Push Buttons** |  |
| **USB to TTL Converter** |  |

**Practical 1**

**: Starting Raspbian OS, Familiarizing with Raspberry Pi Components and interface, Connecting to ethernet, Monitor, USB.**

**Accessories needed**:

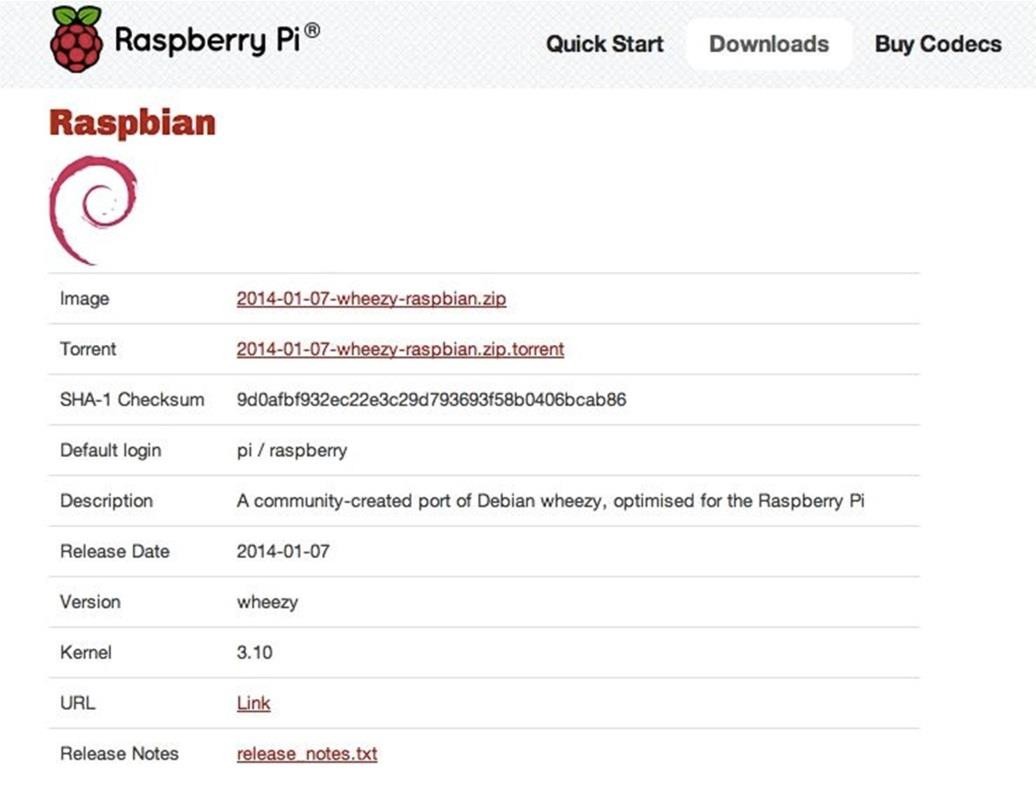
1. Raspberry Pi 5. An SD memory card.
2. A power supply 6. An SD memory card reader.
3. An HDMI cable. 7. Internet connection. (Ethernet or

wireless)

1. A USB mouse and keyboard.

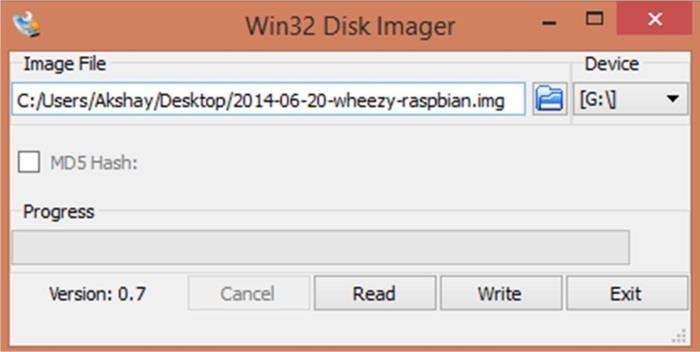
**Steps to make bootable SD card for Raspberry Pi (Raspbian stretch)**

* 1. Get Raspbian OS on your micro SD card: Download Raspbian from raspberrpi.org and click download and then click on Raspbian (select Raspbian Stretch with desktop file.) mostly as zip file.

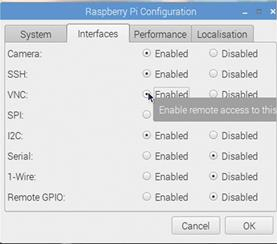


* 1. Unzip file to find a disc image.
  2. Format new SD card using SD Formatter tool.
  3. You need to install Win32DiskImager tool to write image to SD card.

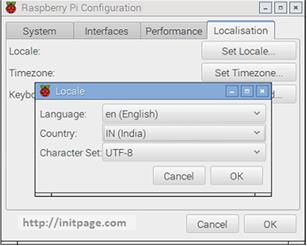
1. In Win32DiskImager tool select image file and click Write button.



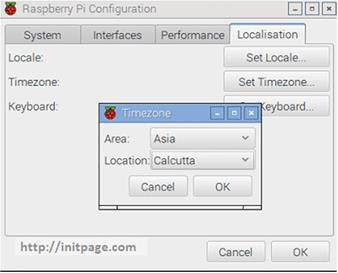
1. Once it is written your card name will change into boot.
2. Now can insert this card in your Raspberry Pi.
3. Using the Raspberry Pi Configuration tool: Presently, the Raspberry Pi Configuration tool displays four tabs:
   * + System. Options to expand the file system and change password and hostname sit alongside various login choices.
     + Interfaces. Support for the various hardware and software features, such as Camera Module, SSH, and VNC.
     + Performance. Overclocking and GPU memory options can improve the performance of a Raspberry Pi.
     + Localization Set up an international keyboard, global Wi-Fi options, and adjust the locale and time zone.



