INDEX

Practical No 1

Aim: Installing Raspbian on Raspberry Pi and executing applications on it using Python and node.js

Hardware: Raspberry Pi Kit

Software: Raspberry Pi OS, Python, Nodejs

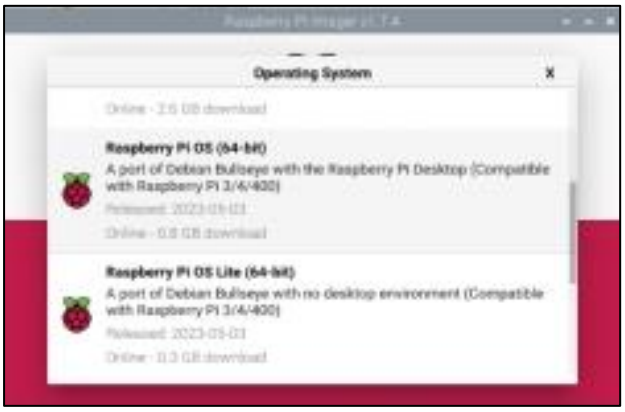
Steps to be followed:

Step 1: Installing Raspberry Pi

- To Install Raspberry Pi OS in SD Card, use **Raspberry Pi Imager**

- Select **'Choose OS'**

- Now we are only installing Raspberry PI OS only select **Raspberry PI OS (OTHER)**



- Choose This Version **64bit**(don't select the LITE VERSION)
- After Connecting all the required connection to the Raspberry Pi now boot it up.

Step 2: Intsall Python and Node.js

- To install Python, open terminal and give the below command:

```
sudo apt update
sudo apt install python3 idle3
```

Then set the variable path:

```
Echo 'export PATH="$PATH:/home/admin/.local/bin"'>> ~/bashrc
```

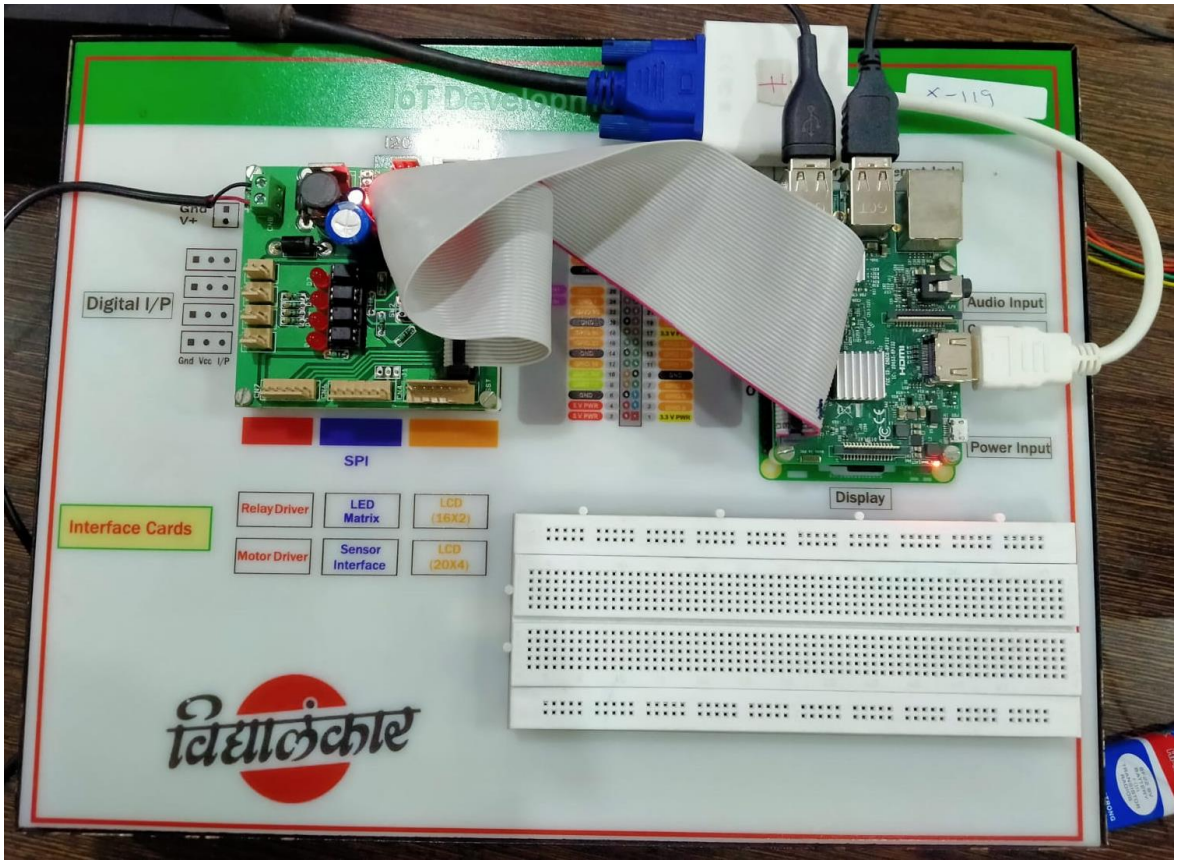
- To install Node.js give the below command in the terminal:

```
sudo apt-get install nodejs
```

Step 3: Running python code and nodejs code in raspberry pi

- Open Any Programming Editor and write the code

Raspberry Pi Circuit:



