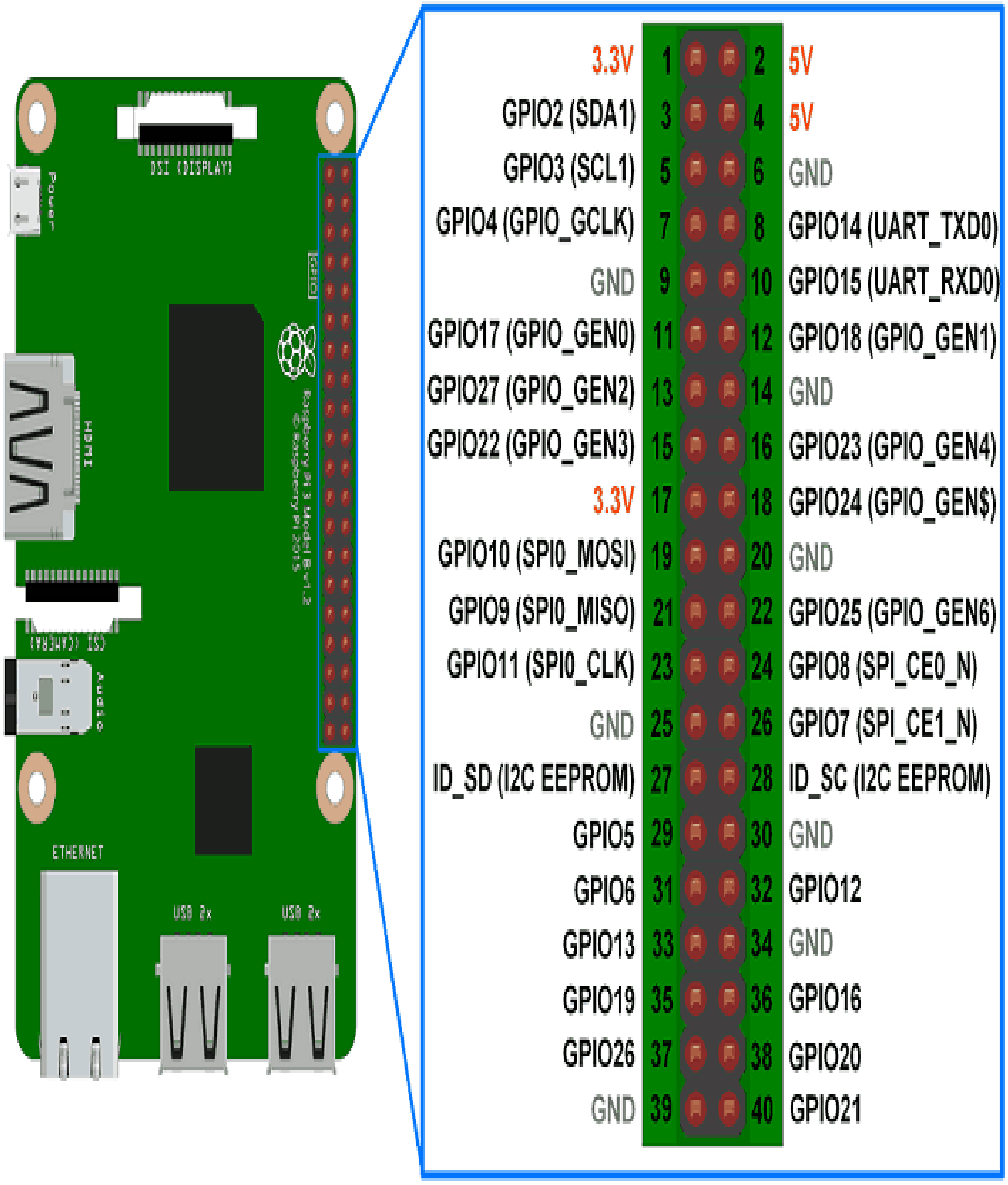
INDEX

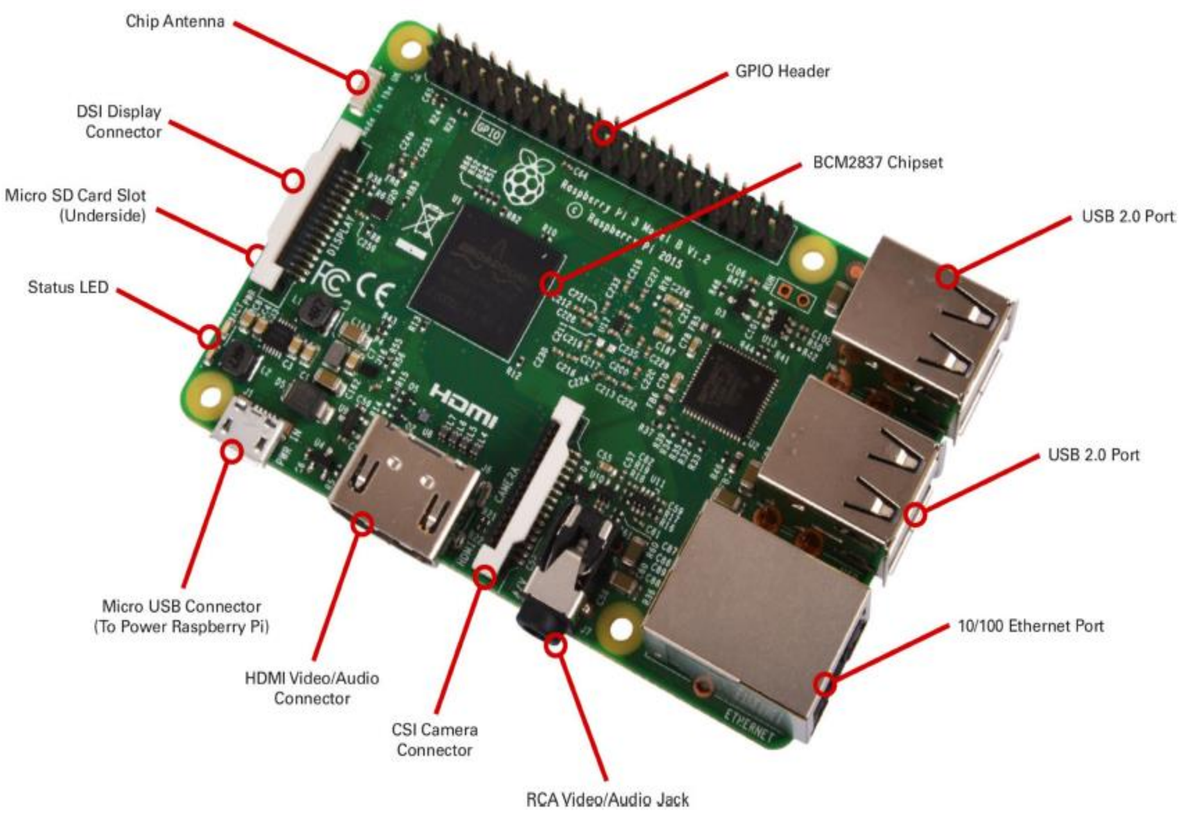
# Introduction to Raspberry Pi

GPIO PINS

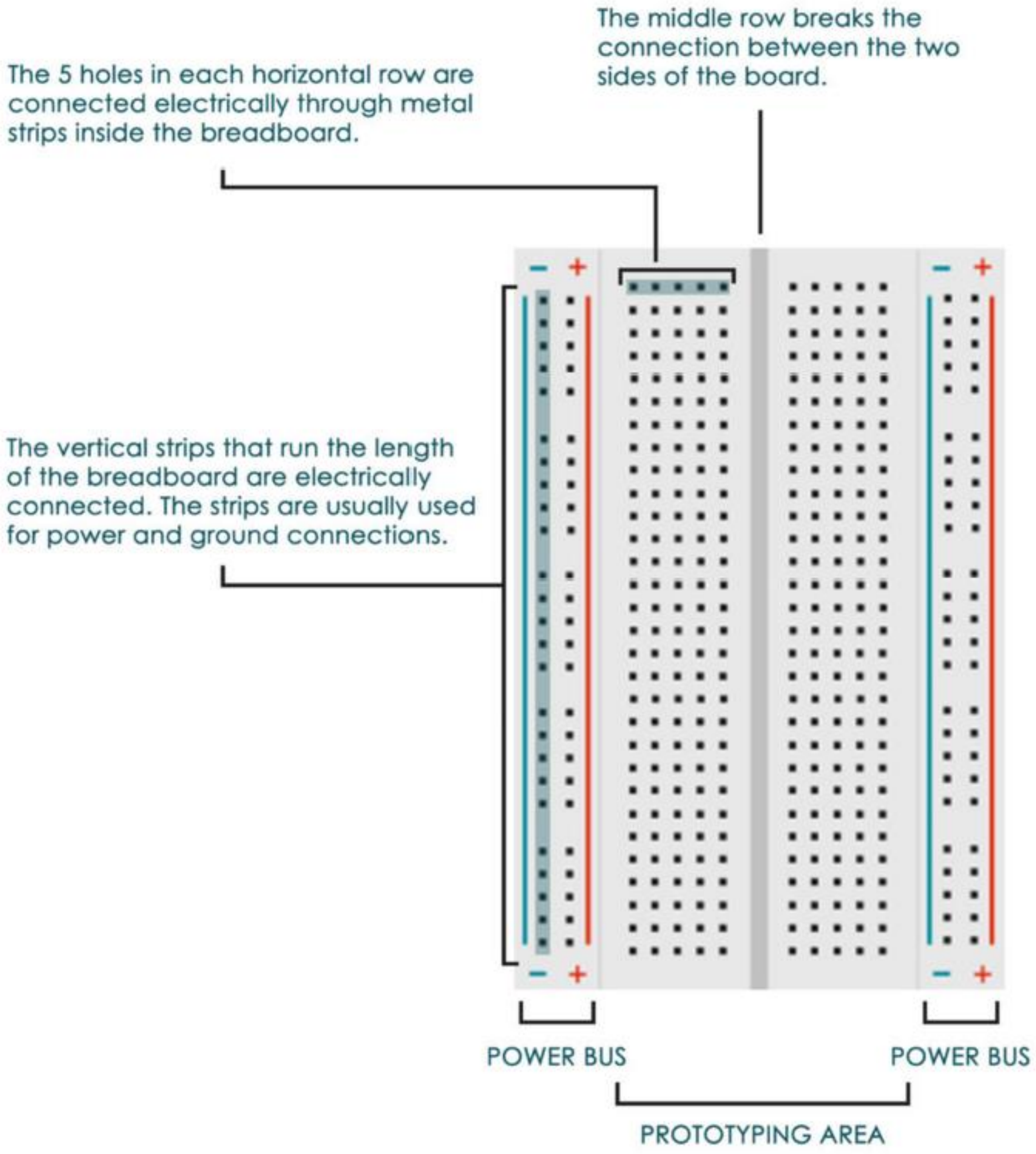


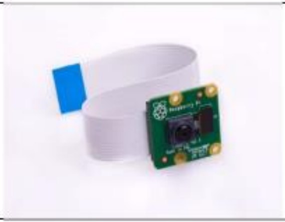
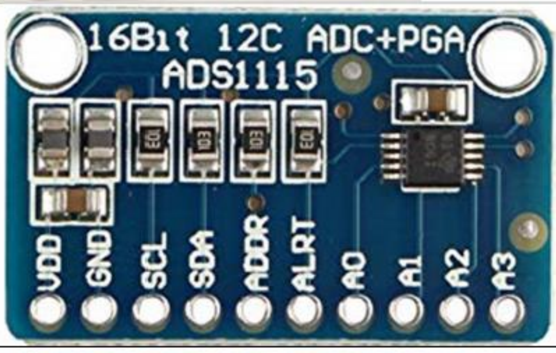
DSI (DISPLAY)
Power
HDMI
CSI (CAMERA)
Audio
ETHERNET
USB 2x
USB 2x
GPIO
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
3.3V 1 2 5V
GPIO2 (SDA1) 3 4 5V
GPIO3 (SCL1) 5 6 GND
GPIO4 (GPIO_GCLK) 7 8 GPIO14 (UART_TXD0)
GND 9 10 GPIO15 (UART_RXD0)
GPIO17 (GPIO_GEN0) 11 12 GPIO18 (GPIO_GEN1)
GPIO27 (GPIO_GEN2) 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

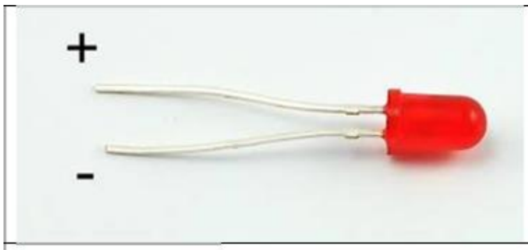
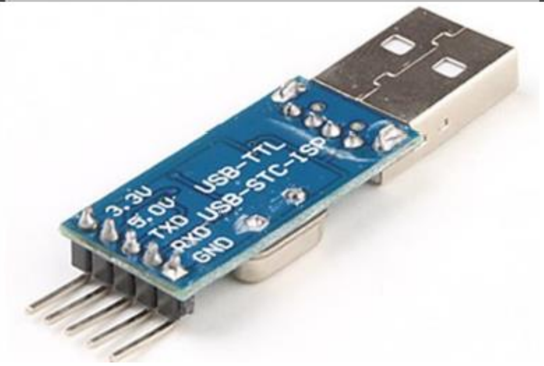
**Hardware Components used:**


Chip Antenna
GPIO Header
DSI Display Connector
BCM2837 Chipset
Micro SD Card Slot (Underside)
USB 2.0 Port
Status LED
USB 2.0 Port
Micro USB Connector (To Power Raspberry Pi)
10/100 Ethernet Port
HDMI Video/Audio Connector
CSI Camera Connector
RCA Video/Audio Jack

# BREAD BOARD


The middle row breaks the connection between the two sides of the board.
The 5 holes in each horizontal row are connected electrically through metal strips inside the breadboard.
The vertical strips that run the length of the breadboard are electrically connected. The strips are usually used for power and ground connections.
POWER BUS
POWER BUS
PROTOTYPING AREA

| | |
|---|---|
|  | Camera Module: |
| | 4-digit 7-segment display module: |
| | 2 channel 5V relay module: |
| EM-18 READER MODULE 0003516710 053,43302 0003566892 054,46948 | RFID cards and Reader module: |
|  16Bit 12C ADC+PGA ADS1115 VDD GND SCL SDA ADDR ALRT A0 A1 A2 A3 | ADS115 module: |

| | |
|---|---|
|  | LED: |
| | Push button: |
| | 220 K , 10 K and 1K resistors: |
| | Potentiometer (POT): |
| | M-F and F-F jumper wires: |
|  | USB to TTL converter: |