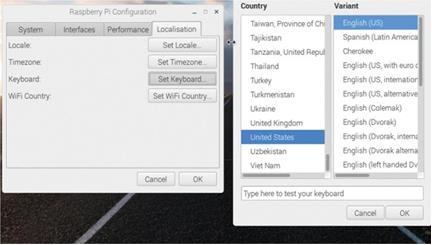
1. Set Locale option: Language as English, Country as India and Character Set as UTF-8.



1. Set Time zone option: Select Area as Asia and Location as Calcutta



1. Set keyboard option



1. Set Wi-Fi country option: Select India here.



**Practical 2**

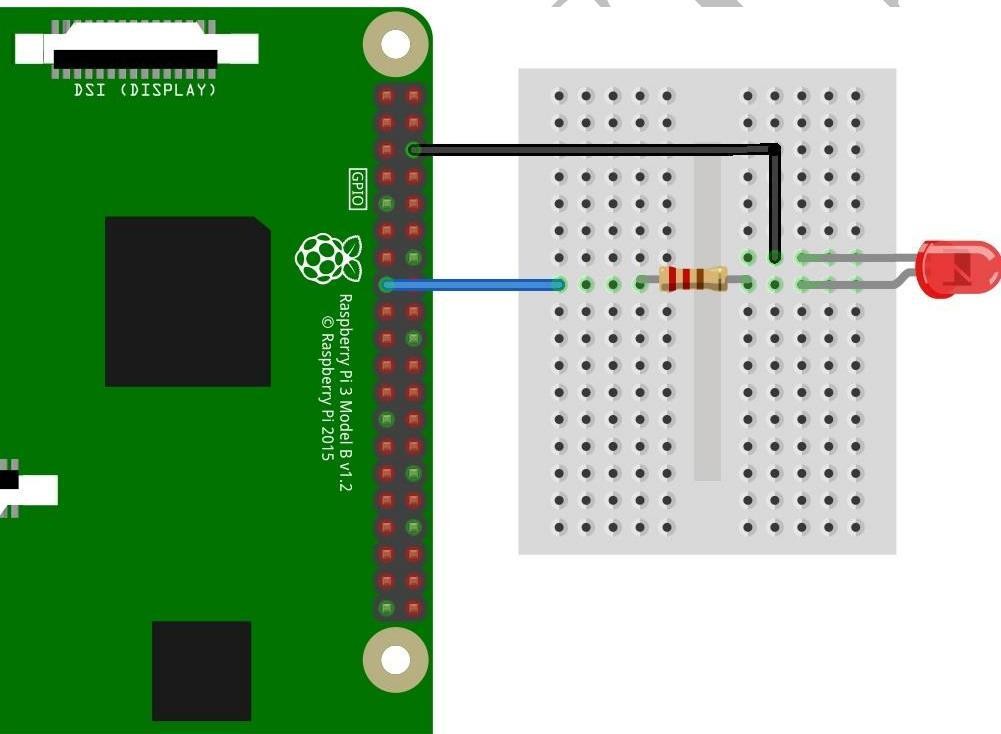
**: Displaying different LED patterns with Raspberry Pi.**

**Additional Hardware required:**

* + - 1. Resistors 3. Jumper wires
      2. LEDs 4. Breadboard

**Connections:**

1. Connect the GPIO22 (i.e. Physical Pin 15) Pin of raspberry pi to one end of the resistor.
2. Connect another end of resistor to the positive end (anode) of LED
3. Connect the negative end (cathode) of LED to Ground of raspberry pi.
4. Then Power on your raspberry pi



To open Python, click on the application Menu, navigate to Programming, and then click on Python 2 (IDLE) an Integrated Development Environment for Python 2. After opening the IDE, go to files and open new file to start your code.

**Code for Single Blinking LED:**

importRPi.GPIO as GPIO from time import sleep GPIO.setmode(GPIO.BOARD)

ledPin = 15

GPIO.setup(ledPin, GPIO.OUT) GPIO.output(ledPin, False) try: while True: GPIO.output(ledPin, True) print("LED ON") sleep(1) GPIO.output(ledPin, False) print("LED OFF") sleep(1) finally:

GPIO.output(ledPin, False)

GPIO.cleanup()

**Code for 8 Blinking LED:**

importRPi.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)

defledpattern(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)

defpatterOne():

fori 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)

defpatternTwo():

fori 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) defpatternThree():

fori 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)

defpatternFour():

fori 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)

defpatternFive():

fori 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: patterOne() patternTwo() patternThree() patternFour() patternFive()

finally:

GPIO.cleanup()

# Practical 3

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

**Additional Hardware requires:**

1. TM1637 4-digit seven segment Display board
2. Jumper wires

**Wiring up your Circuit:**

1. Connect the Pin2 of Rpi to VCC pin of Module
2. Connect Pin 6 of Rpi to GND of Module
3. Connect Pin38 of Rpi to DIO of Module
4. Lastly connect Pin 40 of Rpi to CLK of Module

**Software Guide:**

1. Download libraries from: *https://github.com/timwaizenegger/raspberrypi-examples/tree/master/actorled-7segment-4numbers.*
2. Download the zip file named:” actor-led-7segment-4numbers.zip”
3. Browse on to /home/pi/Downloads/ location on Rpi to find the zip file downloaded.
4. Unzip the file and try to execute the different example codes present in that folder in Python 2 Idle.
5. 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.