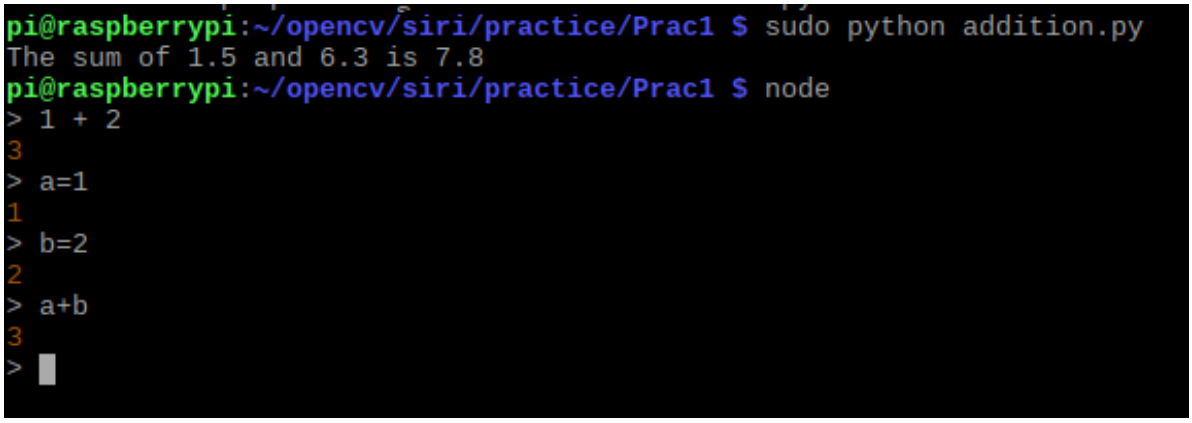
Source Code:

Python Code:

```
num1 = 1.5
num2 = 6.3

# Add two numbers
sum = num1 + num2

# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

For nodejs to execute type `node` on terminal.

Output:


pi@raspberrypi:~/opencv/siri/practice/Prac1 $ sudo python addition.py
The sum of 1.5 and 6.3 is 7.8
pi@raspberrypi:~/opencv/siri/practice/Prac1 $ node
> 1 + 2
3
> a=1
1
> b=2
2
> a+b
3
>

Practical No 2

Aim: Create a home automation system and control the devices remotely.

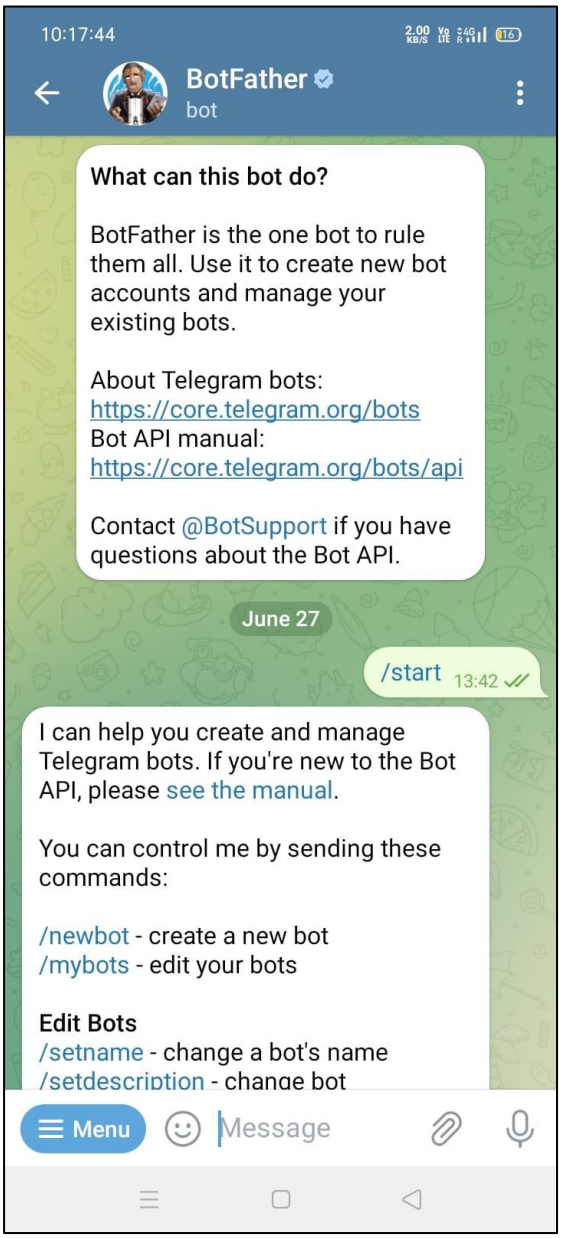
Hardware: Raspberry pi kit, relay, battery, bulb, fan, cables

Software: Raspberry Pi OS, Python, Telegram, Telepot

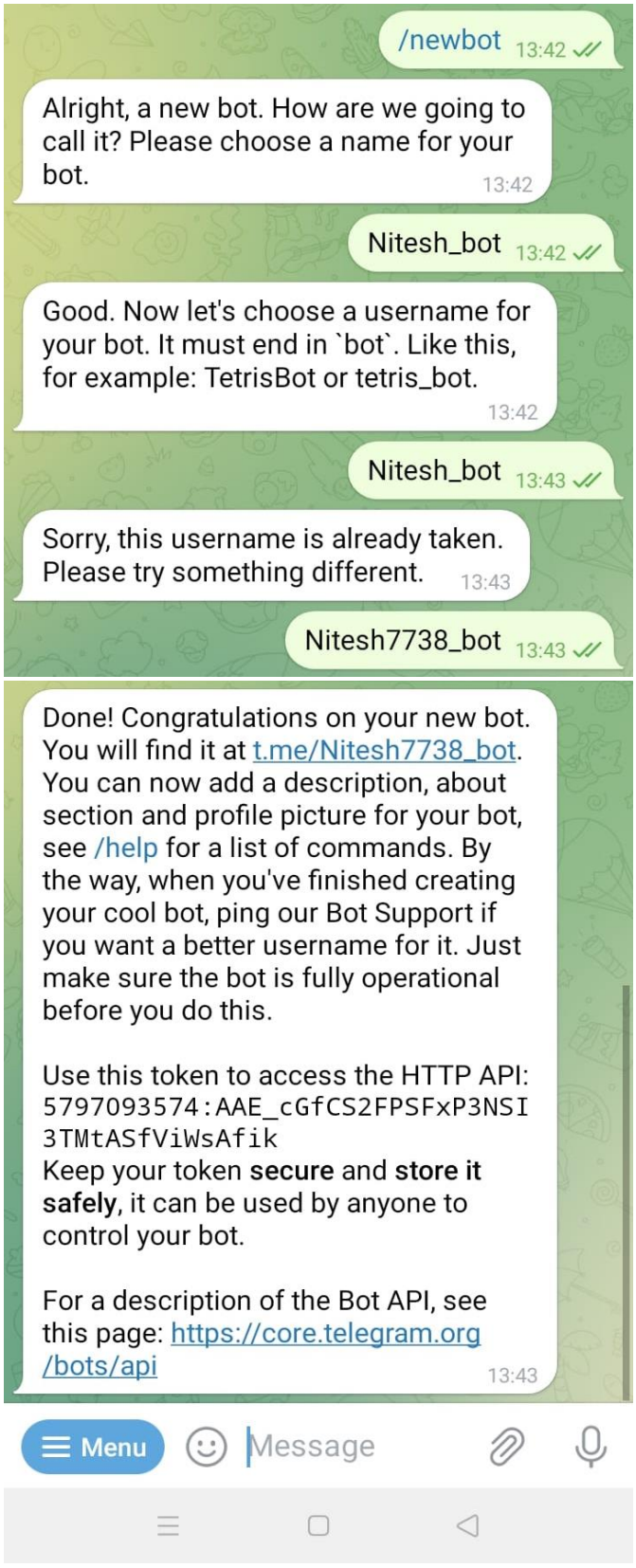
Steps to be followed:

Setting up BotFather in Telegram

- In Telegram App, search for BotFather.
- Send `/start` message

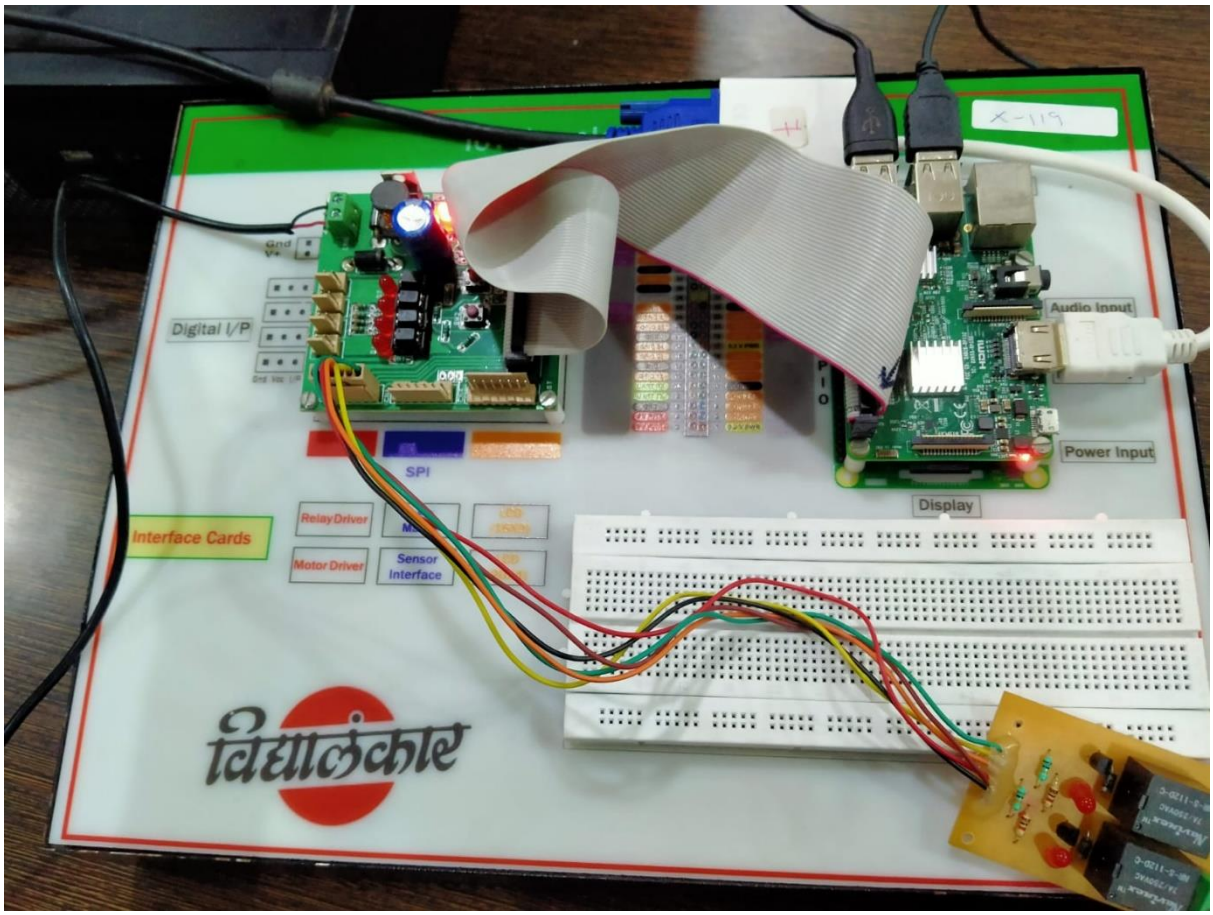


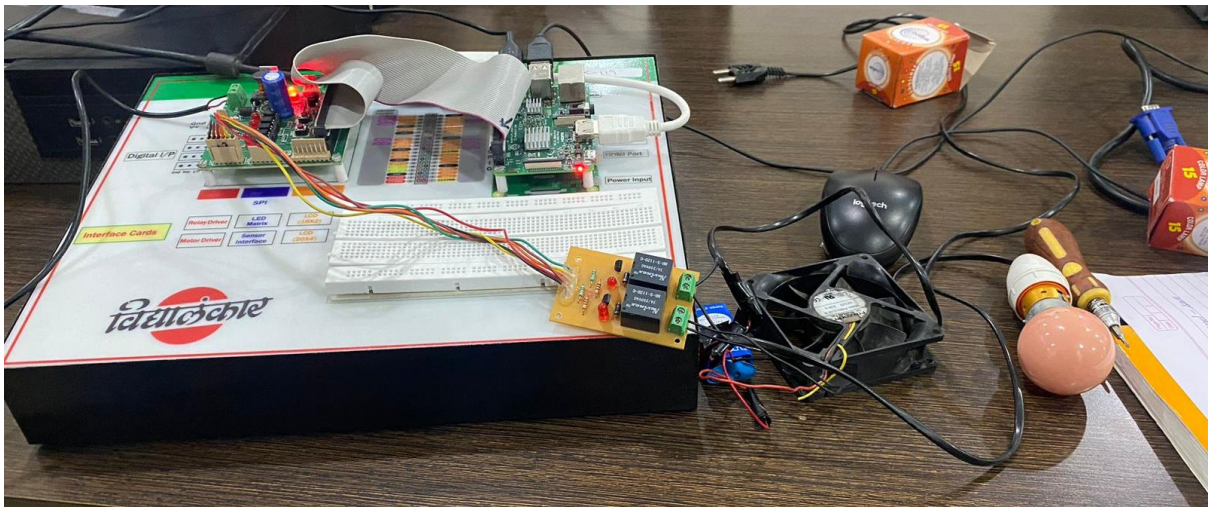
- To create a new bot, send `/newbot` message. (Refer to the image below)
- It will ask the bot name, give suitable name to your bot. If the name is available, bot will be created and token will be given to access the API, else will ask you to give the other name.



Raspberry Pi Circuit:

- Connect the relay, fan and bulb with cables. The other end of the relay connector connect it with relay driver slot on the raspberry pi kit. Refer to the below image for the circuit.



