1.LED Blinking

Aim of experiment:

Blinking of LED with Raspberry Pi.

Requirements:

- **Hardware Requirements**

Sr. No	Component	QTY
1	LED Module	1
2	Jumper Wire	2
3	Raspberry Kit	1

Software Requirements

-List of Libraries to be used for a particular practical

import RPi.GPIO as GPIO

from time import sleep

Connection:

-In the raspberry Pi development board kit, there is an LED module.

-The LED Module has 7 male pins at the right hand side and 2male pins at the RHS.

-The LHS is the positive terminal and RHS is negative terminal.

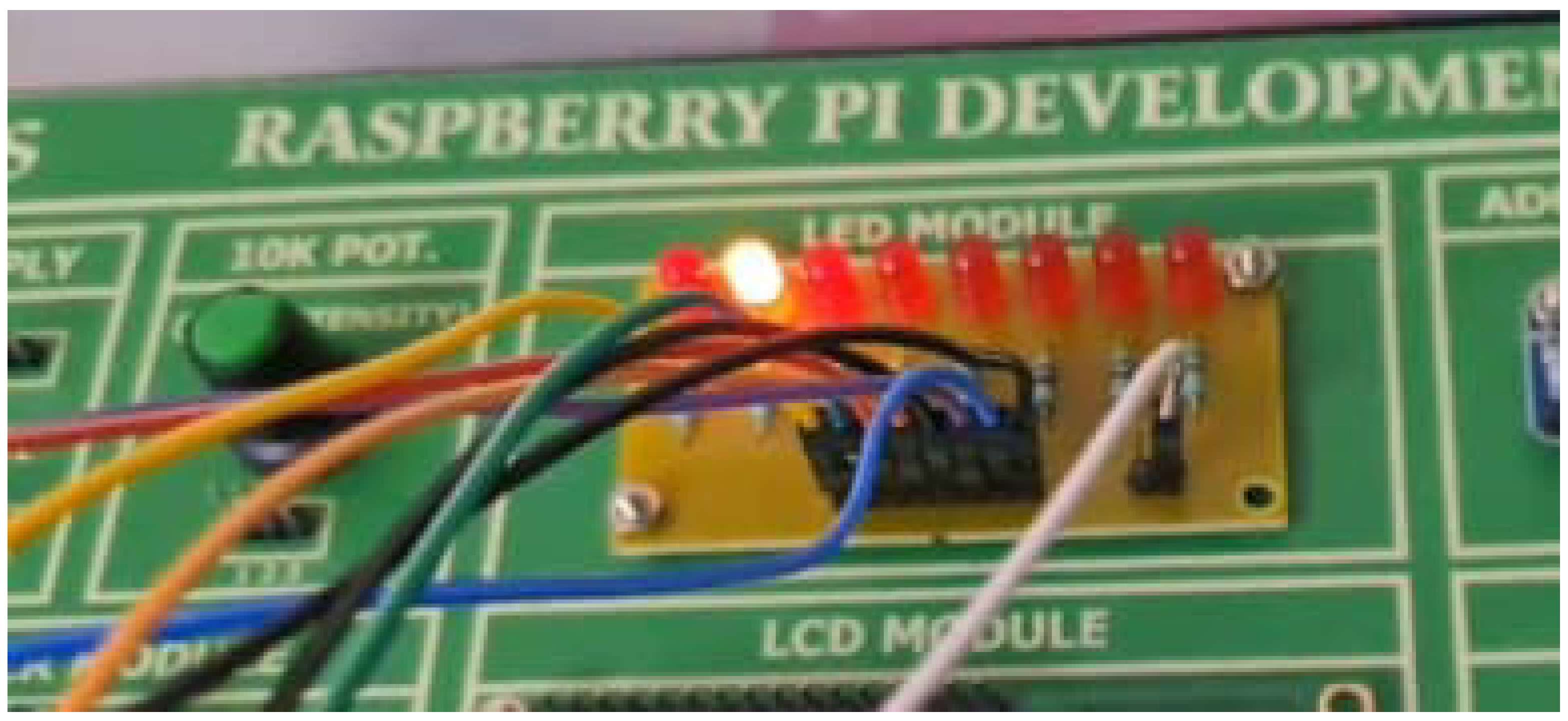
-Connect the raspberry Pi to positive terminal and negative terminal

Code:

```
import RPi.GPIO as GPIO

from time import sleep

GPIO.setmode(GPIO.BOARD)
```

```
r= [3,15,8,11,13,26,29,22]
GPIO.setup(r, GPIO.OUT)


while True:
 for i in r:
    GPIO.output(i, GPIO.HIGH)
    sleep (0.1)
    GPIO.output(i, GPIO.LOW)
    sleep (0.1)
```

Output:



2. Displaying different LED patterns with Raspberry Pi.

Aim of experiment:

Different LED Patterns with Raspberry Pi

Requirements

- **Hardware Requirements**

Sr. No	Component	QTY
1	LED	4
2	Jumper Wires	5
3	Raspberry Kit	1

Software Requirements

-List of Libraries to be used for a particular practical

import tm1637

import time

from datetime import datetime

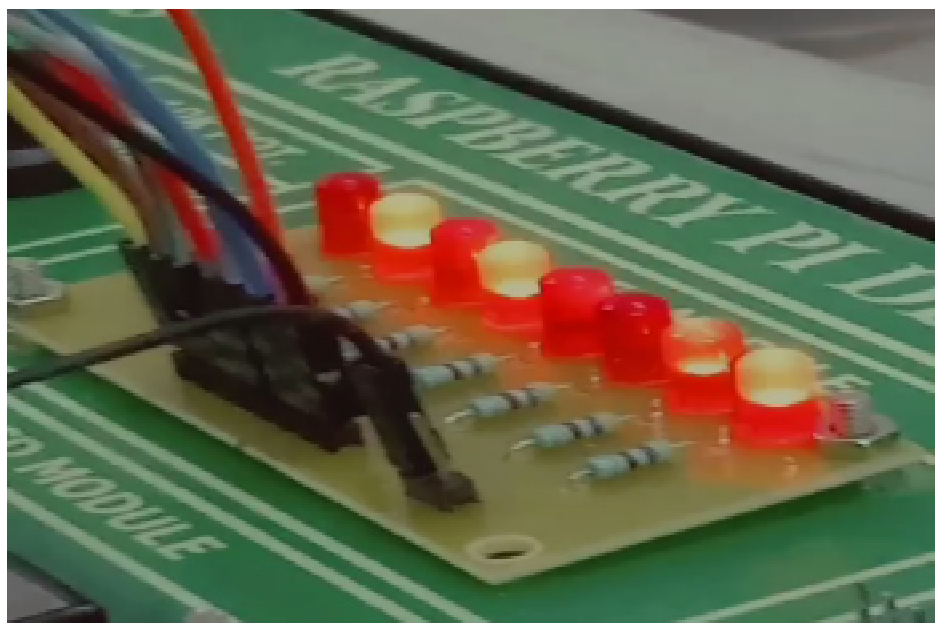
Connection:
-In the raspberry Pi development board kit, there is an LED module.
-The LED Module has 7 male pins at the right hand side and 2male pins at the RHS.
-The LHS is the positive terminal and RHS is negative terminal.
-Connect the raspberry Pi to positive terminal and negative terminal