**Code : clock.py** from time import sleep import tm1637 try:

import thread exceptImportError:

import \_thread as thread

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:

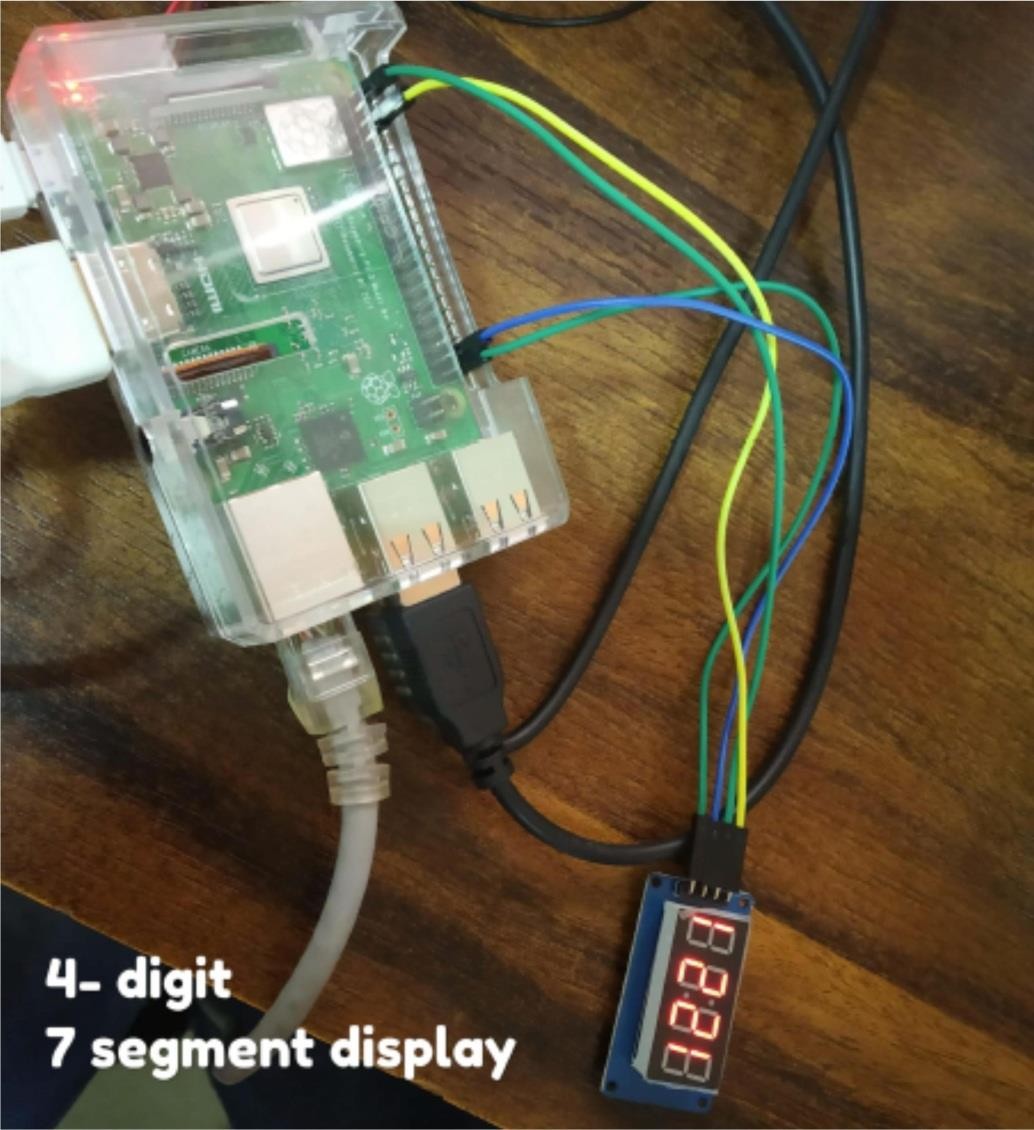
Display.ShowDoublepoint(True) sleep(1)

Display.ShowDoublepoint(False)

sleep(1)

Display.StopClock() thread.interrupt\_main() exceptKeyboardInterrupt: print "Properly closing the clock and open GPIO pins"

Display.cleanup()



**OUTPUT**

**:**

# Practical 4

# :GPS Module interfacing with Raspberry Pi

**Hardware Guide:**

For completing this lesson, you will require the following things along with your initial raspberry pi setup

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

**Wiring up 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:**

1. Open Terminal Window and type the following command to know to which USB port the GPS module is attached: ***ls /dev/ttyUSB\****

1. To find whether our GPS module is working properly and the connections are correct by typing the following command: ***sudo cat /dev/ttyUSB\**** (Replace \* with the port number to which GPS module is attached.)

1. To install GPS Daemon (gpsd) run the following commands :
   1. sudo apt-get update
   2. sudo apt-get install gpsdgpsd-clients python-gps

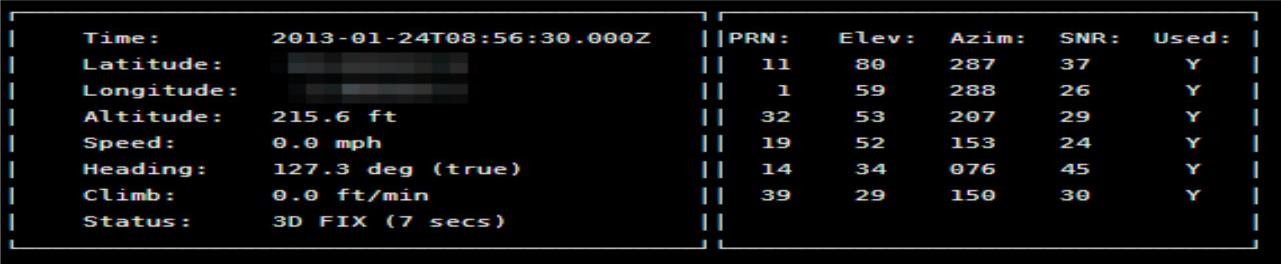
1. You now need to disable the gpsdsystemd service by running the following commands:
   1. sudosystemctl stop gpsd.socket
   2. sudosystemctl disable gpsd.socket