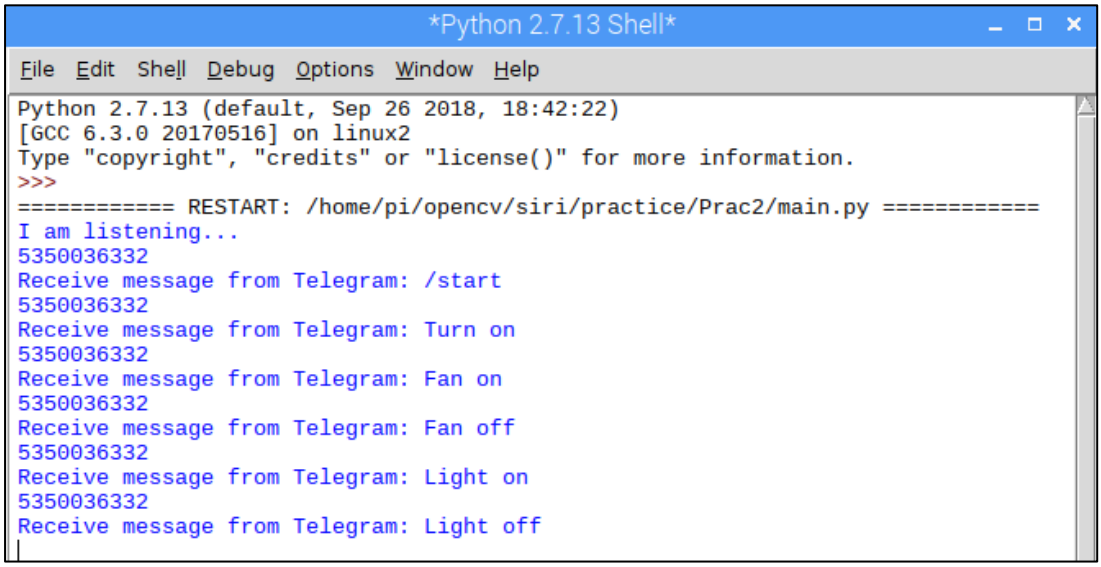


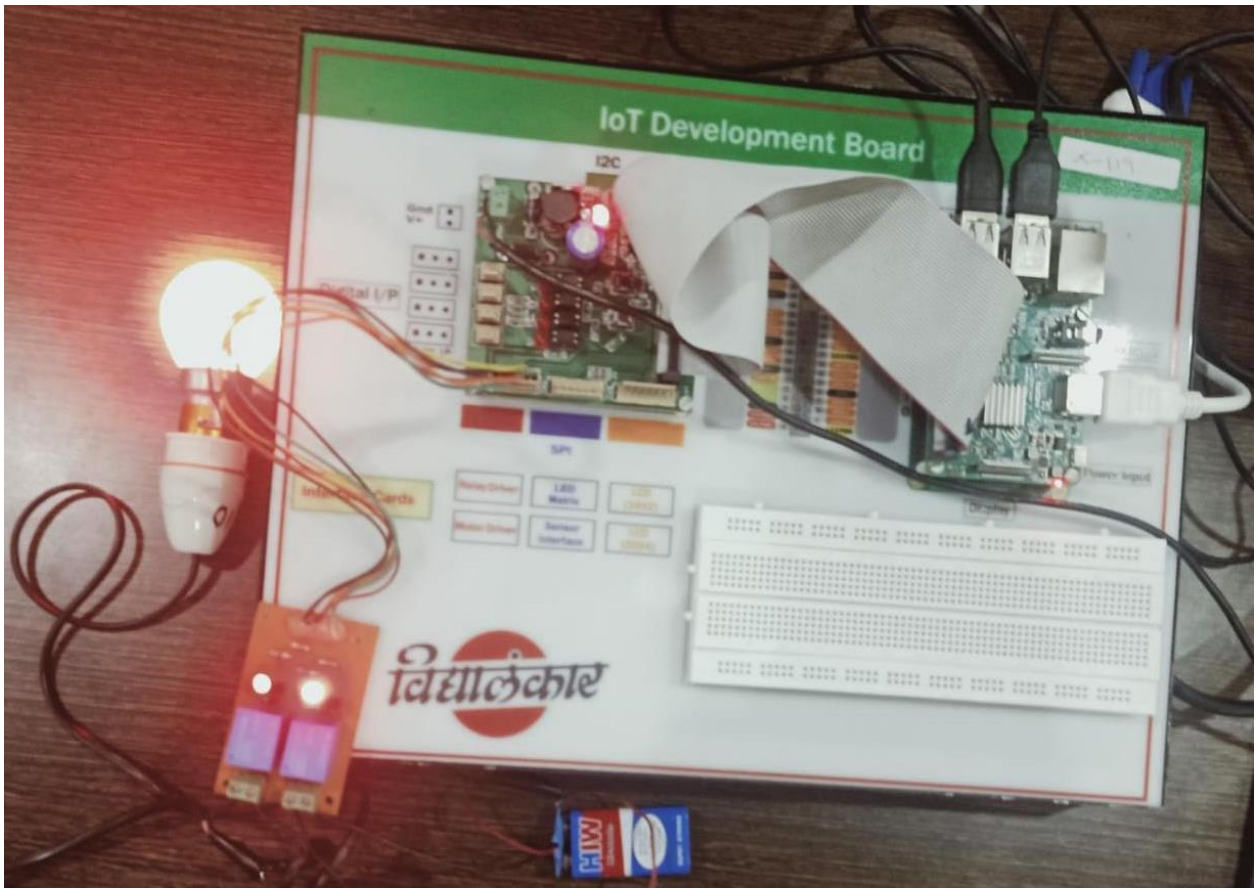
Source Code:

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

RELAY1 = 20
RELAY2 = 16

FAN   = RELAY1
LIGHT = RELAY2

GPIO.setwarnings(False)
# to use Raspberry Pi board pin numbers
GPIO.setmode(GPIO.BCM)
GPIO.cleanup()
# set up GPIO output channel
GPIO.setup(RELAY1, GPIO.OUT)
GPIO.setup(RELAY2, GPIO.OUT)

#Your Telegram token key variable.
telegramBotToken = '6212499066:AAEjXEFaH_LQV8OQ6SFn_ZYpa0RbhDQyHe8'


#function to on and off devices
def on(pin):
        GPIO.output(pin,GPIO.HIGH)
        return "on"

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

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

    print 'Receive message from Telegram: %s' % command

    if 'Fan' in command or 'fan' in command:
            if 'on' in command:
                    bot.sendMessage(chat_id, str( "Fan " + on(FAN) ))
            elif 'off' in command:
                    bot.sendMessage(chat_id, str( "Fan " + off(FAN) ))

    elif 'Light' in command or 'light' in command:
            if 'on' in command:
                    bot.sendMessage(chat_id, str( "Light " + on(LIGHT) ))
            elif 'off' in command:
                    bot.sendMessage(chat_id, str("Light " + off(LIGHT) ))

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

while 1:
```

```
    time.sleep(10)
```