Code:

```
import tm1637

import time

from datetime import datetime

tm=tm1637.TM1637(clk=18, dio=17)
```

```
clear= [0,0,0,0]
tm.write(clear)
time.sleep(10)
#print("hii")
s='SAHYOG'
tm.scroll(s, delay=250)
time.sleep(0)
tm.write(clear)
time.sleep(0)
while True:
    now1=datetime.now()
    hh=int(datetime.strftime(now1,'%H'))
    mm=int(datetime.strftime(now1,'%M'))
    tm.numbers(hh,mm,colon=True)
    time.sleep(1)
    tm.write(clear)
print("hii")
```

Output:



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

Aim of experiment:

Display Time Over 4 Digit 7-Segment Display Using Raspberry Pi.

Requirements

- **Hardware Requirements**

Sr. No	Component	*QTY*
1	Seven segment display	1
2	Jumper Wires	4
3	Raspberry Kit	1

Software Requirements

-List of Libraries to be used for a particular practical

TM1637 library

Connection:

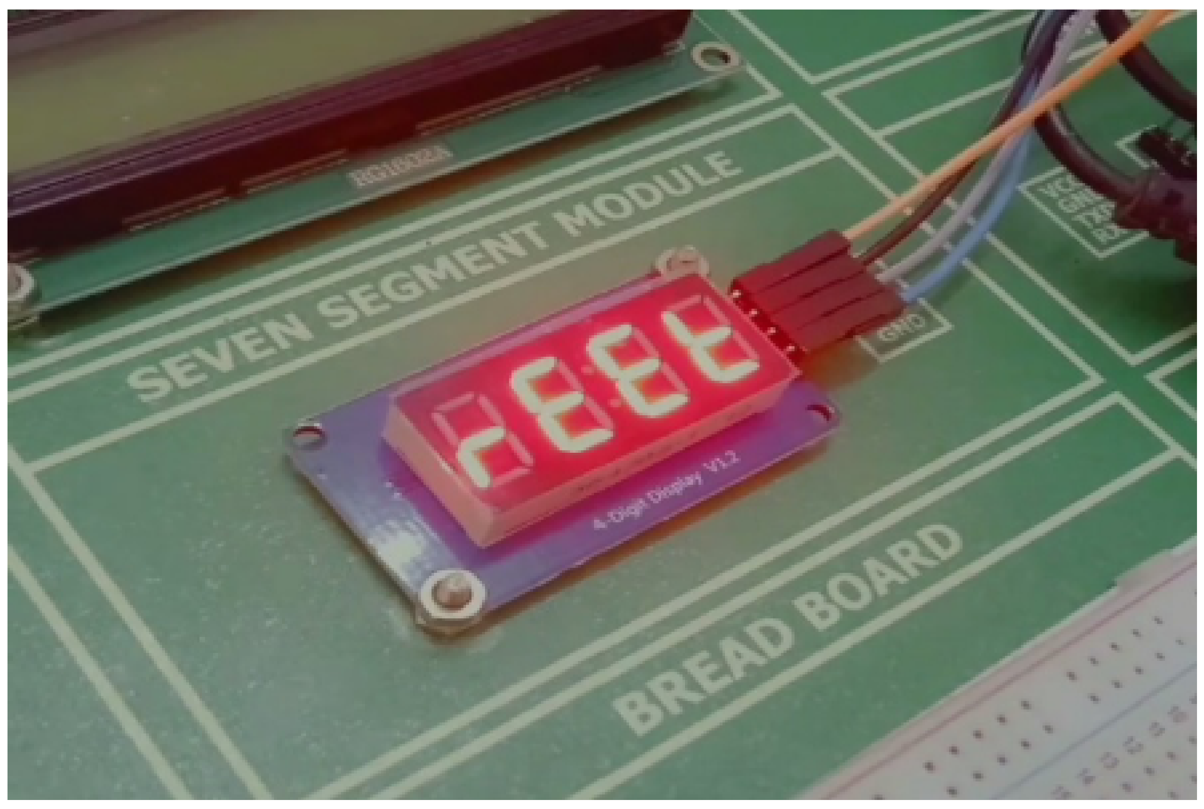
-In the raspberry Pi development board kit, there is a Seven Segment module.
-The Seven Segment Module has 4 male pins.
-The male pins contain GND, VCC, DI0, CLK.
-Later they are connected to the Raspberry Pi Physical pin.
-GND is connected to 14 number of RPI Physical pin
-VCC is connected to 4 number of RPI Physical pin
-DI0 is connected to 18 number of RPI Physical pin
-CLK is connected to 16 number of RPI Physical pin

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

Code:

```
import tm1637

import time

from datetime import datetime

tm=tm1637.TM1637(clk=18, dio=17)

clear= [0,0,0,0]

tm.write(clear)

time.sleep(1)

s='This is pretty cool'

tm.scroll(s, delay=250)

time.sleep(0)

tm.write(clear)

time.sleep(0)

while True:

    now=datetime.now()

    hh=int(datetime.strftime(now,'%H'))

    mm=int(datetime.strftime(now,'%M'))

    tm.numbers(hh,mm,colon=True)

   ##  time.sleep(1000)

    tm.write(clear)
```

Output:



4.Raspberry Pi Based Oscilloscope

Aim of experiment:

Raspberry pi Based Oscilloscope.