## Practical No :1

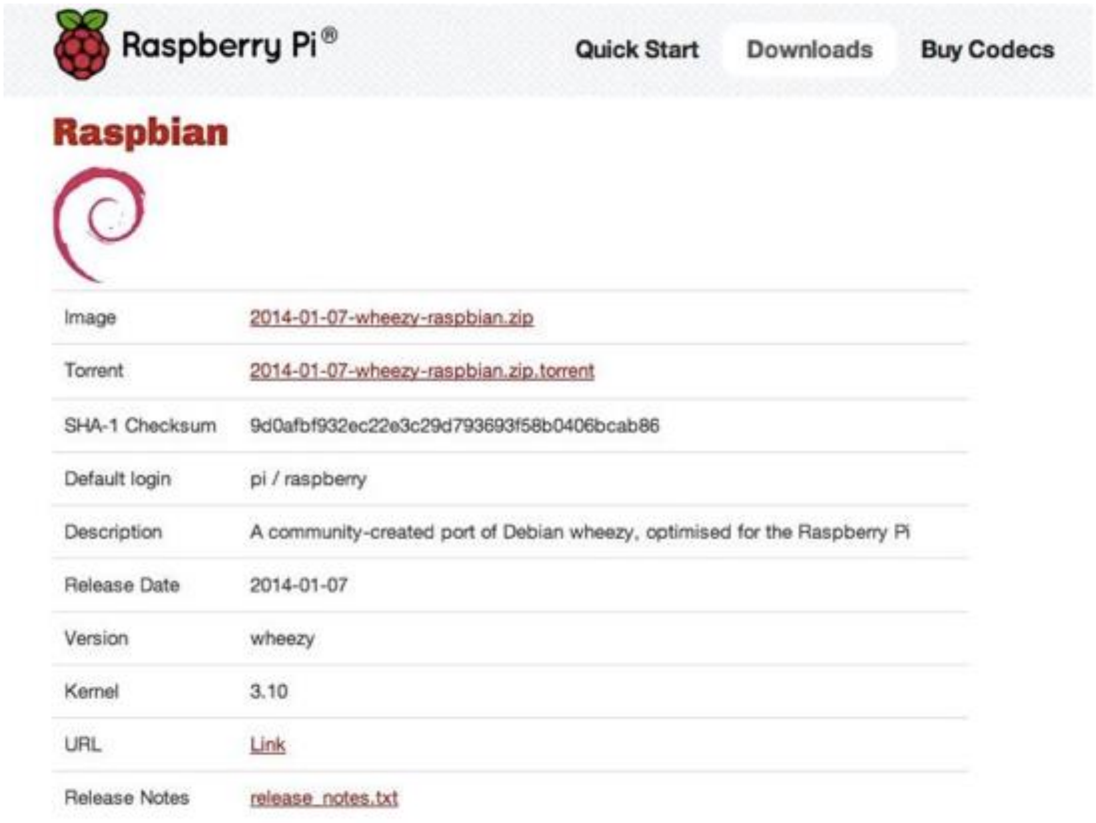
**Aim:** Starting Raspbian OS, Familiarising with Raspberry Pi Components and interface, Connecting to ethernet, Monitor, USB.

**Hardware Required:**

1. Raspberry Pi
2. A power supply
3. An HDMI cable.
4. A USB mouse and keyboard.
5. An SD memory card.
6. An SD memory card reader.
7. Internet connection. (Ethernet or wireless)

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

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



Raspbian

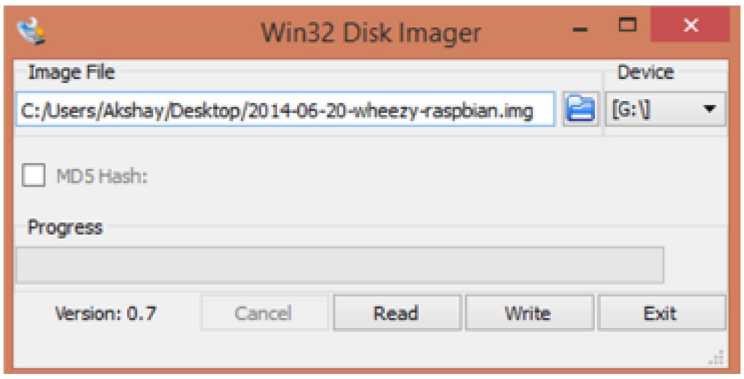
| | |
|---|---|
| Image | 2014-01-07-wheezy-raspbian.zip |
| Torrent | 2014-01-07-wheezy-raspbian.zip.torrent |
| SHA-1 Checksum | 9d0afbf932ec22e3c29d793693f58b0406bcab86 |
| Default login | pi / raspberry |
| Description | A community-created port of Debian wheezy, optimised for the Raspberry Pi |
| Release Date | 2014-01-07 |
| Version | wheezy |
| Kernel | 3.10 |
| URL | Link |
| Release Notes | release_notes.txt |

3. Unzip file to find a disc image.

4. Format new SD card using SD Formatter tool.

5. You need to install Win32DiskImager tool to write image to SD card.

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



7. Once it is written your card name will change into boot.

8. Now can insert this card in your Raspberry Pi.

**Configuring Pi:**

Raspberry Pi comes with a default user name and password and so always use it whenever it is being asked. The credentials are:

```
login: pi
password: raspberry
```

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

We can open this window with this command:

$sudo raspi-config

and configure the raspberry Pi

## Practical No:2

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

**Hardware required:**

1. Resistors
2. LEDs
3. Jumper wires
4. Breadboard

**Connections:**

1. Connect LEDs on breadboard
2. Connect resistor to one end of led (to Positive end or anode).
3. Connect other end of resistor to GPIO pin on raspberry Pi.
4. Connect other end of LED to ground (negative end or cathode).