1. After installing gpsd and disabling the gpsdsystemd service type the following command to get live streaming gps data **sudogpsd /dev/ttUSB0 -F /var/run/gpsd.sock**

1. If the gps has proper range we shall get the following output:



1. Type ***cgps –s*** to get a less detailed and nice output as follows:



**CODE:**

import serial from time import sleep importwebbrowser

import sys

defGPS\_Info():

globalNMEA\_buff

globallat\_in\_degrees globallong\_in\_degrees nmea\_time = [] nmea\_latitude = [] nmea\_longitude = [] nmea\_time = NMEA\_buff[0] nmea\_latitude = NMEA\_buff[1]

nmea\_longitude = NMEA\_buff[3]

print("NMEA Time: ", nmea\_time,'\n')

print ("NMEA Latitude:", nmea\_latitude,"NMEA Longitude:", nmea\_longitude,'\n')

lat = float(nmea\_latitude) longi = float(nmea\_longitude) lat\_in\_degrees = convert\_to\_degrees(lat)

long\_in\_degrees = convert\_to\_degrees(longi)

defconvert\_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/ttyUSB0")

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) if (GPGGA\_data\_available>0):

GPGGA\_buffer = received\_data.split("$GPGGA,",1)[1]

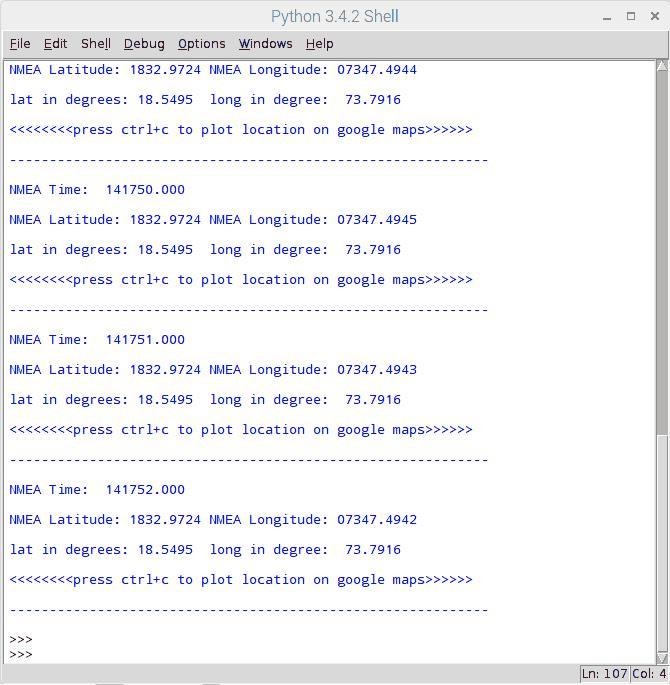
NMEA\_buff = (GPGGA\_buffer.split(','))

GPS\_Info()

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

print("press ctrl+c to plot location on google maps\n") print("------------------------------------------------------------\n") exceptKeyboardInterrupt:

webbrowser.open(map\_link) sys.exit(0)



**OUTPUT:**

**Practical 5**

**: Interfacing Raspberry Pi with RFID.**

**Hardware Guide:**

1. RFID module (EM-18 RFID Reader Module and RFID cards)
2. USB to TTL converter
3. Connecting wires
4. 5V external Power Supply for RFID Reader **Wiring up your Circuit:**

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

1. Connect the positive of 5V external supply to VCC pin of module.
2. Connect the negative/GND of 5V external supply to GND of Module
3. Finally connect the USB to TTL converter to USB of raspberry Pi

**CODE:**

import time import serial

defread\_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":

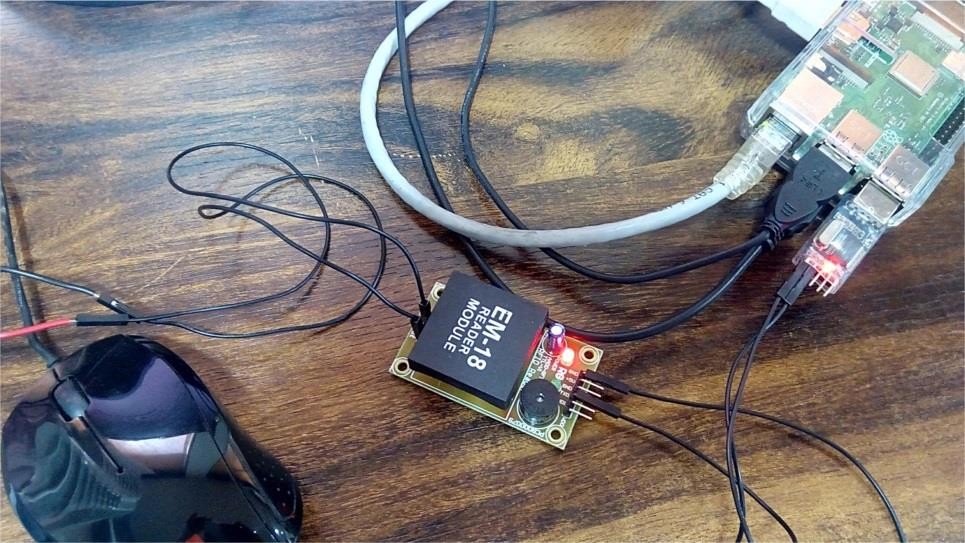
print("Acces Granted")

else:

print("Access Denied")

exceptKeyboardInterrupt:

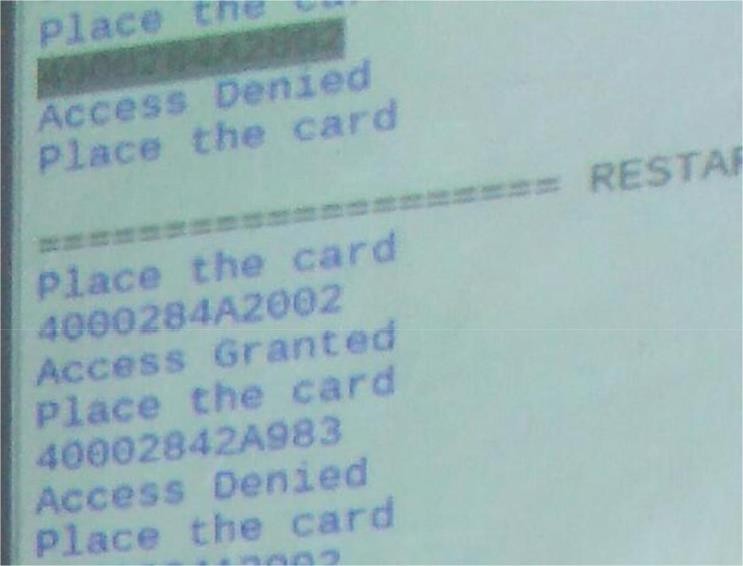
data.close()



**OUTPUT:**

**Python Output on IDLE**

**:**



# Practical 6

# : Visitor Monitoring with Raspberry Pi and Pi Camera

**Additional Hardware Required:**

1. Pi camera 5. Resistor (1k,10k)
2. DC Motor 6. Push Button
3. IC L293D 7. Jumper wires
4. Bread Board 8. External 5V supply (adapter)

**Connections:**

1. Connect Pi camera to Raspberry Pi by removing case like following: (pull out slot , Insert camera in , Press slot again properly) 2. Connect DC motor to L293D driver module with screw driver.



1. Connect push button of Breadboard.
2. Connect on leg of push button to Resistor (Brown Black Orange ) 10K.

**Jumper wire Connections:**

1. Connect driver module A1 pin to Pin 11 of Pi.
2. Connect driver module A2 pin to Pin 13 of Pi.
3. Connect driver module EN pin to external +5v taken on breadboard.
4. Connect driver module VCC pin to external +5v taken on breadboard.
5. Connect driver module GND pin to external -5v taken on breadboard.
6. Connect pin 25 on Pi to pushbutton leg which is not connected to resistor.
7. Connect pin 35 on Pi to pushbutton leg which is also connected to resistor.
8. Connect pin 1 on Pi to other leg of resistor which is not connected to pushbutton.
9. Make common ground by connecting Pin 6 on Pi to external -5v taken on breadboard.

**Software configuration:**

Install Pi camera library files by typing the following commands in the terminal

***$sudo apt-get install python-picamera***

***$sudo apt-get install python3-picamera***

**CODE*:***

import RPi.GPIO as gpio import picamera import time

m11=17 m12=27 led=5 buz=26 button=19

HIGH=1 LOW=0

gpio.setwarnings(False) gpio.setmode(gpio.BCM) gpio.setup(led, gpio.OUT) gpio.setup(buz, gpio.OUT) gpio.setup(m11, gpio.OUT) gpio.setup(m12, gpio.OUT) gpio.setup(button, gpio.IN) gpio.output(led , 0) gpio.output(buz , 0) gpio.output(m11 , 0) gpio.output(m12 , 0) data=""

def capture\_image():

print("Please Wait..");

data= time.strftime("%d\_%b\_%Y\%H:%M:%S") camera.start\_preview() time.sleep(5) print (data) camera.capture('/home/pi/Desktop/Visitors/%s.jpg'%data) camera.stop\_preview() print("Image Captured Successfully") time.sleep(2)

def gate():

print("Welcome ") gpio.output(m11, 1) gpio.output(m12, 0) time.sleep(1.5) gpio.output(m11, 0) gpio.output(m12, 0) time.sleep(3) gpio.output(m11, 0) gpio.output(m12, 1) time.sleep(1.5) gpio.output(m11, 0) gpio.output(m12, 0) print(" Thank You ") time.sleep(2)

print("Visitor Monitoring") print("Using RPI") time.sleep(3) camera = picamera.PiCamera() camera.rotation=180 camera.awb\_mode= 'auto' camera.brightness=55 time.sleep(2)