Requirements

- **Hardware Requirements**

Sr. No	Component	*QTY*
1	Oscilloscope Module	1
2	Jumper Wires	4
3	Raspberry Kit	1

Software Requirements

-List of Libraries to be used for a particular practical

Adafruit_ADS1x15.ADS1115 library
Python Matplotlib library
Drwanow library

Connection:

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

-In the raspberry Pi development board kit, there is an Oscilloscope module.
-The Oscilloscope Module has 4 male pins.
-The male pins contain GND, VDD, SCL, SDA.
-Later they are connected to the Raspberry Pi Physical pin.
-GND is connected to 9 number of RPI Physical pin
-VDDis connected to 17 number of RPI Physical pin

-SCL is connected to 5 number of RPI Physical pin
-SDA is connected to 3 number of RPI Physical pin

Code:

```
import time

import matplotlib.pyplot as plt

import drawnow
from drawnow import drawnow,figure
import Adafruit_ADS1x15
adc = Adafruit_ADS1x15.ADS1115()
GAIN = 1
value=[]
cnt = 0
plt.ion()
adc.start_adc(0,gain=GAIN)

def makeFig():
        plt.ylim(-9000,19000)
            plt.title("Oscilloscope")
            plt.grid(True)
            plt.ylabel('ADC outputs')
            plt.plot(value,'ro-',label = 'lux')
            plt.legend(loc ='lower right')

         while True:
            values = adc.get_last_result()
            print('Channel 0:{0}'.format(value))
            time.sleep(0.5)
            value.append(int(values))
            drawnow(makeFig)
            cnt=cnt+1
            if(cnt>50):
               value.pop(0)
```

Special procedure:

Step 1: Enable I2C Run the following command from the terminal: $ sudo raspi-config In the configuration panel, select "Interface Options," then select "I2C" and enable it.

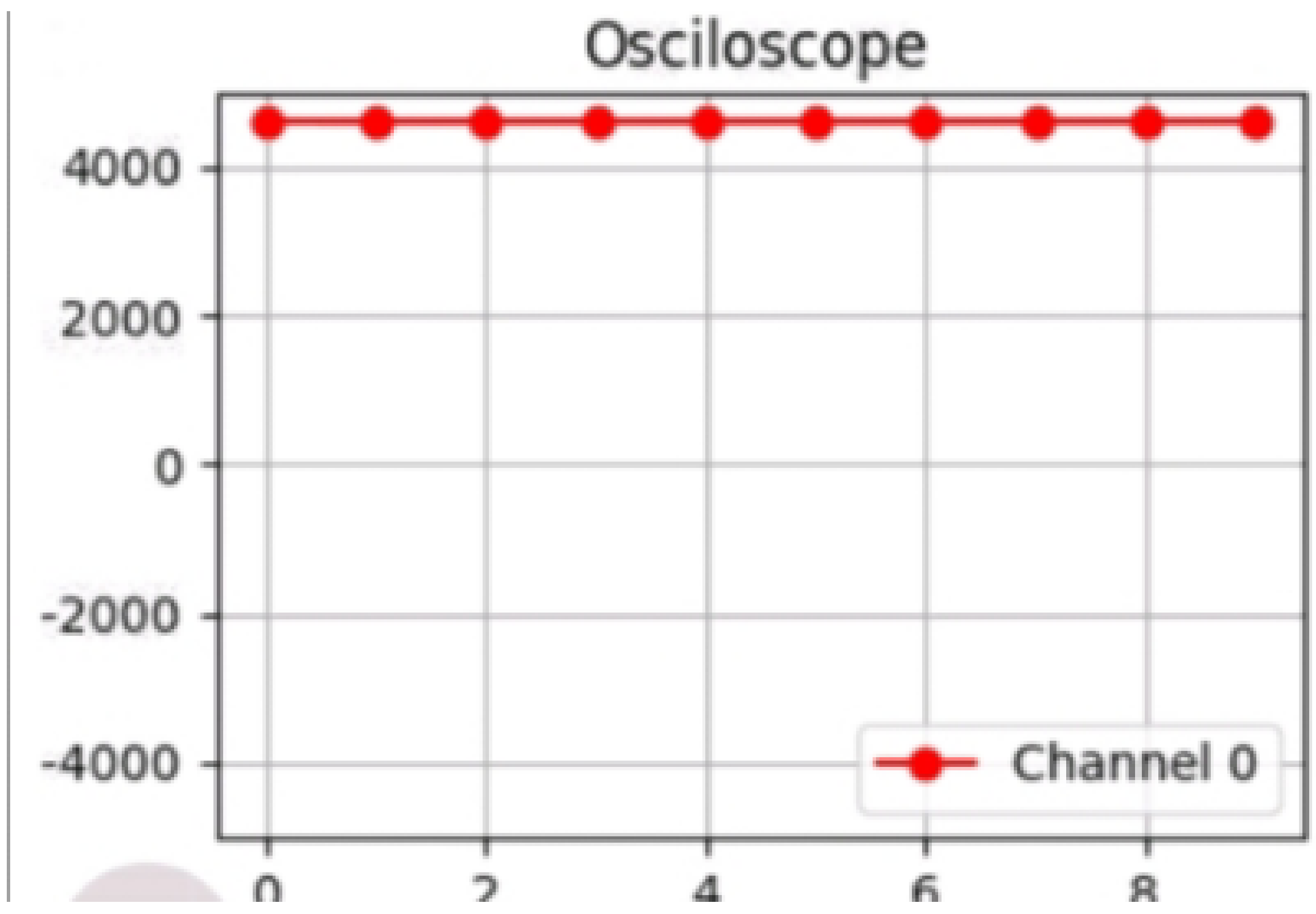
Step 2: Update and install dependencies $ sudo apt-get update $ sudo apt-get upgrade $ cd ~ $ sudo apt-get install build-essential python-dev python-smbus git $ git clone https://github.com/adafruit/Adafruit_python_AdS1x15.git $ cd Adafruit_Python_ADS1x15 $ sudo python setup.py install

Step 3: Test the library and I2C communication $ cd examples Run the sampletest.py example, which displays the value of the four channels on the ADC in a tabular form. $ python simpletest.py

Step 4: Install matplotlib for data visualization $ sudo apt-get install python-matplotlib $ pip3 install matplotlib

Step 5: Install drawnow for live updates to the data plot $ sudo apt-get install python-pip $ sudo pip3 install drawnow To Check I2C devices $ i2cdetect -y 1

Output:



5.Interfacing Raspberry Pi with RFID.