**Write following code in Python IDLE 3 save it as 'led pattern.py'**

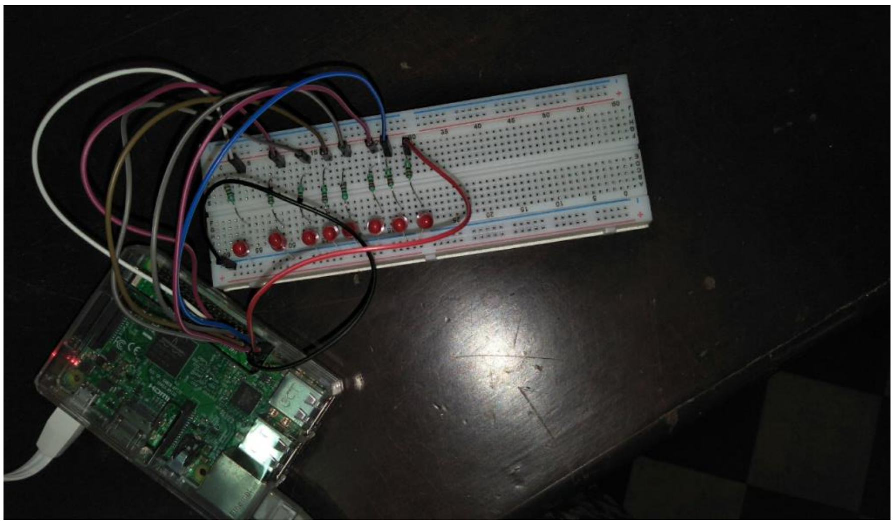
```
#CONNECT 8 LED IN THE SAME FORMAT TO THE PIN NUMBERS

# Code for LED pattern:

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
```

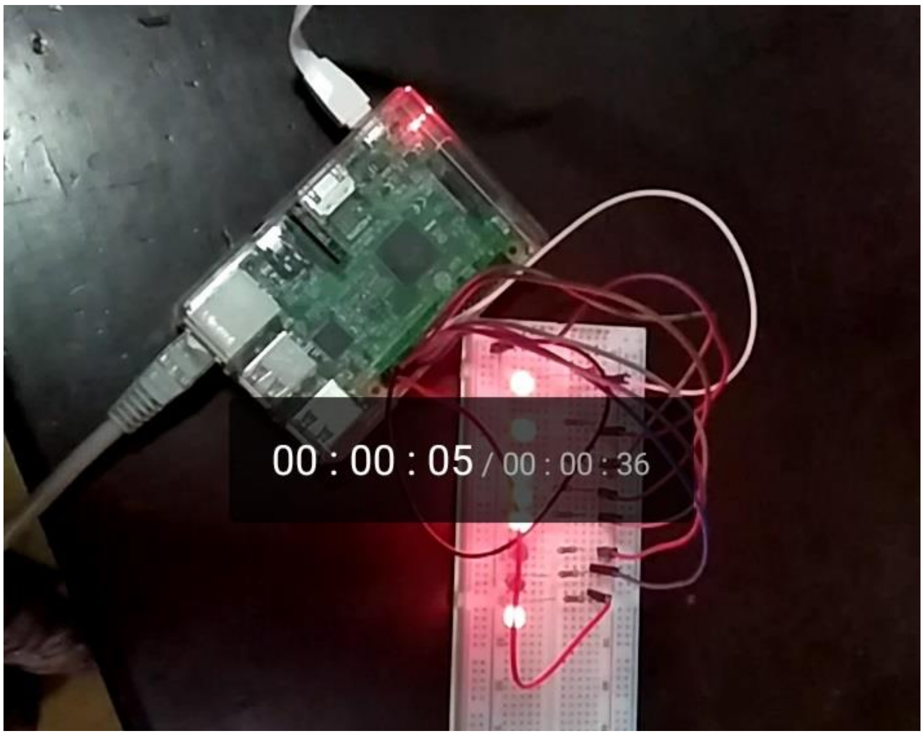
```
#setup the ledPin(i.e GPIO22) as output

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:
                #reset the GPIO Pins
                GPIO.cleanup()
```




00 : 00 : 05 / 00 : 00 : 36

## Practical No: 3

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

**Hardware required**:

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

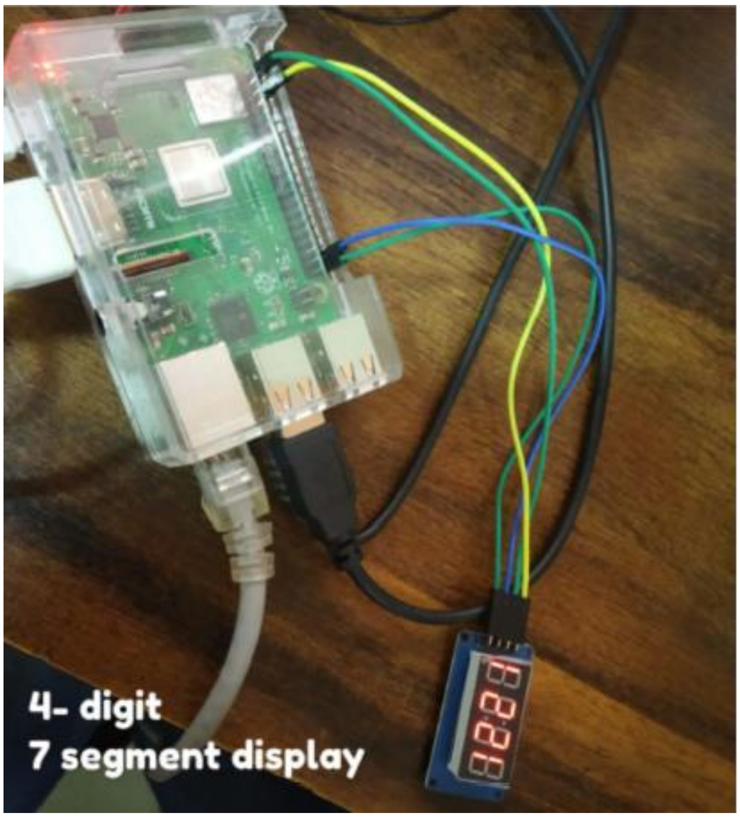
| Sr No | T1637 Pin Name | RPi Pin | RPi Function |
|---|---|---|---|
| 1 | Vcc | 2 | 5V |
| 2 | GND | 6 | GND |
| 3 | DIO | 38 | GPIO 20 |
| 4 | Clk | 40 | GPIO 21 |

**Libraries needed:**

tm1637.py is a driver library. Download and save it in same folder as your code.

**Write the the following code  and save it as timedisplay.py:**

```
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 CLK 21)

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()
```



## PRACTICAL No: 4

**Aim:** Controlling Raspberry Pi with Telegram.

**Hardware Required:**

1. LED
2. 2. Breadboard
3. 3. Resistor
4. 4.Jumper wires

**Software required**:

On Mobile Phone: Telegram

**Steps to be followed:**

1: Download Telegram from PlayStore on your android phone.

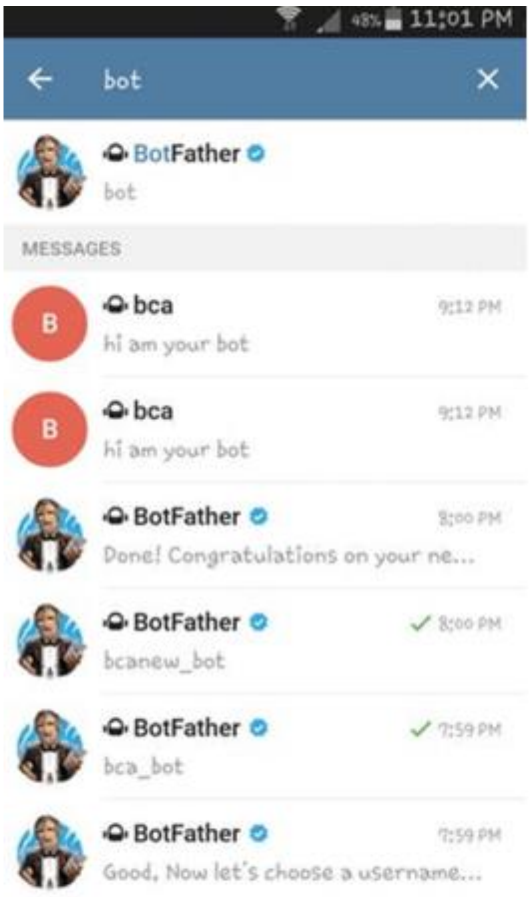
2: Install Telegram.