Output:





Practical No 3

Aim: Implement Microservices on IoT device

Hardware: Raspberry pi kit

Software: Raspberry Pi OS, Python

Source Code:

Install Flask package with the below command in the terminal:

```
sudo pip install Flask
```

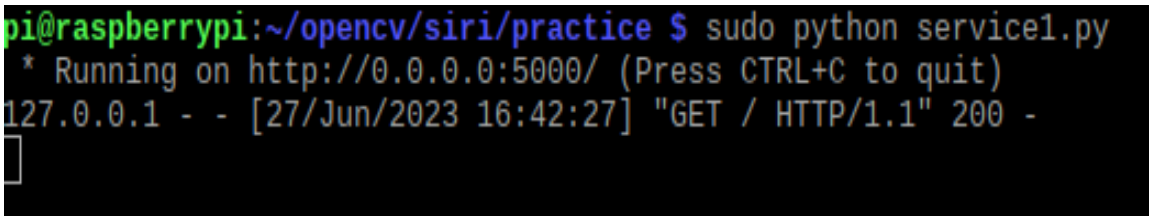
service1.py

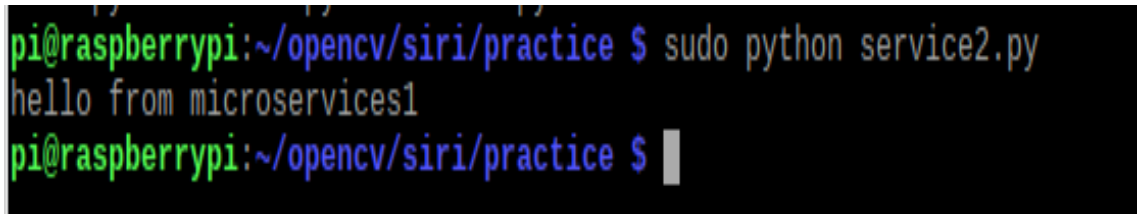
```
from flask import Flask
app=Flask(__name__)
@app.route('/')
def hello():
    return "hello from microservices1"
if __name__ =='__main__':
    app.run(host='0.0.0.0', port=5000)
```

service2.py

```
import requests
//replace the url in below statement with the url that you get
// after running the service1.py
response=requests.get("http://0.0.0.0:5000")
print (response.text)
```