Aim of experiment:

Interfacing Raspberry Pi with RFID (Read Card)

Requirements

- **Hardware Requirements**

Sr. No	Component	*QTY*
1	RFID Module	1
2	RFID Cards	1
3	Jumper Wires	4
4	Raspberry Kit	1

Software Requirements

-List of Libraries to be used for a particular practical

import RPi.GPIO as GPIO

from mfrc522 import SimpleMFRC522

Connection:

RFID	RPi Physical Pin	GPIO Number
3V3	Pin 1	3.3v
GND	Pin 9	GND
SCK	Pin 23	SCLK
SDA	Pin 24	GPIO 8
MISO	Pin 21	MISO
MOSI	Pin 19	MOSI
RST	Pin 22	GPIO 25

Code:

```
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522
GPIO.setwarnings(False)
reader = SimpleMFRC522()
while True:
        try:
            id, text = reader.read()
            print(id)
          print(text)
       finally:
           GPIO.cleanup()
```

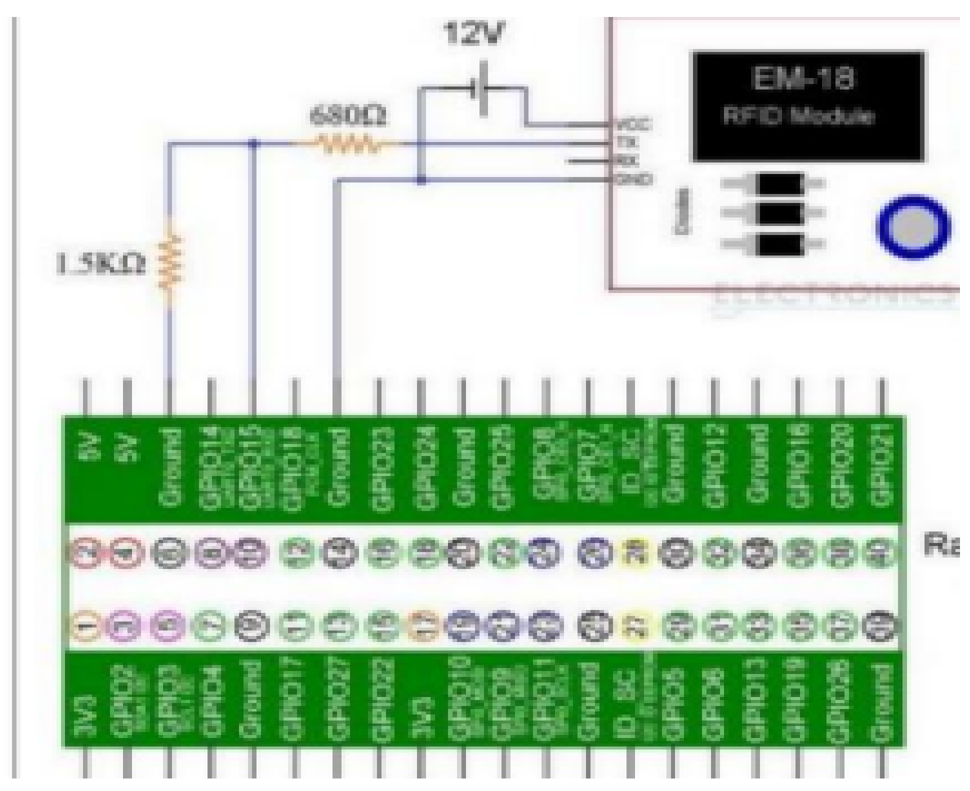
Special procedures other than connection:

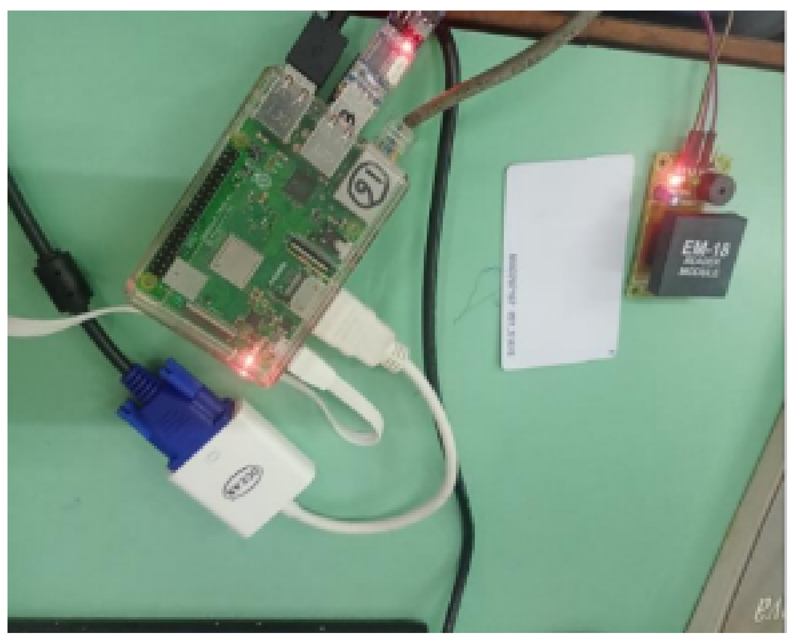
1) $sudo raspi-config.

2) Select the "Serial" option To access the login shell over serial communication, Select "No" option

3) $dmesg | grep tty.

Output:




EM-18
READER
MODULE

6.CAMERA:

Aim of experiment:

Visitor Monitoring with Raspberry Pi Camera

Requirements

- **Hardware Requirements**

Sr. No	Component	*QTY*
1	Camera Module	1
2	Raspberry Kit	1

Software Requirements

-List of Libraries to be used for a particular practical

from time import sleep
from picamera import PiCamera

Connection:
-To connect the Pi Camera:
Insert the Ribbon cable of Pi Camera into camera slot, slightly pull up the tabs of the connector at RPi board and insert the Ribbon cable into the slot, then gently push down the tabs again to fix the ribbon cable

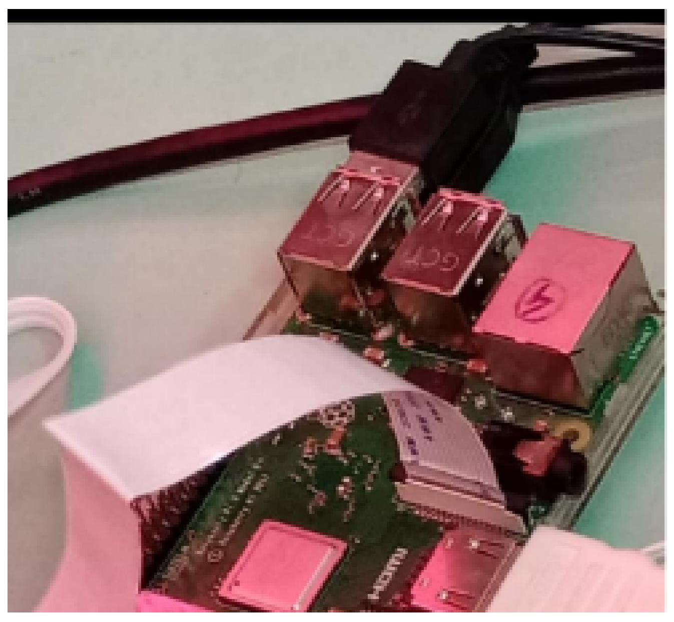
Code:

```
from time import sleep
from picamera import PiCamera

camera = PiCamera()
camera.resolution = (1280, 720)
camera.start_preview()
sleep(0.5)
camera.capture('/home/sahyog123/Pictures/new1.jpg')
camera.stop_preview()
```

Special procedures other than connection:

-sudo apt update
-sudo apt upgrade
-pip install picamera
-pip3 install picamera
-sudo apt-get install python-picamera.
-sudo apt-get install python3-picamera.
-sudoraspi-config
-Select Enable camera and Enable it sudo reboot
-To verify connection:
    raspistill -o test.