2: Open Telegram app in your system or mobile

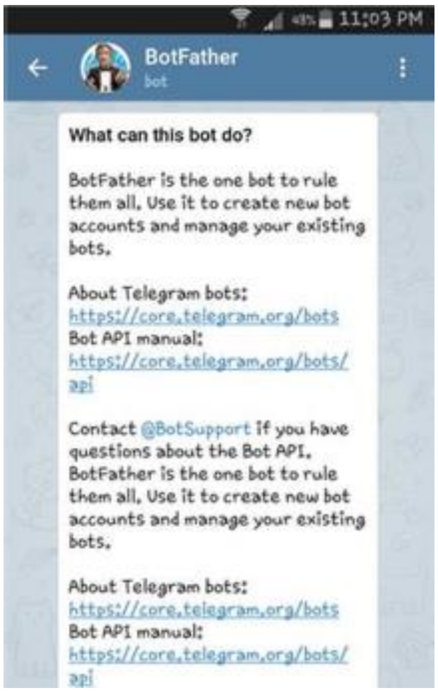
3: Click On Start Messaging Button

4: Enter your mobile number to register with telegram service.

5: Search for name "BotFather"



6: Click on "BotFather”



What can this bot do?

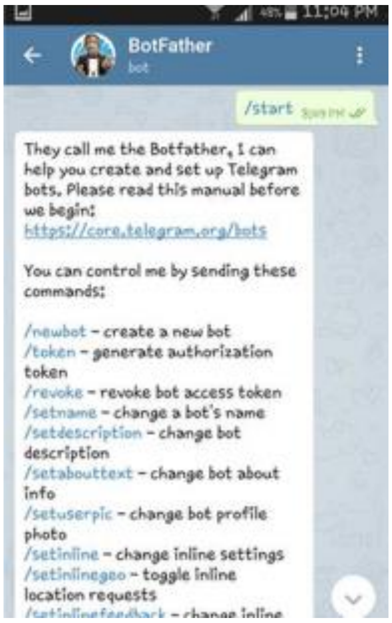
BotFather is the one bot to rule them all. Use it to create new bot accounts and manage your existing bots.

About Telegram bots:
https://core.telegram.org/bots
Bot API manual:
https://core.telegram.org/bots/api

Contact @BotSupport if you have questions about the Bot API.
BotFather is the one bot to rule them all. Use it to create new bot accounts and manage your existing bots.

About Telegram bots:
https://core.telegram.org/bots
Bot API manual:
https://core.telegram.org/bots/api

7: To Start "BotFather" type /start in message


BotFather
bot

/start

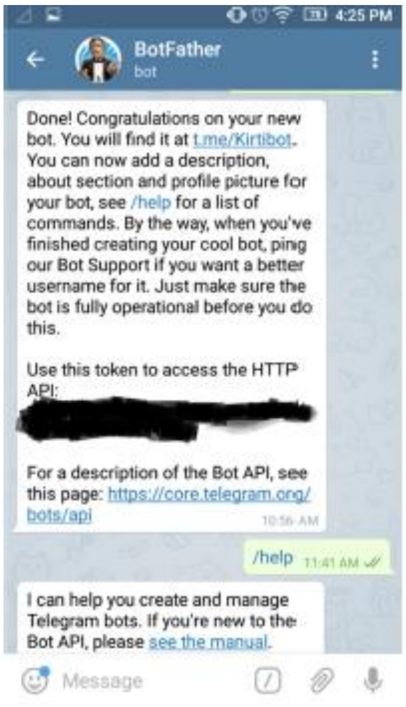
They call me the Botfather, I can help you create and set up Telegram bots. Please read this manual before we begin: https://core.telegram.org/bots

You can control me by sending these commands:

/newbot - create a new bot
/token - generate authorization token
/revoke - revoke bot access token
/setname - change a bot's name
/setdescription - change bot description
/setabouttext - change bot about info
/setuserpic - change bot profile photo
/setinline - change inline settings
/setinlinegeo - toggle inline location requests
/setinlinefeedback - change inline

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

9: Obtain access token


BotFather
bot

Done! Congratulations on your new bot. You will find it at t.me/Kirtibot. 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:
[redacted]

For a description of the Bot API, see this page: https://core.telegram.org/bots/api 10:56 AM

/help 11:41 AM

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

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

Set up On Raspberry Pi:-

1: Install "Python Package Index" and Telepot using :

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

2: Test your bot using python 2 IDLE and type:

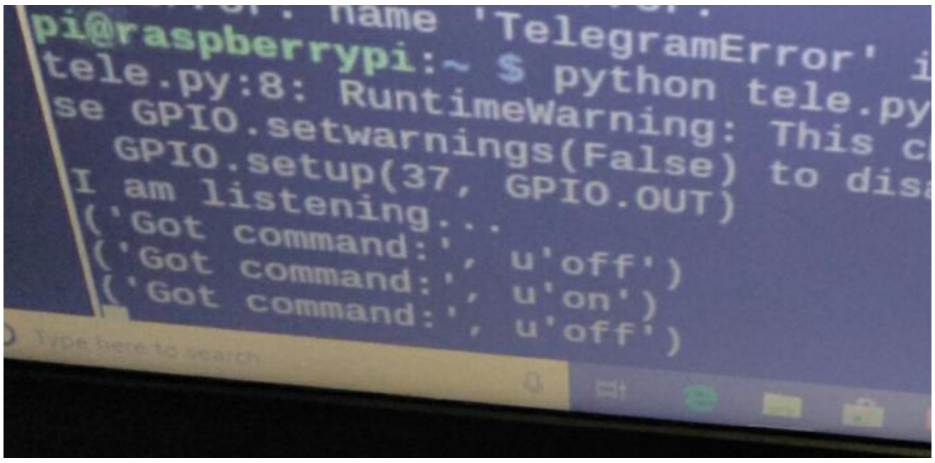
```
import telepot
bot = telepot.Bot('Bot Token')
bot.getMe()
```

If it prints your bot details means everything is correct.

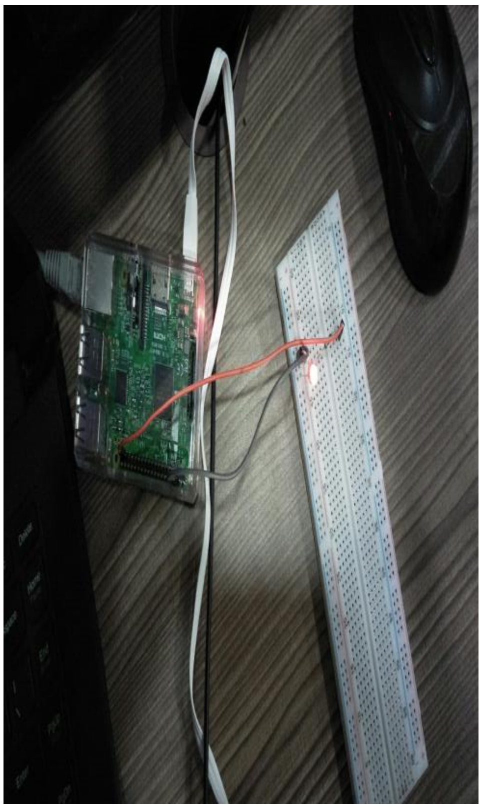
If not then token is wrong.