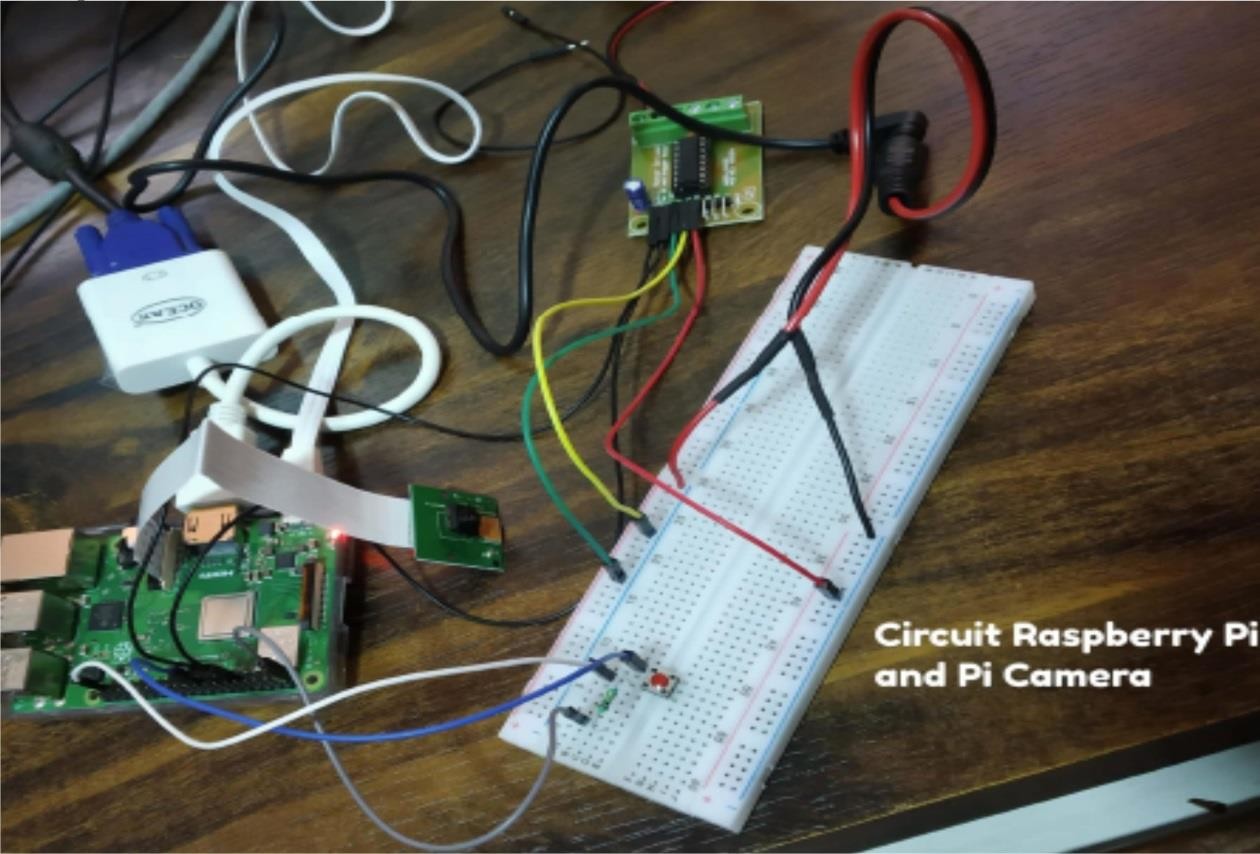
while 1:

print(" Please Press Button") print(" to open the gate ") gpio.output(led, 1) if gpio.input(button)==0: gpio.output(buz, 1) gpio.output(led, 0) time.sleep(0.5) gpio.output(buz, 0) capture\_image()

gate()

time.sleep(0.5)

**OUTPUT:**



# Practical 7

# : Fingerprint Sensor interfacing with Raspberry Pi



**Hardware Guide:**

1. Fingerprint Sensor
2. USB to TTL/UART converter
3. Connecting wire.

**Connections:**

1. Connect the Red wire of Finger Print sensor to VCC of USB to TTL Converter.
2. Connect the Yellow wire to Rx of USB to TTL Converter.
3. Connect the White wire to Tx of USB to TTL Converter 4. Connect the Black wire to the GND of USB to TTL Converter.

**Software Guide:**

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

1. Then download some required packages by using given commands:
2. wget –O – http://apt.pm-codeworks.de/pm-codeworks.de.gpg | apt-key add –
3. 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:

1. **sudo apt-get update**
2. **sudo apt-get install python-fingerprint**
3. now exit root by typing **exit**

**CODE: Enroll.py** import time from pyfingerprint.pyfingerprint import PyFingerprint

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.py

import hashlib from pyfingerprint.pyfingerprint import PyFingerprint 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.py

from pyfingerprint.pyfingerprint import PyFingerprint 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()))