Output:



7.VIDEO:

Aim of experiment:

Visitor Monitoring with Raspberry Pi Video

Requirements

- **Hardware Requirements**

Sr. No	Component	*QTY*
1	Camera Module	1
2	Raspberry Kit	1

Software Requirements

-List of Libraries to be used for a particular practical

from time import sleep
from picamera import PiCamera

Connection:

-To connect the Pi Camera:
Insert the Ribbon cable of Pi Camera into camera slot, slightly pull up the tabs of the connector at RPi board and insert the Ribbon cable into the slot, then gently push down the tabs again to fix the ribbon cable

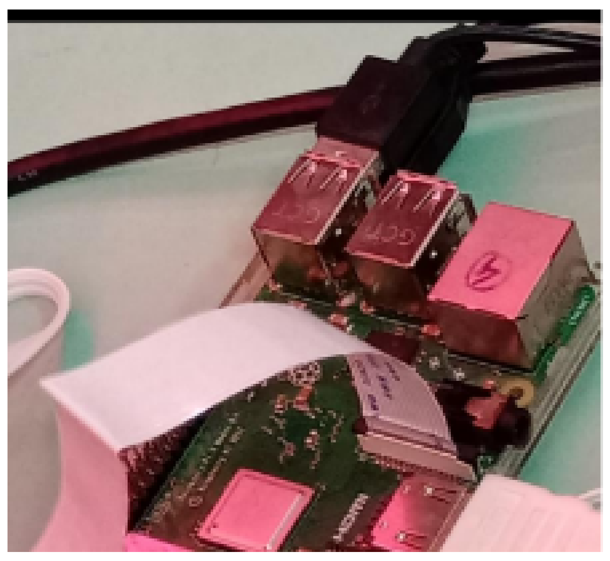
Code:

```
from time import sleep
from picamera import PiCamera

camera = PiCamera()
camera. resolution = (1280, 720)
camera.start_preview()
camera.start_recording('/home/sahyog123/videos/video.h265')
sleep(10)
camera.stop_recording()
camera.stop_preview()
```

Special procedures other than connection

-sudo apt update
-sudo apt upgrade
-pip install picamera
-pip3 install picamera
-sudo apt-get install python-picamera.
-sudo apt-get install python3-picamera.
-sudoraspi-config
-Select Enable camera and Enable it sudo reboot
-To verify connection:
    raspistill -o test.

Output:



8.Fingerprint Sensor interfacing with Raspberry Pi

Aim of experiment:

Fingerprint with Raspberry Pi Video

Requirements

- **Hardware Requirements**

Sr. No	Component	*QTY*
1	Fingerprint Module	1
2	Jumper Wires	4
3	Raspberry Kit	1

Software Requirements

-List of Libraries to be used for a particular practical

```
import time
import board
import busio
from digitalio import DigitalInOut, Direction
import adafruit_fingerprint
```

Connection:

Fingerprint Module	RPi Physical Pin	GPIO Number
VCC	Pin 2	5V
GND	Pin 9	GND
TxD	Pin 15	Rx
RxD	Pin 14	Tx

Code:

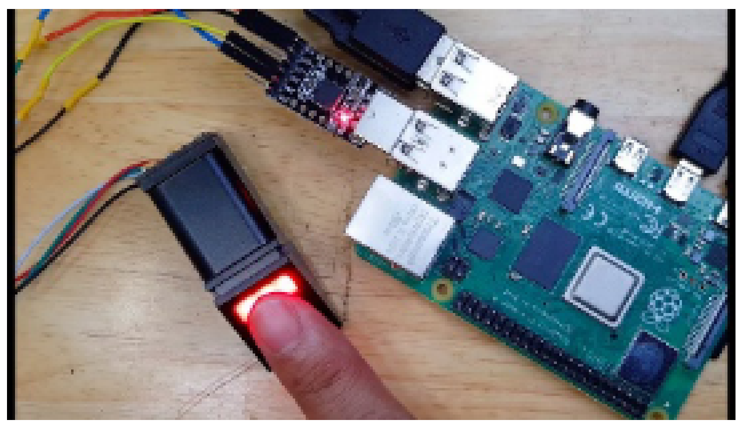
```
import time
import board
import busio
from digitalio import DigitalInOut, Direction
import adafruit_fingerprint

led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

# uart = busio.UART(board.TX, board.RX, baudrate=57600)

# If using with a computer such as Linux/RaspberryPi, Mac, Windows with
USB/serial converter:
import serial
# uart = serial.Serial("/dev/ttyUSB0", baudrate=57600, timeout=1)

# If using with Linux/Raspberry Pi and hardware UART:
# import serial
uart = serial.Serial("/dev/ttyS0", baudrate=57600, timeout=1)

finger = adafruit_fingerprint.Adafruit_Fingerprint(uart)


def get_fingerprint():
    """Get a finger print image, template it, and see if it matches!"""
    print("Waiting for image...")
    while finger.get_image() != adafruit_fingerprint.OK:
        pass
    print("Templating...")
    if finger.image_2_tz(1) != adafruit_fingerprint.OK:
        return False
    print("Searching...")
    if finger.finger_search() != adafruit_fingerprint.OK:
        return False
    return True
```

```
# pylint: disable=too-many-branches
def get_fingerprint_detail():
    """Get a finger print image, template it, and see if it matches!
    This time, print out each error instead of just returning on failure"""
    print("Getting image...", end="")
    i = finger.get_image()
    if i == adafruit_fingerprint.OK:
        print("Image taken")
    else:
        if i == adafruit_fingerprint.NOFINGER:
            print("No finger detected")
        elif i == adafruit_fingerprint.IMAGEFAIL:
            print("Imaging error")
        else:
            print("Other error")
        return False

    print("Templating...", end="")
    i = finger.image_2_tz(1)
    if i == adafruit_fingerprint.OK:
        print("Templated")
    else:
        if i == adafruit_fingerprint.IMAGEMESS:
            print("Image too messy")
        elif i == adafruit_fingerprint.FEATUREFAIL:
            print("Could not identify features")
        elif i == adafruit_fingerprint.INVALIDIMAGE:
            print("Image invalid")
        else:
            print("Other error")
        return False

    print("Searching...", end="")
    i = finger.finger_fast_search()
    # pylint: disable=no-else-return
    # This block needs to be refactored when it can be tested.
    if i == adafruit_fingerprint.OK:
        print("Found fingerprint!")
        return True
    else:
        if i == adafruit_fingerprint.NOTFOUND:
            print("No match found")
        else:
            print("Other error")
        return False


# pylint: disable=too-many-statements
def enroll_finger(location):
    """Take a 2 finger images and template it, then store in 'location'"""
    for fingerimg in range(1, 3):
        if fingerimg == 1:
            print("Place finger on sensor...", end="")
        else:
            print("Place same finger again...", end="")

        while True:
            i = finger.get_image()
            if i == adafruit_fingerprint.OK:
                print("Image taken")
                break
            if i == adafruit_fingerprint.NOFINGER:
                print(".", end="")
            elif i == adafruit_fingerprint.IMAGEFAIL:
                print("Imaging error")
                return False
            else:
                print("Other error")
                return False

        print("Templating...", end="")
        i = finger.image_2_tz(fingerimg)
        if i == adafruit_fingerprint.OK:
            print("Templated")
        else:
            if i == adafruit_fingerprint.IMAGEMESS:
                print("Image too messy")
            elif i == adafruit_fingerprint.FEATUREFAIL:
```

```
            print("Could not identify features")
        elif i == adafruit_fingerprint.INVALIDIMAGE:
            print("Image invalid")
        else:
            print("Other error")
        return False

    if fingering == 1:
        print("Remove finger")
        time.sleep(1)
        while i != adafruit_fingerprint.NOFINGER:
            i = finger.get_image()

print("Creating model...", end="")
i = finger.create_model()
if i == adafruit_fingerprint.OK:
    print("Created")
else:
    if i == adafruit_fingerprint.ENROLLMISMATCH:
        print("Prints did not match")
    else:
        print("Other error")
    return False

print("Storing model #%d..." % location, end="")
i = finger.store_model(location)
if i == adafruit_fingerprint.OK:
    print("Stored")
else:
    if i == adafruit_fingerprint.BADLOCATION:
        print("Bad storage location")
    elif i == adafruit_fingerprint.FLASHERR:
        print("Flash storage error")
    else:
        print("Other error")
    return False

return True
```

```
def get_num():
    """Use input() to get a valid number from 1 to 127. Retry till success!"""
    i = 0
    while (i > 127) or (i < 1):
        try:
            i = int(input("Enter ID # from 1-127: "))
        except ValueError:
            pass
    return i


while True:
    print("----------------")
    if finger.read_templates() != adafruit_fingerprint.OK:
        raise RuntimeError("Failed to read templates")
    print ("Fingerprint templates:", finger.templates)
    print("e) enroll print")
    print("f) find print")
    print("d) delete print")
    print("----------------")
    c = input("> ")

    if c == "e":
        enroll_finger(get_num())
    if c == "f":
        if get_fingerprint():
            print("Detected #", finger.finger_id, "with confidence", finger.confidence)
        else:
            print("Finger not found")
    if c == "d":
        if finger.delete_model(get_num()) == adafruit_fingerprint.OK:
            print("Deleted!")
        else:
            print("Failed to delete")
```

Output:



9.Telegram Bot:

Aim of experiment:

Controlling Raspberry Pi Using Telegram.

Requirements

- **Hardware Requirements**

Sr. No	Component	*QTY*
1	LED	1
2	Jumper Wires	4
3	Raspberry Kit	1
4	Phone with telegram	1

List of Libraries to be used for a particular practical

```
import RPi.GPIO as GPIO
import teleport
from telepot.loop import MessageLoop
import time
```

Connection:

Install telegram bot on raspberry pi :-

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

Steps to be follow to access LED through the Telegram:

I. Install Telegram app on smartphone from playstore.

II. To open Telegram, request the bot father to create a new bot.

III. Search “ BotFather” and click on it.

IV. Request to start a service using /start.

V. Create a new bot using /newbot.

VI. Provide the name for new bot that we want to create and give same name followed by _bot postfix (e.g. tyit_bot).

In exam Add Roll no. as botname e,g (xyz_bot)

VII. After this process the BotFather wil give a Token for access.

VIII. Now search for the bot that we have created.

IX. Now give the command to LEDs as per your requirements

Code:

```
# Import necessary libraries
 import RPi.GPIO as GPIO # Import the RPi.GPIO library to control GPIO pins import
telepot
# Import the telepot library for Telegram bot functionality from telepot.loop import
MessageLoop
# Import MessageLoop to handle Telegram messages
import time

# Set up GPIO mode and suppress warnings
 GPIO.setmode(GPIO.BOARD)
 GPIO.setwarnings(False)

# Define GPIO pin numbers for the LEDs
blue = 7
red = 11
yellow = 13
# Set up GPIO pins as outputs and turn them off
GPIO.setup(blue, GPIO.OUT)
GPIO.output(blue, False)
GPIO.setup(red, GPIO.OUT)
GPIO.output(red, False)
GPIO.setup(yellow, GPIO.OUT)
GPIO.output(yellow, False)

# Define a function to handle incoming Telegram messages
def action(msg):
    chat_id = msg['chat']['id']
    command = msg['text']
  print('Received: %s' % command)
message = ''

# Check for 'on' command and control LEDs accordingly
if 'on' in command:
```

```
          inmessage = 'on'
          if 'blue' in command:
          message += " blue"
          GPIO.output(blue, 1)
          if 'yellow' in command:
          message += " yellow"
          GPIO.output(yellow, 1)
          if 'red' in command:
          message += " red"
          GPIO.output(red, 1)
          if 'all' in command:
          message += " all"
          GPIO.output(blue, 1)
          GPIO.output(yellow, 1)
          GPIO.output(red, 1)
          message += " lights"
          telegram_bot.sendMessage(chat_id, message)

          # Check for 'off' command and turn off LEDs accordingly
           if 'off' in command:
              message = 'off'
              if 'blue' in command:
              message += " blue"
              GPIO.output(blue, 0)
              if 'yellow' in command:
              message += " yellow"
              GPIO.output(yellow, 0)
              if 'red' in command:
              message += " red"
              GPIO.output(red, 0) if 'all' in command: message += " all" GPIO.output(blue, 0)
          GPIO.output(yellow, 0)
              GPIO.output(red, 0) message += " lights" telegram_bot.sendMessage(chat_id,
          message)

# Initialize the Telegram bot with your token
 telegram_bot = telepot.Bot('Your-Telegram-Bot-Token-Goes-Here')
print('Up and Running… ')
 # Start the MessageLoop to listen for incoming Telegram messages
MessageLoop(telegram_bot, action).run_as_thread()
```

```
# Keep the script running
while True:
	time.sleep(10) # Sleep for 10 seconds to avoid excessive CPU
```

10.IoT based Web Controlled Home Automation using Raspberry Pi

Aim of experiment:

Controlling AC appliances with the click of buttons on a webpage using internet.