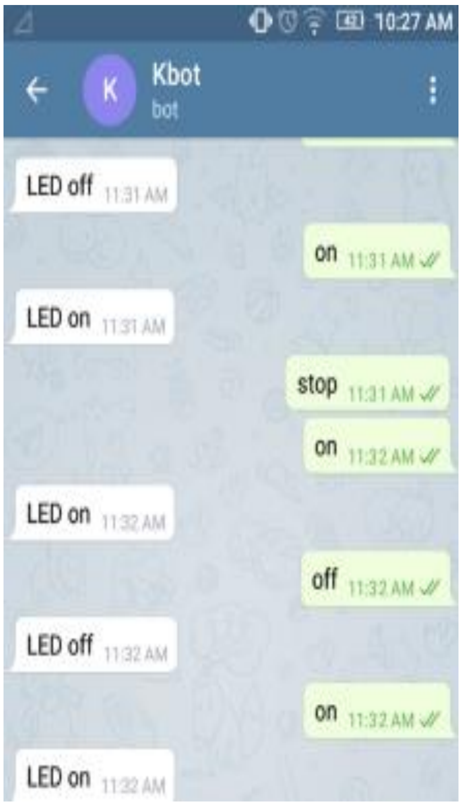
**Write the following code in Pyhton 2 IDLE and save it as 'mybot.py'**

```
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( ' ')
```



```
pi@raspberrypi:~ $ python tele.py
tele.py:8: RuntimeWarning: This c
se GPIO.setwarnings(False) to disa
  GPIO.setup(37, GPIO.OUT)
I am listening...
('Got command:', u'off')
('Got command:', u'on')
('Got command:', u'off')
```





```
LED off 11:31 AM
                      on 11:31 AM
LED on 11:31 AM
                      stop 11:31 AM
                      on 11:32 AM
LED on 11:32 AM
                      off 11:32 AM
LED off 11:32 AM
                      on 11:32 AM
LED on 11:32 AM
```

# Practical No:5

**Aim :** Capturing Image with Raspberry Pi and Pi Camera

**Hardware Required:**

Require Camera Module along with initial raspberry Pi set up

**Connect the Camera Module :**

1)Switch off the Raspberry Pi

2)Locate the Camera port, Connect the Camera Module.

3)Start up the Camera Pi

4)Open the Raspberry Pi Configuration tool from the main menu.

5)On the terminal , $sudo raspi-config.

6)Enable the camera by going to the Interfacing option .

**Write the following code:**

```
from time import sleep

from picamera import PiCamera


camera=PiCamera()

camera.resolution=(1280,720)          #Selecting resolution 1280x720 px

camera.start_preview()

sleep(2)

camera.capture('/home/pi/Pictures/newImage.jpg')

camera.stop_preview()
```

# Practical No :6

**Aim:** Interfacing Raspberry Pi with RFID.

**Hardware Required:**

1. EM-18 RFID Reader Module
2. USB-To-TTL converter
3. Few RFID Cards or RFID Tags
4. 5V external Power Supply for RFID Reader
5. Jumper Wires

**Connections:**

1: connect Tx pin of RFID module to Rx pin of USB-to-TTL converter.

2: connect GND (not the centre one) pin of RFID module to GND pin of USB-to-TTL converter.

3: connect positive pin (down side) of RFID module to External +5v pin of adapter.

4: connect negative pin (down side) of RFID module to External +5v pin of adapter.

5: Finally connect USB-to-TTL converter into USB port of raspberry pi.

**Software Required:**

1.Setting up Raspberry Pi for Serial Communication

```
$sudo raspi-config
```

2.In the “Interfacing Options”, select the “Serial” option.
Select “Yes” option.

3.Finish the process and Reboot the Raspberry Pi.

4.To check your USB port number use following command:

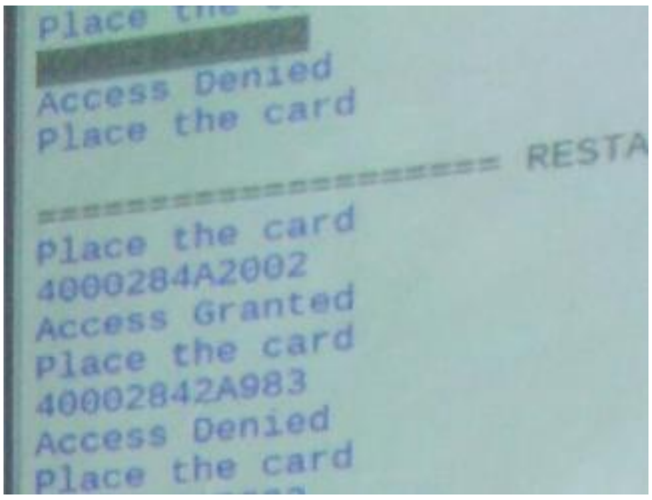
```
ls /dev/ttyUSB*
```

In following code it is ttyUSB0

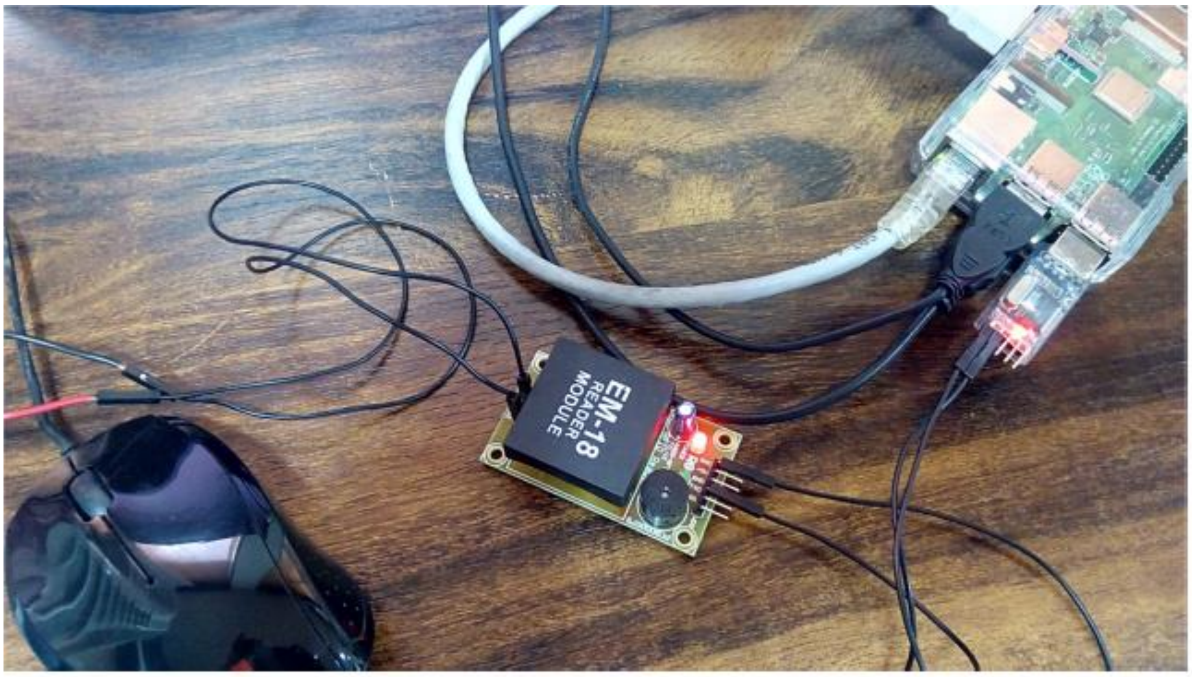
Write the following code in Python 2 IDLE and save it as ‘rfid.py’

```
import time
import serial
data=serial.Serial(port='/dev/ttyUSB0',baudrate=9600,parity=serial.PARITY_NONE,
stopbits=serial.STOPBITS_ONE,bytesize=serial.EIGHTBITS)
try:
        while 1:
                print "Place the card"
                x=data.read(12)
                print(x)
        if x=="1C00377ACC9D":
                print "Access Granted"
        else:
                print "Access Denied"
        except KeyboardInterrupt:
                data.close()
```