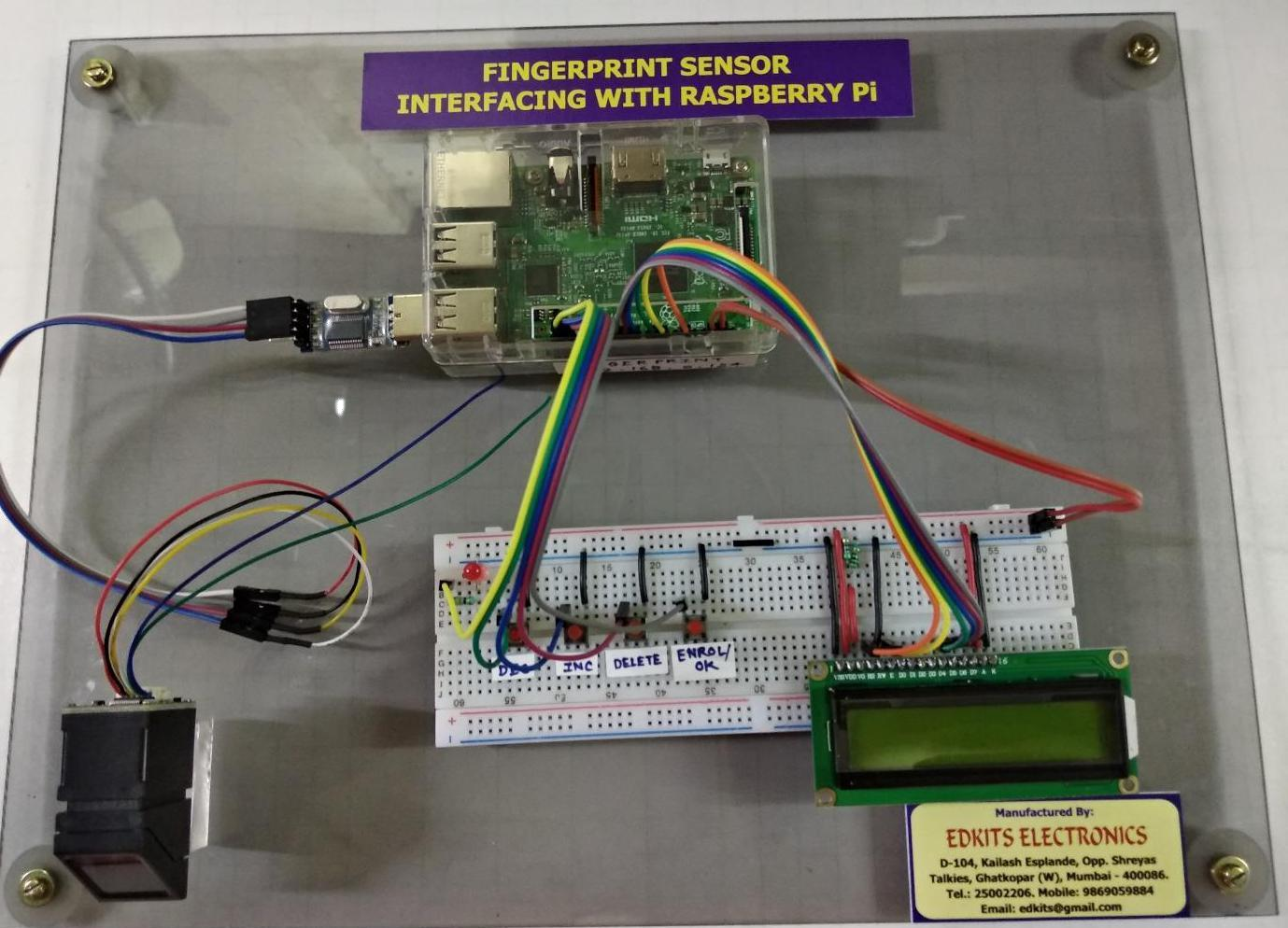
## Tries to delete the template of the finger try:

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:**



**Practical 8**

**: Controlling Raspberry Pi with Telegram.**

**Additional Hardware Required:**

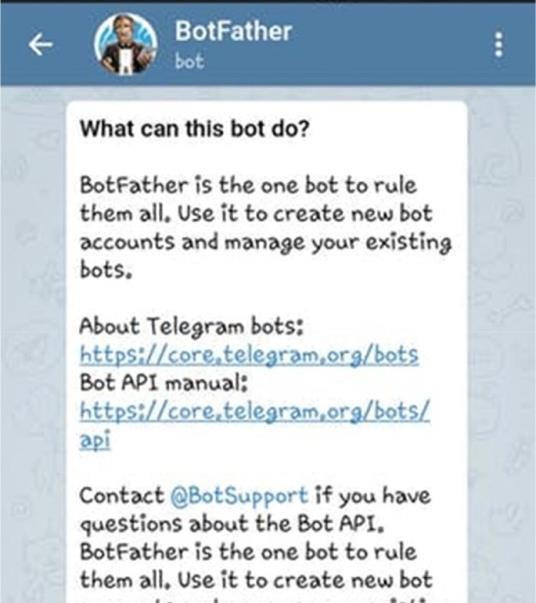
1.LED 3.Resistor

2.Breadboard 4.Jumper wires

**Software required:** On Mobile Phone: Telegram

**Steps:**

1. Download Telegram from play store on your android phone.
2. Install Telegram.
3. Open Telegram app in your system or mobile
4. Click On Start Messaging Button
5. Enter your mobile number to register with telegram service.
6. Search for name "BotFather"

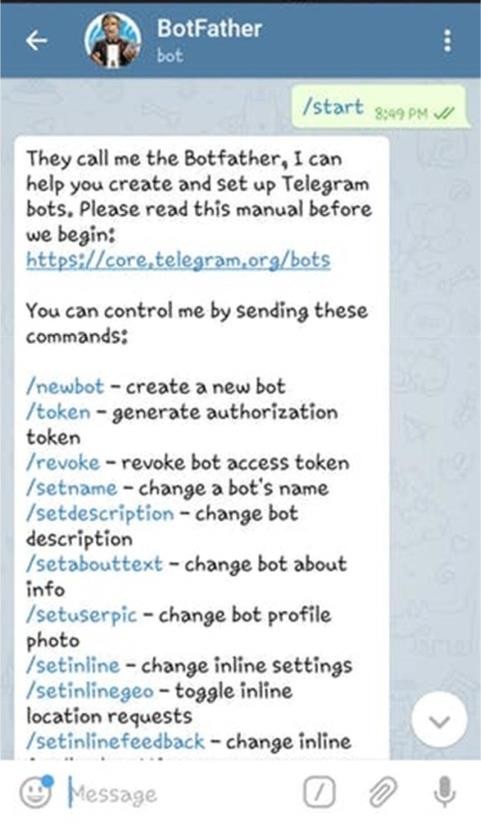


7.