Requirements:

Hardware requirements:

Sr. No	**Component**	***QTY***
1	AC Lamp/Bulb	1
2	Jumper Wires	4
3	Raspberry Kit	1
4	Relay module	1

Software Requirements

-List of Libraries to be used for a particular practical

WEBIOPi framework

Connection:

Relay Board Pins	Function	RPI Physical Pin	Raspberry Function
5V	5V Power	4	5V Power
I/P	Data In	7	GPIO 4
GND	Ground	6	GND

LED Terminal	RPI Physical pin	LED Terminal	RPI Physical pin
Positive	37	Negative Terminal	39

Open browser and connect to http://Pi's IP address :8000

The system prompt will ask for username and password username: webiopi

password: raspberry

After login click on GPIO Header. Now click on Pin 37 to turn on and off the AC Bulb.

Special procedures other than connection:

1.wget http://sourceforge.net/projects/webiopi/files/WebIOPi-0.7.1.tar.gz

2.Extract file: tar xvzf WebIOPi-0.7.1.tar.gz

cd WebIOPi-0.7.1/

3.Install patch:

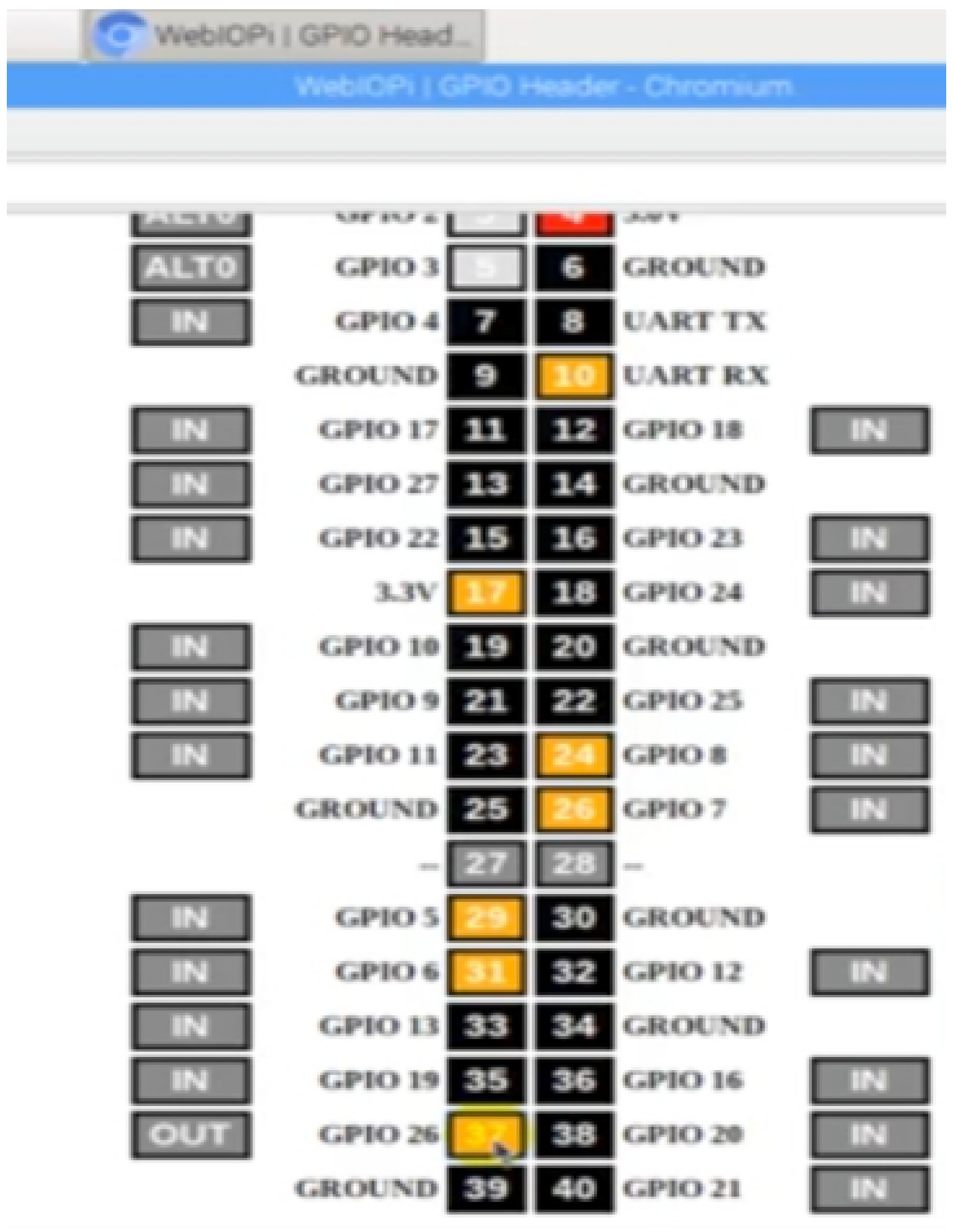
wget https://raw.githubusercontent.com/doublebind/raspi/master/webiopi-pi2bplus.patch

