

A thick black L-shaped frame surrounds the text. It consists of a horizontal bar at the top left, a vertical bar on the left side, and another L-shaped piece at the bottom right.

IOT PHASE 3

Development part 1

Topic: Traffic control management systems

CONTENT

- Introduction
- Required components
- System implementation
- Block diagram
- Steps
- Python code
- Conclusion

INTRODUCTION

- In an old automatic traffic controlling a traffic light uses timer for every phase.
- Using electronic sensors is another way in order to detect vehicles, and produce signal that to this method the time is being wasted by a green light on an empty road.
- Traffic congestion also occurred while using the electronic sensors for controlling the traffic.
- All these drawbacks are supposed to be eliminated by using image processing. We propose a system for controlling the traffic light by image processing.
- The vehicles are detected by the system through images instead of using electronic sensors embedded in the pavement.

INTRODUCTION

- A camera will be placed alongside the traffic light.
- It will capture image sequences.
- Image processing is a better technique to control the state change of the traffic light.
- It shows that it can decrease the traffic congestion and avoids the time being wasted by a green light on an empty road.
- It is also more reliable in estimating vehicle presence because it uses actual traffic images.
- It visualizes the practicality, so it functions much better than those systems that rely on the detection of the vehicles' metal content.

REQUIRED COMPONENTS

- Raspberry Pi
- Camera
- Yellow green and red indicators
- Ultrasonic sensors
- ThingSpeak- iot platform
- Smart counter unit

SMART COUNTER UNIT

- This unit counts the number of vehicle passing on the road.
- Hardware and system code make this unit smart i.e. capable of counting two wheeler and four wheeler vehicles separately per minute.
- This unit also counts the total number of vehicles.
- Unit uses separate two counters for two type of vehicles i.e. two wheeler and four wheeler.
- Prototype road is made in such a way that the smart counter unit can count vehicle counts individually.

RASPBERRY PI

- System is using raspberry pi 3 B+ model as a controller. Raspberry pi is used to perform operations and control the system flow.
- Reference [6] the Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range.
- • Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- • 1GB LPDDR2 SDRAM
- • 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- • Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- • Extended 40-pin GPIO header
- • Full-size HDMI
- • 4 USB 2.0 ports