[Note: now place the RFID cards one by one over RFID module to check the output]



```
Place
[illegible]
Access Denied
Place the card
======================== RESTA
Place the card
4000284A2002
Access Granted
Place the card
40002842A983
Access Denied
Place the card
```


EM-18
READER
MODULE

## Practical No:7

**Aim:** Installing Windows 10 IoT Core on Raspberry Pi

**Hardware Required:**

1. Class 10 microSD card
2. Card Reader

**Software Configuration required:**

Do the following on your Windows machine or laptop.

1: open browser and open following link

https://developer.microsoft.com/en-us/windows/iot/Downloads.htm

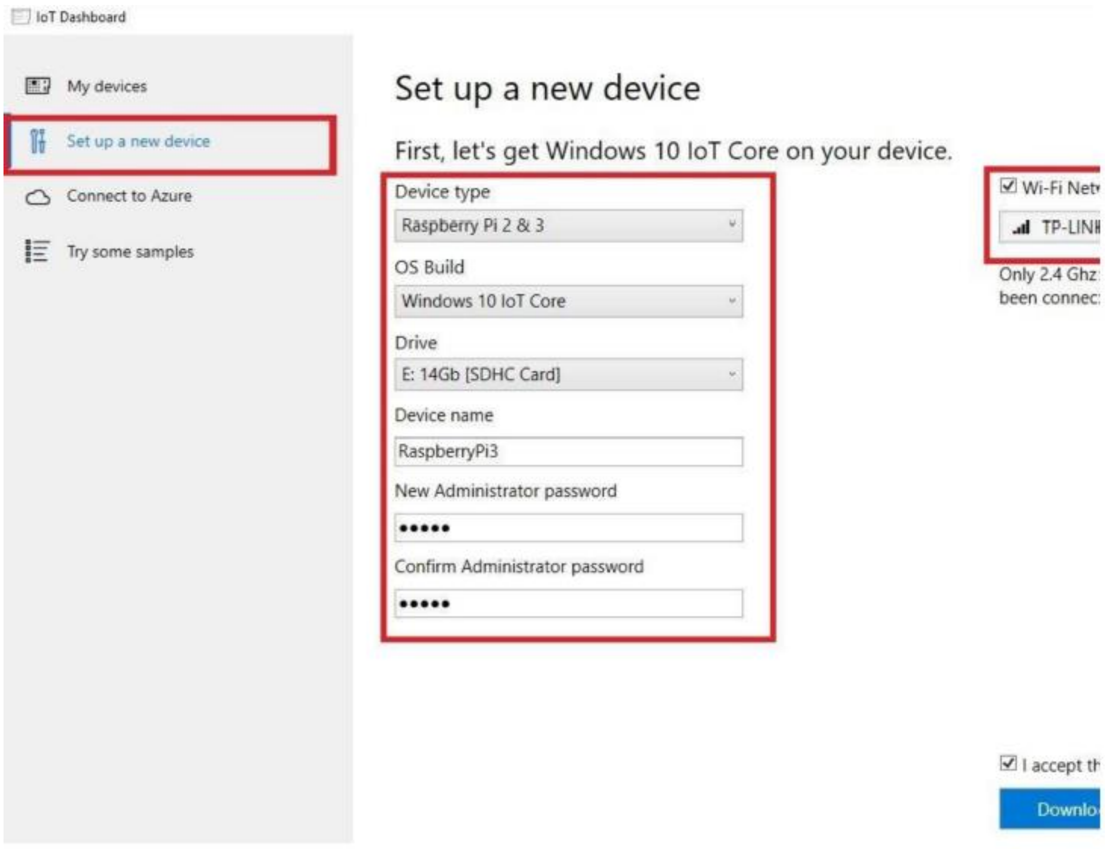
2: Click Get Windows 10 IoT Core Dashboard to download the necessary application

It will download setup.exe

3: Run that exe and install it and open it.

4: Select set up a new device from the sidebar

5: Select the options as shown in the image below. Make sure you select the correct drive for your microSD card and give your device a name and admin password.


IoT Dashboard

My devices

Set up a new device

Connect to Azure

Try some samples

Set up a new device

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

Device type

Raspberry Pi 2 & 3

OS Build

Windows 10 IoT Core

Drive

E: 14Gb [SDHC Card]

Device name

RaspberryPi3

New Administrator password

•••••

Confirm Administrator password

•••••

☑ Wi-Fi Netw

TP-LINK

Only 2.4 Ghz

been connec

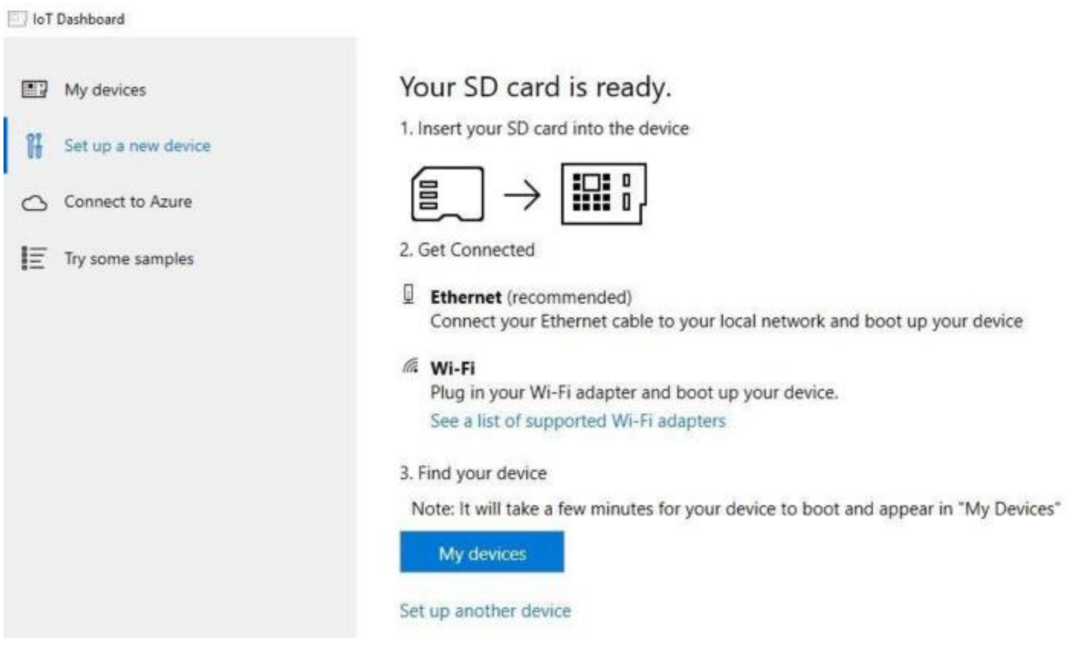
☑ I accept th

Downlo

6: Select the WiFi network connection you want your Raspberry Pi to connect to, if required. Only networks your PC connects to will be shown.