Output:





Practical No 4

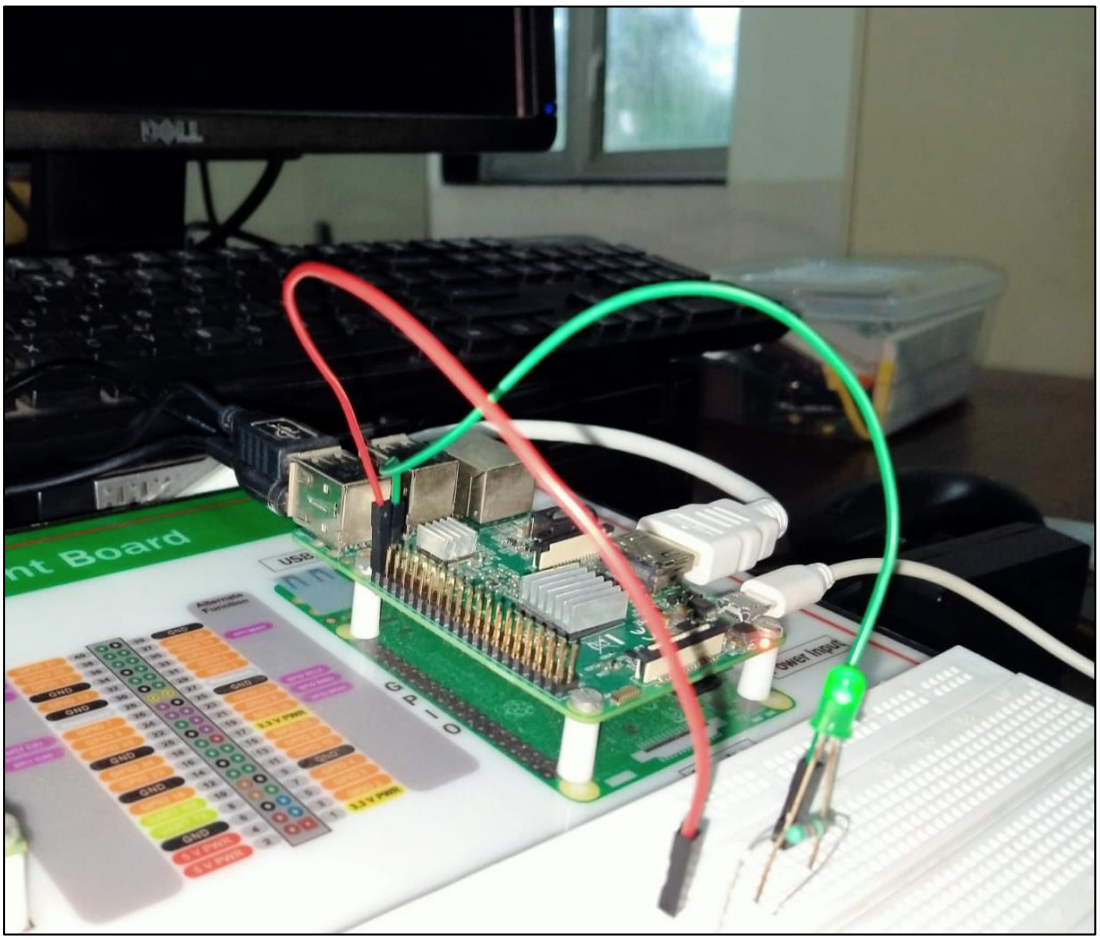
Aim: Build your own IoT platform

Hardware: Raspberry pi kit, LED, breadboard, cables, resistor, USB adapter 2.1 Amp

Software: Raspberry Pi OS, Python

Rasberry Pi circuit:

- Connect LED, resistor and cables on breadboard with raspberry pi as shown in the below image:



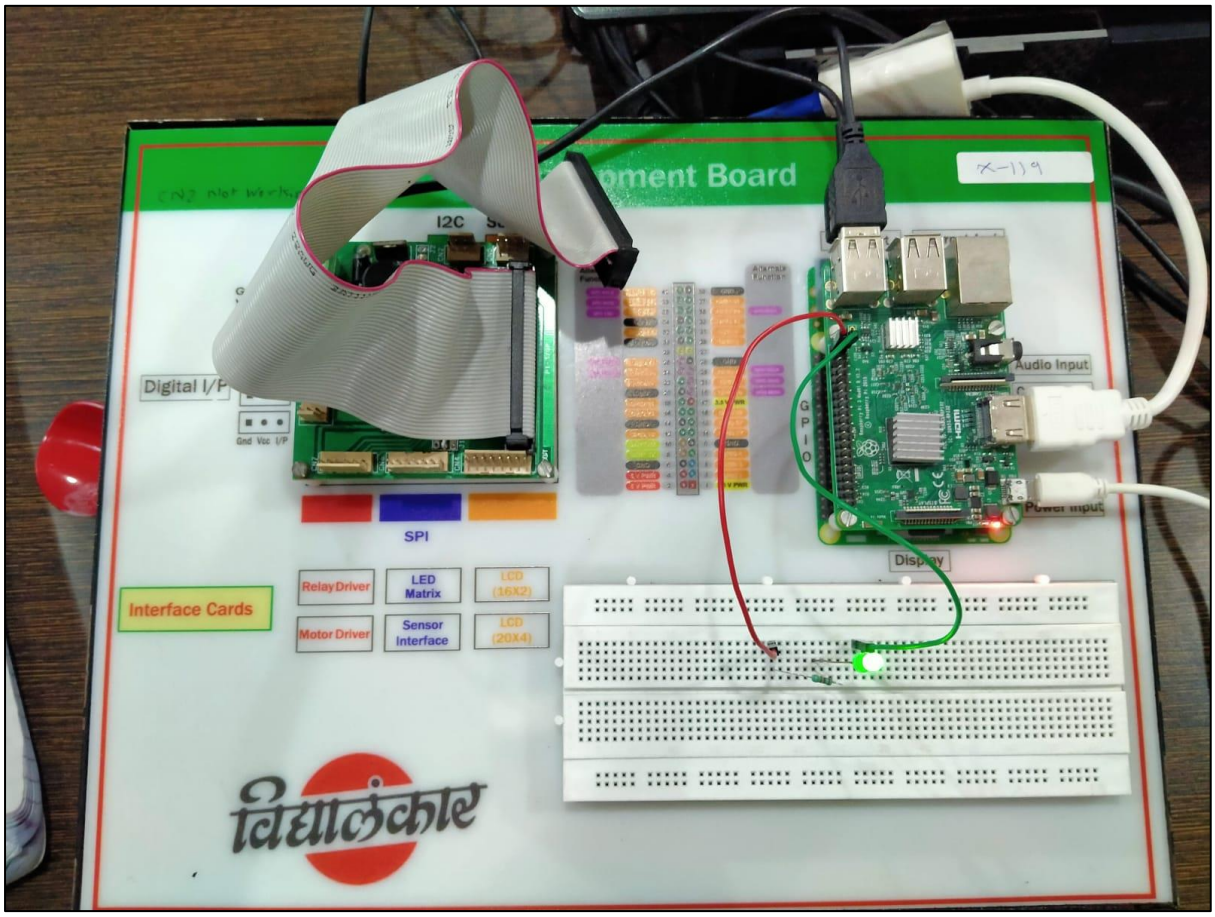
Source Code:

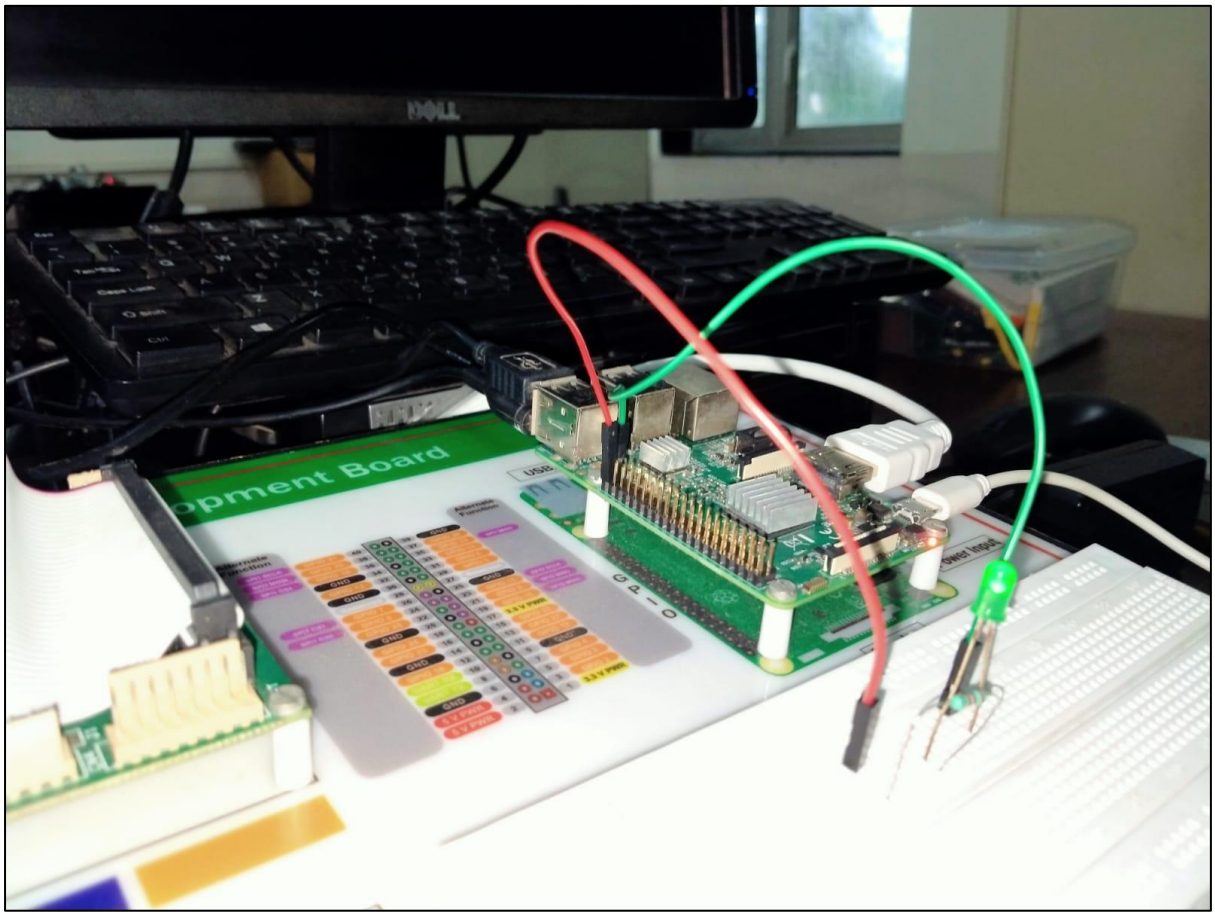
```
import RPi.GPIO as GPIO
from flask import Flask, request

#Define GPIO pin
led_pin = 21

#set GPIO mode
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)

#Create Flask app
app = Flask(__name__)

#Define route to handle HTTP POST request
@app.route('/',methods=['POST'])
def handle_post():
      message = request.get_data(as_text=True)
      if message == "ON":
            GPIO.output(led_pin, GPIO.HIGH)
      elif message == "OFF":
            GPIO.output(led_pin, GPIO.LOW)
        return 'OK'

if __name__ =='__main__':
      app.run(host='0.0.0.0', port=8080)
```