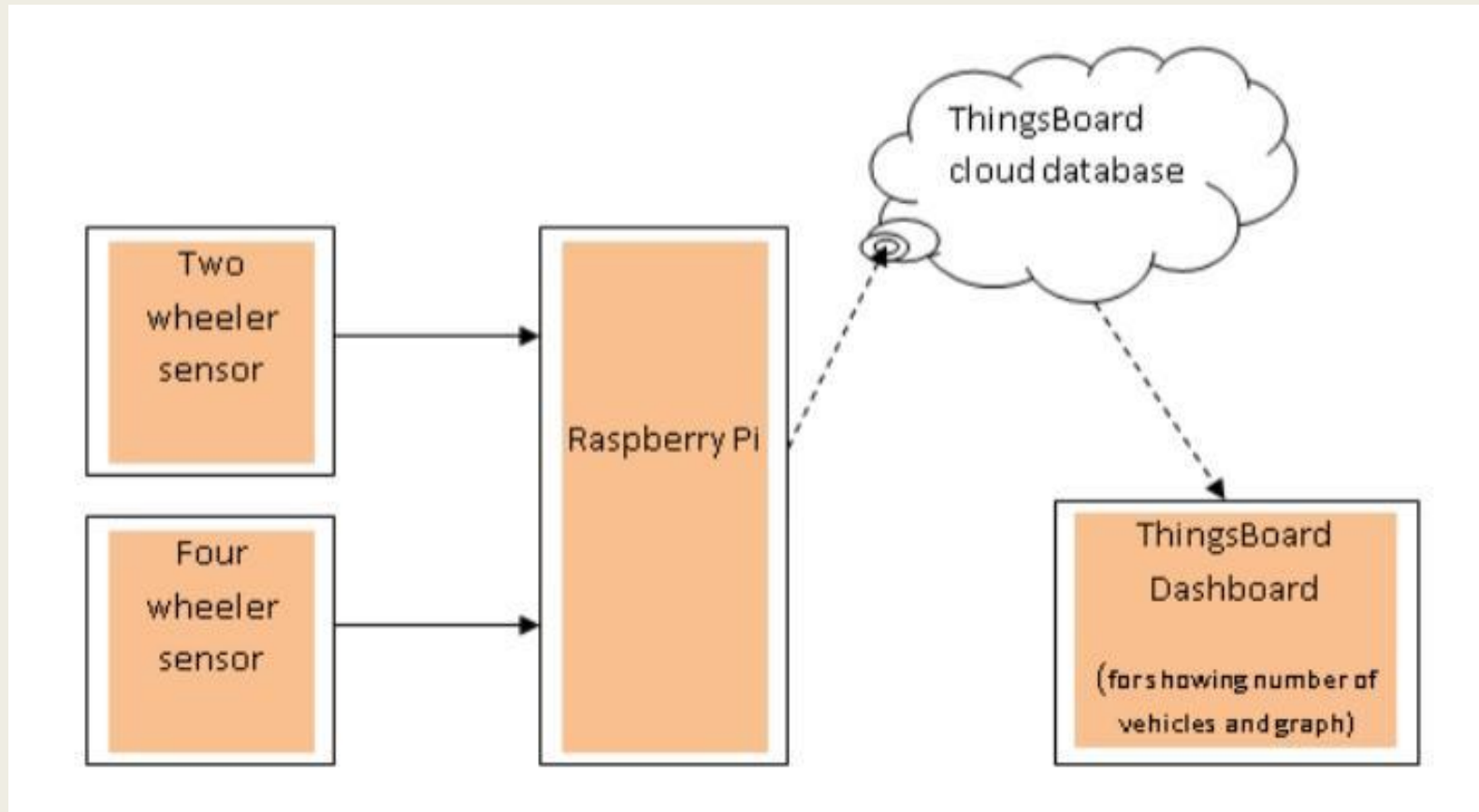
RASPBERRY PI

- `` Extended 40-pin GPIO header
- `` Full-size HDMI
- `` 4 USB 2.0 ports
- `` CSI camera port for connecting a Raspberry Pi camera
- `` DSI display port for connecting a Raspberry Pi touchscreen display
- `` 4-pole stereo output and composite video port
- `` Micro SD port for loading your operating system and storing data
- `` 5V/2.5A DC power input
- `` Power-over-Ethernet (PoE) support (requires separate PoE HAT)

System Implementation

- System uses hardware comprising of counting unit, WiFi enabled raspberry pi, LED screen.
- Software has python code to control the flow through raspberry pi and ThingsBoard open source IOT platform.