patch -p1 -i webiopi-pi2bplus.patch

4. Setup: sudo ./setup.sh

5. Connect the 5v relay with Raspberry Pi's GPIO PINS using jumper wires

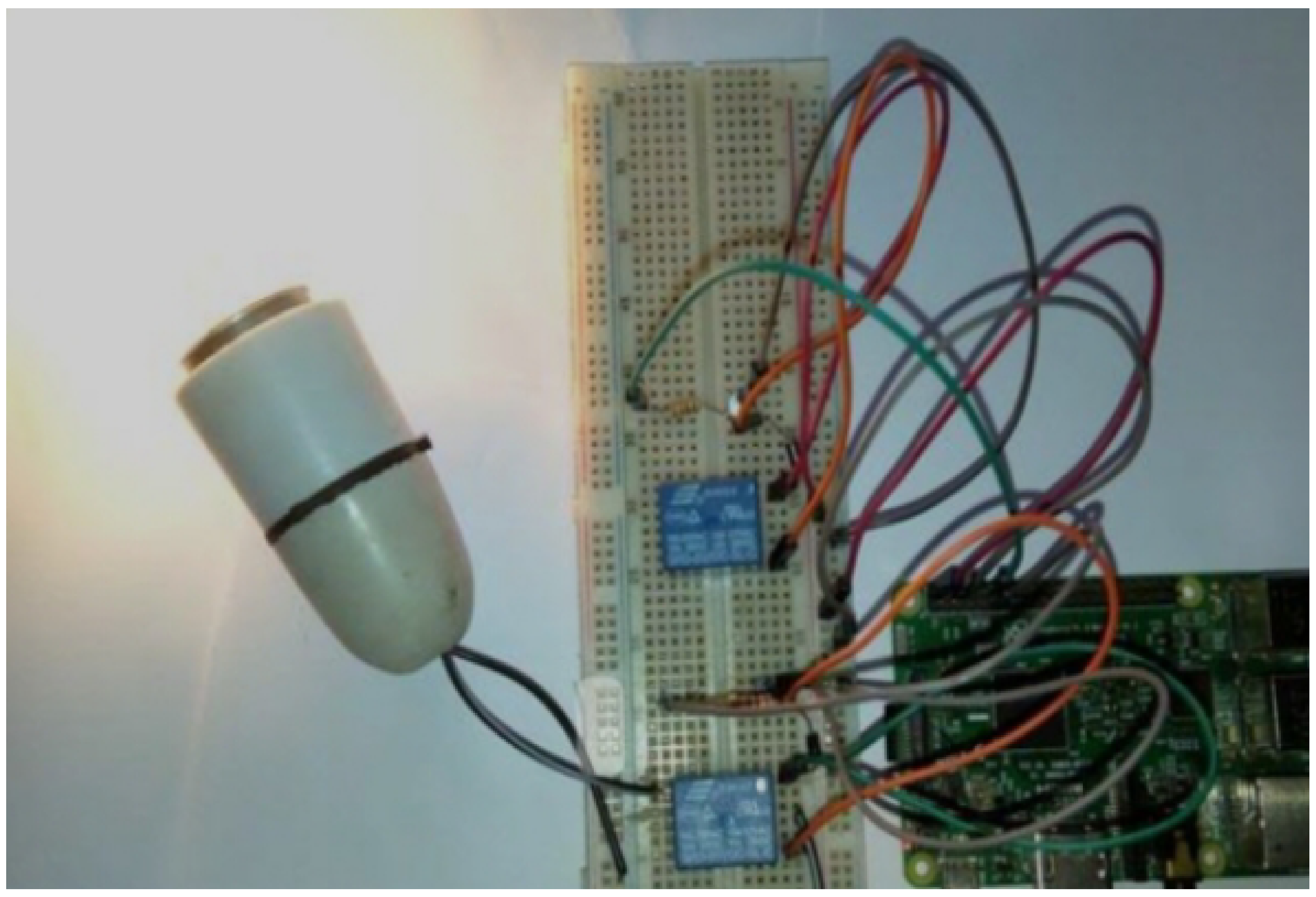
Output:



WebIOPi | GPIO Head...

WebIOPi | GPIO Header - Chromium

ALT0	GPIO 3	5	6	GROUND	
IN	GPIO 4	7	8	UART TX	
	GROUND	9	10	UART RX	
IN	GPIO 17	11	12	GPIO 18	IN
IN	GPIO 27	13	14	GROUND	
IN	GPIO 22	15	16	GPIO 23	IN
	3.3V	17	18	GPIO 24	IN
IN	GPIO 10	19	20	GROUND	
IN	GPIO 9	21	22	GPIO 25	IN
IN	GPIO 11	23	24	GPIO 8	IN
	GROUND	25	26	GPIO 7	IN
	--	27	28	--	
IN	GPIO 5	29	30	GROUND	
IN	GPIO 6	31	32	GPIO 12	IN
IN	GPIO 13	33	34	GROUND	
IN	GPIO 19	35	36	GPIO 16	IN
OUT	GPIO 26	37	38	GPIO 20	IN
	GROUND	39	40	GPIO 21	IN



11.Raspberry Pi GPS Module Interfacing

Aim of the experiment:

To interface GPS Module with Raspberry PI

Requirements:

Hardware Requirements

Sr. No	**Component**	***QTY***
1	USB–TTL Converter	1
2	Jumper Wires	6
3	Raspberry Kit	1
4	GPS Module	1

Software Requirements

-List of Libraries to be used for a particular practical:

Minicom package

Connections:

GPS PINS	USB-TTL PINS
VCC	VCC
Rx	No connection
Tx	Rx
GND	GND

Connect USB-TTL converter to Raspberry pI

Code:

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

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

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

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

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

#convert raw NMEA string into degree decimal format
```

```
def convert_to_degrees(raw_value):
    decimal_value = raw_value/100.00
    degrees = int(decimal_value)
    mm_mmmm = (decimal_value - int(decimal_value))/0.6
    position = degrees + mm_mmmm
    position = "%.4f" %(position)
    return position
gpgga_info = "$GPGGA,"
ser = serial.Serial ("/dev/ttyS0")              #Open port with baud rate
GPGGA_buffer = 0
NMEA_buff = 0
lat_in_degrees = 0
long_in_degrees = 0


try:
    while True:
        print("Hello World")
        received_data = (str)(ser.readline())                   #read NMEA string received
        GPGGA_data_available = received_data.find(gpgga_info)   #check for NMEA GPGGA
string
        if (GPGGA_data_available>0):
            GPGGA_buffer = received_data.split("$GPGGA,",1)[1]  #store data coming after
"$GPGGA," string
            NMEA_buff = (GPGGA_buffer.split(','))               #store comma separated data in
buffer
            print(NMEA_buff)
            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)
```

Special Procedures other than connection:

The first thing we will do under this is to edit the /boot/config.txt file. To do this, run the commands below:

sudo nano /boot/config.txt

At the bottom of the config.txt file, add the following linesdtparam=spi=ondtoverlay=pi3-disable-btcore_freq=250enable_uart=1force_turbo=1

ctrl+x to exit and press y and enter to save.

The next step is to disable the Pi's serial the getty service, the command will prevent it from starting again at reboot:

sudo systemctl stop serial-getty@ttyS0.service

sudo systemctl disable serial-getty@ttyS0.service

sudo systemctl enable serial-getty@ttyAMA0.service

sudo apt- get install minicom

To check for port:

dmesg | grep tty

Test can done using:

sudo cat /dev/ttyUSB0