7: Click download and install.

The application will now download the necessary files from Microsoft and flash them to your microSD card.


IoT Dashboard

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"

8: 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.

9: Insert the microSD card into the Raspberry Pi and power it up.

**Practical No:8**

**Aim :** To implement GPS Module Interfacing With Raspberry Pi.

**Hardware Required :**

1.GPS module

2.USB to TTL converter

3.Connecting Wires

**Connections:**

1. Connect the VCC Pin of GPS Module to 3.3 V 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 $T_x$ Pin of GPS Module to $R_x$ Pin of USB to TTL Converter.
4. Connect the $R_x$ Pin of GPS Module to $T_x$ Pin of USB to TTL Converter.
5. Lastly connect the USB to TTL converter to USB port of Raspberry Pi.

**Write the following code in Python and save it as 'gps.py'**

```
''' GPS Interfacing with Raspberry Pi using Python'''

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

    global lat
    global longi


    nmea_time = []

    nmea_latitude = []

    nmea_longitude = []

    nmea_time = NMEA_buff[0]

    nmea_latitude = NMEA_buff[1]

    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 = nmea_latitude

    longi = nmea_longitude


    lat_in_degrees = convert_to_degrees(29)    #get latitude in degree decimal format

    long_in_degrees = convert_to_degrees(67) #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/ttyUSB0")            #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)



        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)
```

## Practical No 9

**Aim :** To implement Home  Automation using Raspberry Pi Web Server using Flask to control GPIO

**Components Required** :

- Raspberry Pi
- SD Card (minimum size 8Gb and class 10)
- Micro USB Power Supply
- Ethernet cable or WiFi dongle
- Breadboard
- 2x LEDs
- 2x 470Ω Resistors
- Jumper wires

**To install Flask :**

```
pi@raspberrypi ~ $ sudo apt-get update

pi@raspberrypi ~ $ sudo apt-get upgrade

pi@raspberrypi ~ $ sudo apt-get install python-pip python-flask

pi@raspberrypi ~ $ sudo pip install flask
```

**Creating the Python Script**

```
pi@raspberrypi ~ $ mkdir web-server
pi@raspberrypi ~ $ cd web-server
pi@raspberrypi:~/web-server $
```

Create a new file called *app.py*.

```
pi@raspberrypi:~/web-server $ nano app.py
```

```
import RPi.GPIO as GPIO
from flask import Flask, render_template, request
app = Flask(__name__)

GPIO.setmode(GPIO.BCM)

# Create a dictionary called pins to store the pin number, name, and pin
state:
pins = {
   23 : {'name' : 'GPIO 23', 'state' : GPIO.LOW},
   24 : {'name' : 'GPIO 24', 'state' : GPIO.LOW}
   }

# Set each pin as an output and make it low:
for pin in pins:
   GPIO.setup(pin, GPIO.OUT)
   GPIO.output(pin, GPIO.LOW)

@app.route("/")
def main():
   # For each pin, read the pin state and store it in the pins dictionary:
   for pin in pins:
      pins[pin]['state'] = GPIO.input(pin)
   # Put the pin dictionary into the template data dictionary:
   templateData = {
      'pins' : pins
      }
   # Pass the template data into the template main.html and return it to
the user
   return render_template('main.html', **templateData)

# The function below is executed when someone requests a URL with the pin
number and action in it:
@app.route("/<changePin>/<action>")
def action(changePin, action):
   # Convert the pin from the URL into an integer:
   changePin = int(changePin)
   # Get the device name for the pin being changed:
   deviceName = pins[changePin]['name']
   # If the action part of the URL is "on," execute the code indented
below:
   if action == "on":
      # Set the pin high:
      GPIO.output(changePin, GPIO.HIGH)
      # Save the status message to be passed into the template:
      message = "Turned " + deviceName + " on."
   if action == "off":
      GPIO.output(changePin, GPIO.LOW)
      message = "Turned " + deviceName + " off."

   # For each pin, read the pin state and store it in the pins dictionary:
   for pin in pins:
      pins[pin]['state'] = GPIO.input(pin)

   # Along with the pin dictionary, put the message into the template data
dictionary:
   templateData = {
      'pins' : pins
   }

   return render_template('main.html', **templateData)
```

```
if __name__ == "__main__":
   app.run(host='0.0.0.0', port=80, debug=True)
```

## Creating the HTML File

Create a new folder called templates:

```
pi@raspberrypi:~/web-server $ mkdir templates

pi@raspberrypi:~/web-server $ cd templates

pi@raspberrypi:~/web-server/templates $
```

Create a new file called *main.html*.

```
pi@raspberrypi:~/web-server/templates $ nano main.html
```

```
<!DOCTYPE html>
<head>
   <title>RPi Web Server</title>
   <!-- Latest compiled and minified CSS -->
   <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css
" integrity="sha384-
1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
crossorigin="anonymous">
   <!-- Optional theme -->
   <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css" integrity="sha384-
fLW2N01lMqjakBkx3l/M9EahuwpSfeNvV63J5ezn3uZzapT0u7EYsXMjQV+0En5r"
crossorigin="anonymous">
   <!-- Latest compiled and minified JavaScript -->
   <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"
integrity="sha384-
0mSbJDEHialfmuBBQP6A4Qrprq5OVfW37PRR3j5ELqxss1yVqOtnepnHVP9aJ7xS"
crossorigin="anonymous"></script>
</head>

<body>
   <h1>RPi Web Server</h1>
   {% for pin in pins %}
   <h2>{{ pins[pin].name }}
   {% if pins[pin].state == true %}
      is currently <strong>on</strong></h2><div class="row"><div
class="col-md-2">
```

```
      <a href="/{{pin}}/off" class="btn btn-block btn-lg btn-default"
role="button">Turn off</a></div></div>
   {% else %}
      is currently <strong>off</strong></h2><div class="row"><div
class="col-md-2">
      <a href="/{{pin}}/on" class="btn btn-block btn-lg btn-primary"
role="button">Turn on</a></div></div>
   {% endif %}
   {% endfor %}
</body>
</html>
```

## Launching the Web Server

To launch your Raspberry Pi web server move to the folder that contains the file *app.py*:

```
pi@raspberrypi:~/web-server/templates $ cd ..
```

Then run the following command:

```
pi@raspberrypi:~/web-server $ sudo python app.py
```