BLOCK DIAGRAM



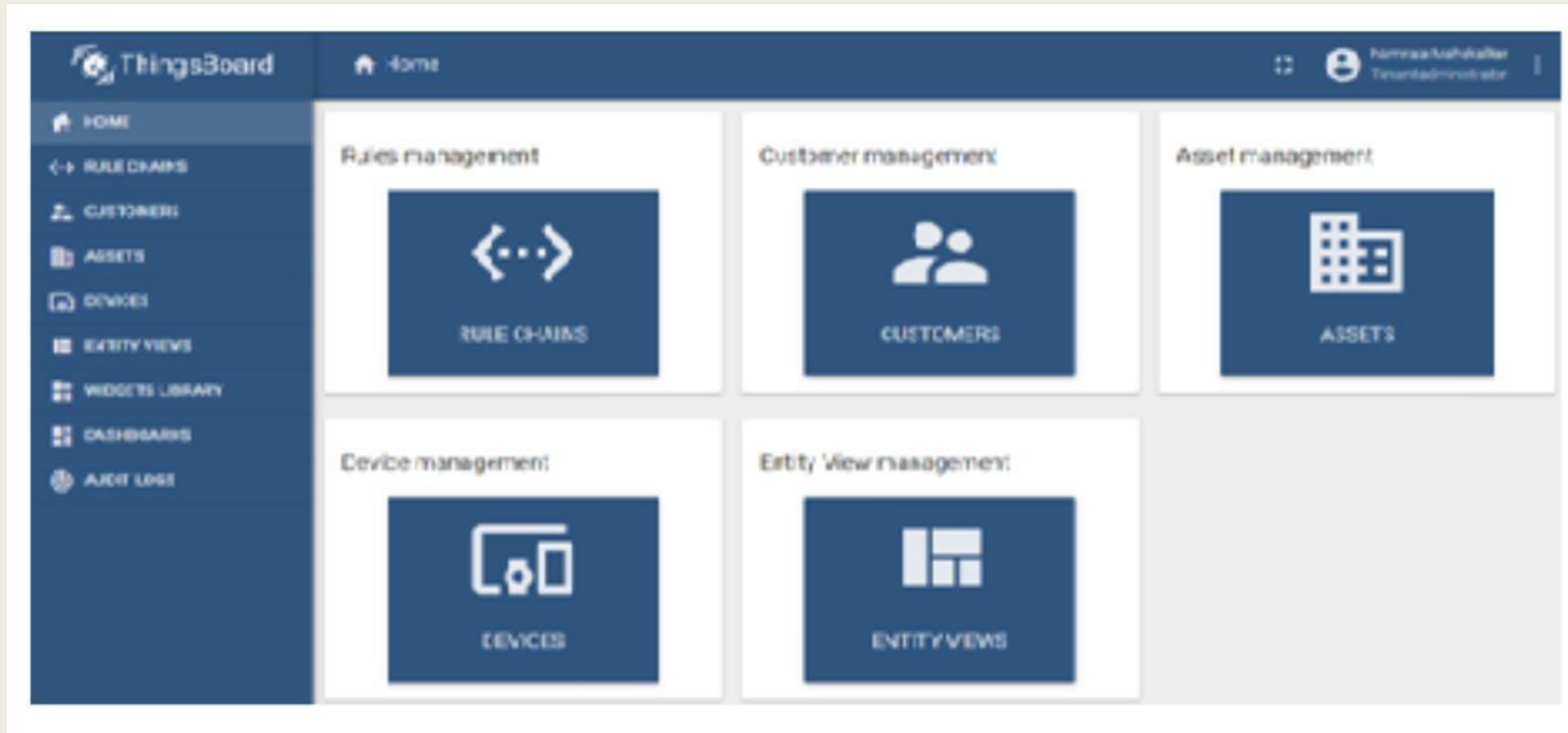
STEPS

- Sensor sense the data and send to the raspberry pi.
- Raspberry pi increment the counter value with the help of python script.
- Raspberry pi is WiFi enabled and the data transmitted over MQTT to the ThingsBoard dashboard via internet cloud.
- ThingsBoard provide the user friendly interface. User has to first create the account on ThingsBoard.
- All the setup related information is given in reference .