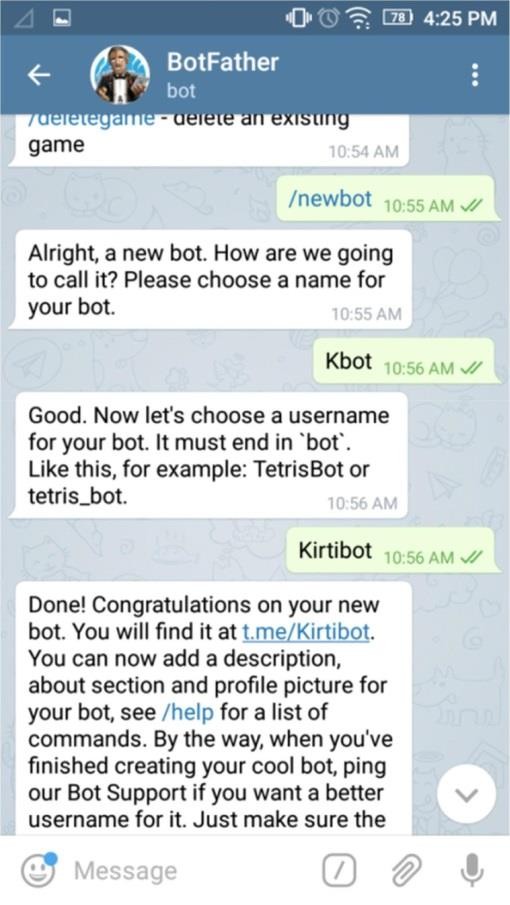
Click on "BotFather



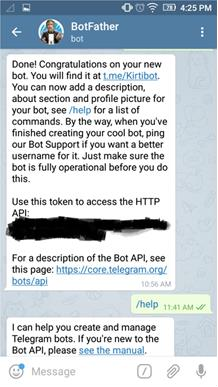
1. To Start "BotFather" type /start in message



1. Now type /newbot in message . and then give name to your BOT and Username also.



1. Obtain access token



This same token we are supposed to use in our code in raspberry pi to connect to our BOT.

1. **Set up On Raspberry Pi**: Install "Python Package Index" and Telepot using : **apt-get install python-pip sudo pip install teleport**

**CODE:**

import sys import time import random import datetime import telepot import RPi.GPIO as GPIO GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

def handle(msg):

chat\_id = msg['chat']['id'] command = msg['text'] print 'Got command:', command if command == 'on':

bot.sendMessage(chat\_id, "LED on")

GPIO.output(11,GPIO.HIGH) elif command =='off':

bot.sendMessage(chat\_id, "LED off")

GPIO.output(11,GPIO.LOW) elif command == 'stop':

exit()

try:

bot = telepot.Bot('Bot Token') bot.message\_loop(handle) print 'I am listening...' while 1:

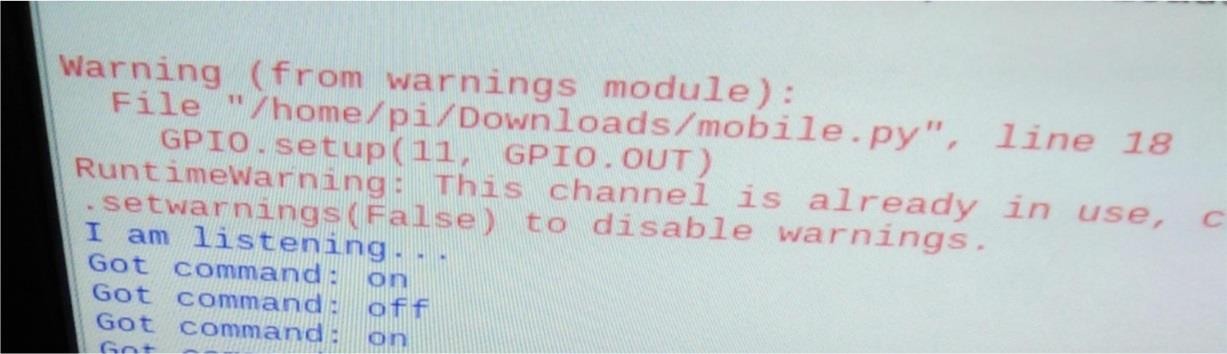
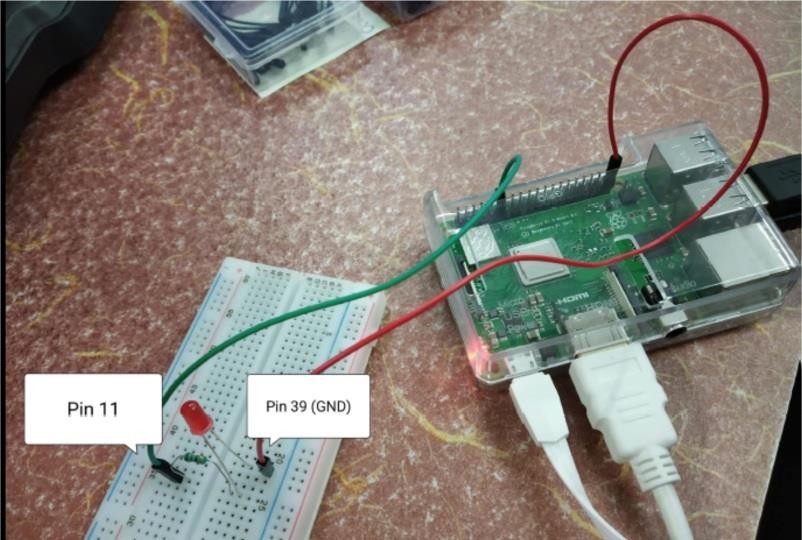
time.sleep(10)

except TelegramError:

print ' '

**Note: do not forget to replace ‘Bot Token’ with your token received from telegram.**

**OUTPUT:**



**Output on python**

**IDLE:**

**Output on Telegram (Mobile phone):**