Output:

```
pi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ chmod +X iot_platform.py
pi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ sudo ch
chage            chattr           chgpasswd        chpasswd         chsh
chardet3         chcon            chgrp            chromium-browser chvt
chardetect       chcpu            chmod            chroot
chardetect3      chfn             chown            chrt
pi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ sudo chmod +X iot_platform.py
pi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ sudo python iot_platform.py
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
127.0.0.1 - - [27/Jun/2023 16:41:31] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [27/Jun/2023 16:41:42] "POST / HTTP/1.1" 200 -
^Cpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ sudo python iot_platform.py
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

```
pi@SHAIKH:~/Desktop/Adv IoT $ cd Practical\ 6
pi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ curl -X POST -d "ON" http://127.0.0.1:8080
OKpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ curl -X POST -d "OFF" http://127.0.0.1:8080
OKpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ curl -X POST -d "ON" http://127.0.0.1:8080
OKpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ curl -X POST -d "OFF" http://127.0.0.1:8080
OKpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ curl -X POST -d "ON" http://127.0.0.1:8080
OKpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ curl -X POST -d "ON" http://127.0.0.1:8080
OKpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ curl -X POST -d "OFF" http://127.0.0.1:8080
OKpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ curl -X POST -d "ON" http://127.0.0.1:8080
OKpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ curl -X POST -d "Off" http://127.0.0.1:8080
OKpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ curl -X POST -d "OFF" http://127.0.0.1:8080
OKpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $ curl -X POST -d "On" http://127.0.0.1:8080
OKpi@SHAIKH:~/Desktop/Adv IoT/Practical 6 $
```


I2C
Digital I/P
SPI
Interface Cards
Relay Driver
LED Matrix
LCD (16X2)
Motor Driver
Sensor Interface
LCD (20X4)
Audio Input
GPIO
विद्यालंकार


DELL
USB
GND
3.3V PWR

Practical No 5

Aim: Face Detection using IoT Device

Hardware: Raspberry pi kit, Pi Camera

Software: Raspberry Pi OS, Python

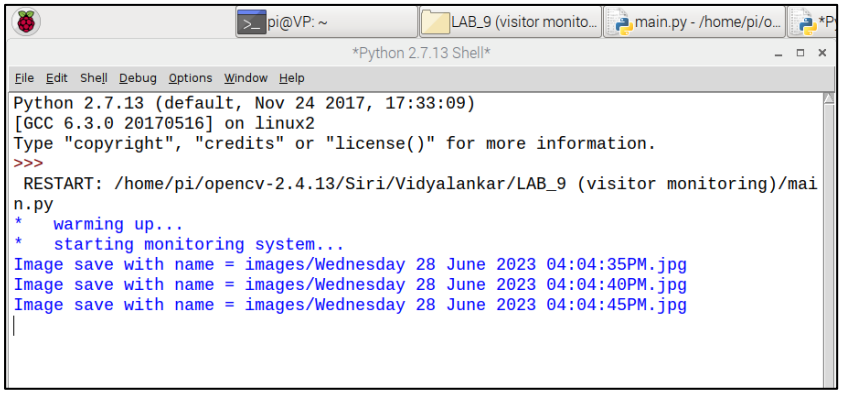
Rasberry Pi circuit:

- Connect pi camera on the Raspberry Pi kit as shown below:

Source Code:

```
import cv2
import numpy as np
import datetime
import time

# Initialize the camera capture object
cap = cv2.VideoCapture(0) # 0 represents the default camera

# Load the pre-trained face detection model
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Start the main loop to capture frames from the camera
while True:
    ret, frame = cap.read() # Read a frame from the camera

# Convert the frame to grayscale for face detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

# Perform face detection
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3,
minNeighbors=5)

# Draw rectangles around the detected faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        timestamp = datetime.datetime.now()
        ts = timestamp.strftime("%A %d %B %Y %I:%M:%S%p")
        cv2.imwrite("images/" + str(ts) + ".jpg", frame)
        print "Image save with name = " + "images/" + str(ts) + ".jpg"

# Display the frame with detected faces
    cv2.imshow('Face Detection', frame)

# Break the loop if 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the camera and close the windows
cap.release()
cv2.destroyAllWindows()
```

Output:


pi@VP: ~
LAB_9 (visitor monito...
main.py - /home/pi/o...
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (default, Nov 24 2017, 17:33:09)
[GCC 6.3.0 20170516] on linux2
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: /home/pi/opencv-2.4.13/Siri/Vidyalankar/LAB_9 (visitor monitoring)/main.py
* warming up...
* starting monitoring system...
Image save with name = images/Wednesday 28 June 2023 04:04:35PM.jpg
Image save with name = images/Wednesday 28 June 2023 04:04:40PM.jpg
Image save with name = images/Wednesday 28 June 2023 04:04:45PM.jpg