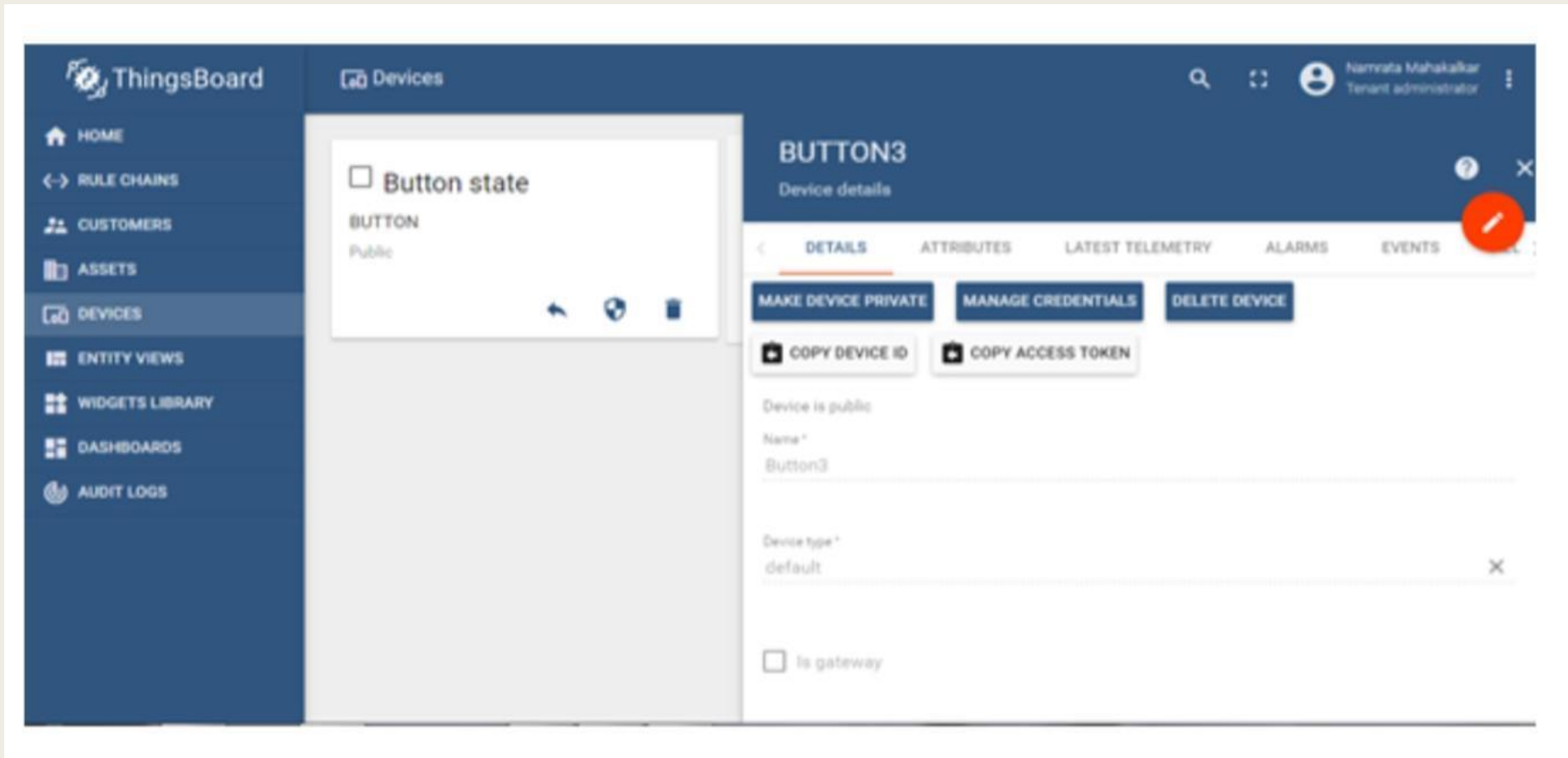
STEPS

- User can create the device, dashboard, component linking defining relationship among them.
- User can also configure and see various type of graphs and can attach to the devices so that analysis becomes easy.
- User can even set the permission for different customer to different devices.
- User can also make the device as public so anyone can access the information with the help of given link.

ThingsBoard: Home page



ThingsBoard: Create a new device



ThingsBoard: Create a new dashboard

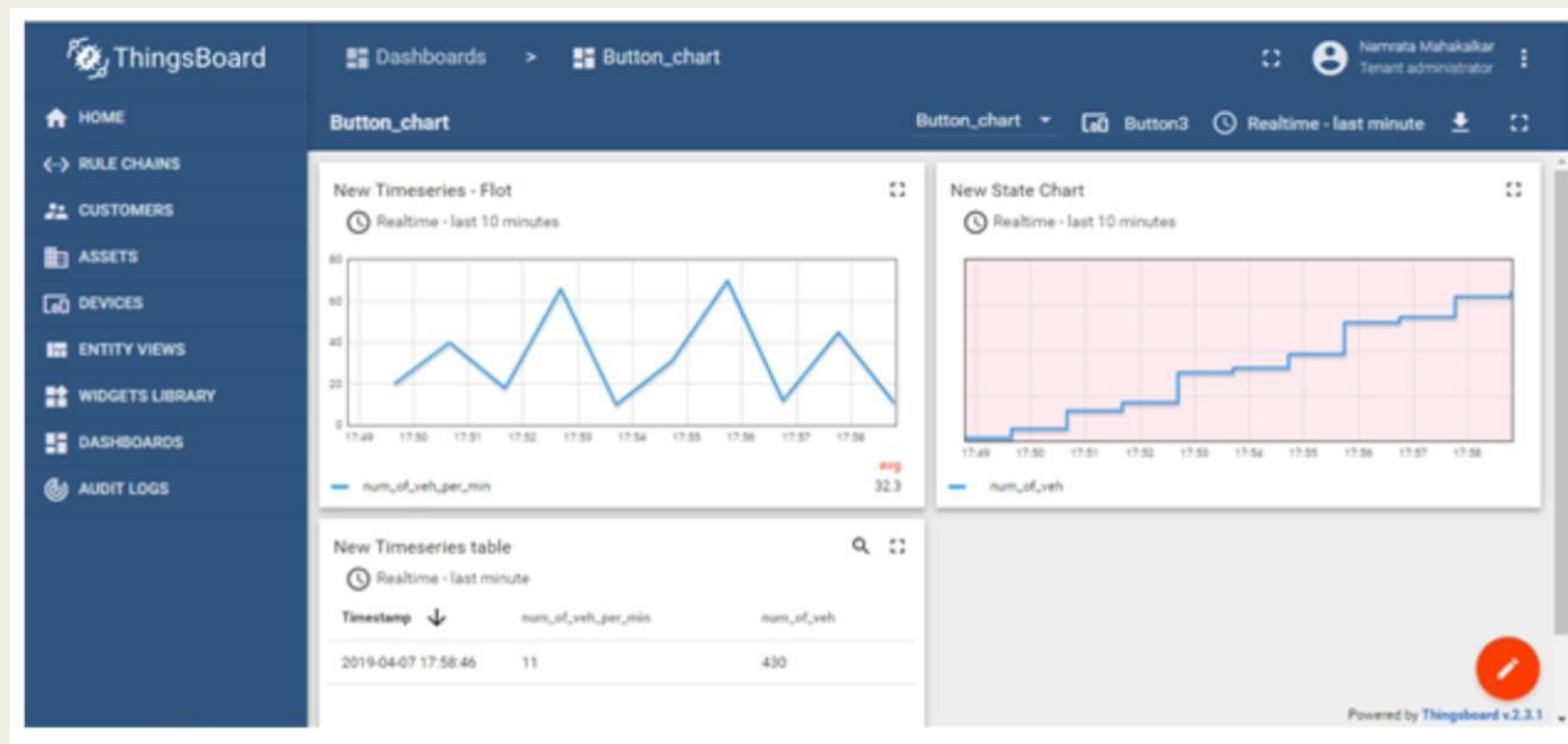
The screenshot displays the ThingsBoard web interface. On the left is a dark blue sidebar with navigation links: HOME, RULE CHAINS, CUSTOMERS, ASSETS, DEVICES, ENTITY VIEWS, WIDGETS LIBRARY, DASHBOARDS, and AUDIT LOGS. The main header is dark blue and shows 'Dashboards > Button_chart' along with user information 'Namrata Mahalkar, Tenant administrator'. Below the header, the dashboard title 'Button_chart' is followed by a dropdown menu showing 'Button3' and a clock icon indicating 'Realtime - last minute'. The dashboard contains three widgets:

- New Timeseries - Flot**: A line chart titled 'Realtime - last 10 minutes' showing a single data series 'num_of_veh_per_min' on a grid. The x-axis represents time from 19:02 to 19:11, and the y-axis ranges from -1 to 1. A red 'avg' label is visible at the bottom right of the chart area.
- New State Chart**: A state chart titled 'Realtime - last 10 minutes' showing a single data series 'num_of_veh' on a grid. The x-axis represents time from 19:02 to 19:11, and the y-axis ranges from -1 to 1. A red 'avg' label is visible at the bottom right of the chart area.
- New Timeseries table**: A table titled 'Realtime - last minute' with columns 'Timestamp', 'num_of_veh_per_min', and 'num_of_veh'. The table is currently empty, displaying 'NO DATA FOUND'.

At the bottom right of the dashboard, there is a red circular button with a white pencil icon and the text 'Powered by Thingsboard v2.3.1'.

Result on ThingsBoard dashboard

- Here two charts for the number of vehicle per minute and number of total vehicle passed with time are shown respectively.
- Third chart is a card shows number of vehicle per minute in numerical values.



Python code

```
mqtt.Client.connected_flag=False#create flag in class
mqtt.Client.suppress_puback_flag=False
client = mqtt.Client("python1") #create new instance
#client.on_log=on_log
client.on_connect = on_connect
client.on_disconnect = on_disconnect
client.on_publish = on_publish
broker="demo.thingsboard.io"
port =1883
topic="v1/devices/me/telemetry"
#need to edit user name
username="xyx" #device house --- Enter your own user name
password=""
```

```
if username != "":
```

```
    pass
```

```
    client.username_pw_set(username, password)
```

```
    client.connect(broker,port) #establish connection
```

```
    while not client.connected_flag: #wait in loop
```

```
        client.loop()
```

```
    time.sleep(.1)
```

```
    time.sleep(.3)
```

```
    data=dict()
```

```
    # set GPIO24 as an output (LED)
```

```
    #for i in range(100):
```

```
        while True:
```

```
            num_of_veh_per_min= 0
```

```
            for i in range(6000): # the counter is incrementing every 10 mSec. the data is updated every 60 Sec  
                i.e. every 1 min. Hence  $6000 * 10 \text{ mSec} = 1 \text{ min}$ 
```

```
channel=GPIO.wait_for_edge(3, GPIO.RISING, timeout= 10)
# wait if a vehicle passes with a time out of 10mSec - debounce time
if channel is None:
    if i%100 == 0: # print the timer value after every second beacuse printing timer value every 10mSec
    increases delay
    print('Data will be uploaded after ',60 - i/100,' Seconds')
else:
    num_of_veh = num_of_veh +1 # increment the total number of vehicles
    num_of_veh_per_min=num_of_veh_per_min+1 # increment the vehicle per min count
    print("Number of vehicle",num_of_veh) # print the total numner of vehicle counted till now
    time.sleep(0.01)
    data["num_of_veh"]=num_of_veh
    data["num_of_veh_per_min"]=num_of_veh_per_min
    data_out=json.dumps(data) #create JSON object
    print("publish topic",topic, "data out= ",data_out)
    ret=client.publish(topic,data_out,0) #publish
    client.loop()
    client.disconnect()
```

Conclusion

- This project presents an Image Processing Based Intelligent Traffic Control System by using Raspberry pi which we are going to implement using PYTHON Programming Language.
- “Image Processing Based Intelligent Traffic control using Raspberry Pi” technique that we propose overcomes all the limitations of the earlier (in use) techniques used for controlling the traffic.
- Earlier in automatic traffic control use of timer had a drawback that the time is being wasted by green light on the empty.
- This technique will avoid all this problem.