19L039 – PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTERPRENEURSHIP

Date	19 November 2022
Team ID	PNT2022TMID12941
Title	Signs with smart connectivity for better road safety

KANISHA R (19L120)

DEEPAK APPA RAO (20L404)

MADHUMITHA K (20L407)

SOUNDARYALAXMI B (20L416)

(BE ECE G1 4th year)

BACHELOR OF ENGINEERING

Branch: ELECTRONICS AND COMMUNICATION ENGINEERING

of Anna University

19TH NOVEMBER 2022

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

CONTENTS

CHAPTER 1

INTRODUCTION

1.1Project Overview

Traffic congestion is a condition in transport that is characterized by slower speeds, longer trip times, and increased vehicular queueing. Traffic congestion on urban road networks has increased substantially since the 1950s. When traffic demand is great enough that the interaction between vehicles slows the speed of the traffic stream, this results in some congestion.

This leads to several negative effects such as: Wasting time of motorists and passengers ("opportunity cost"), delays which may result in late arrival for employment, meetings, and education, resulting in lost business, disciplinary action or other personal losses, wasted fuel which increases air pollution and carbon dioxide emissions owing to increased idling, acceleration and braking, wear and tear on vehicles as a result of idling in traffic and frequent acceleration and braking, leading to more frequent repairs and replacements, blocked traffic may interfere with the passage of emergency vehicles traveling to their destinations where they are urgently needed, higher chance of collisions due to tight spacing and constant stopping-and-going.

In this project, the sign boards in roads will be connected to the internet and upcoming road conditions will be updated frequently to the users. This will help the users to schedule their plan accordingly.

1.2 Purpose

The main purpose of the project is to update the road conditions and weather conditions in sign boards to facilitate users for safe journey.

This also ensures the traffic density is updated and so that emergency vehicles can take another route.

This also facilitates in saving the time of passengers and make them reach their destination earlier.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing problem

Traffic congestion is a condition in transport that is characterized by slower speeds, longer trip times, and increased vehicular queueing. Traffic congestion on urban road networks has increased substantially since the 1950s. When traffic demand is great enough that the interaction between vehicles slows the speed of the traffic stream, this results in some congestion.

This leads to several negative effects such as: Wasting time of motorists and passengers ("opportunity cost"), delays which may result in late arrival for employment, meetings, and education, resulting in lost business, disciplinary action or other personal losses, wasted fuel which increases air pollution and carbon dioxide emissions owing to increased idling, acceleration and braking, wear and tear on vehicles as a result of idling in traffic and frequent acceleration and braking, leading to more frequent repairs and replacements, blocked traffic may interfere with the passage of emergency vehicles traveling to their destinations where they are urgently needed, higher chance of collisions due to tight spacing and constant stopping-and-going.

2.2 References

[1] Harshitha D, Ifra Anjum, Harshitha S P, Indushree V P, "IOT Based Smart Traffic Signal Monitoring System", International Journal of Engineering Research & Technology, 2019.

Proposed Work: This paper mainly focusses on controlling traffic signals in urban and sub urban areas based on the traffic density. The concept of Internet of Things is utilized to collect, process the traffic data, and control various traffic signals. IR sensors are used to measure the traffic density and this data is given to Arduino Uno microcontroller board. Based on sensor data, the traffic density is computed and updated to cloud by using wifi module. The problem of emergency travel is also addressed here. RFID tags are provided to emergency vehicles and the RFID readers will read the RFID tags and update information about the emergency vehicles so that traffic can be cleared. The location of non working traffic signals are also updated by using GPS module.

[2] G Vasantha, B Pavithra, A Poornima, G Sriharisudheer, G Sreenivasulu, R Rajagopal, "IOT Based Smart Roads Intelligent Highways With Warning Message and Diversions According to Climate Conditions", International Journal of Engineering and Technology, 2018.

Proposed Work: This paper mainly focuses on utilizing sensors and actuators along with Internet of Things for highway monitoring and updation. Here, LDR sensors, IR sensors, MQ-6 module, LPC 2148 microcontroller and wifi module are used for the entire operation. The 3 sensors utilized here monitors the intensity of light in the area, accident detection and gas leakage detection respectively. Based on the sensor data, the streets lights can be turned off and the information about accident can

be updated to the cloud through wifi module. The proposed systems provide cost efficient means of highway monitoring.

[3] Ashok P V, SivaSankari S, Vignesh Mani, Suresh Sankaranarayanan, "IOT Based Traffic Signalling System", International Journal of Applied Engineering Research, Volume 9, No. 19, 2017.

Proposed Work: This paper mainly focuses on controlling the traffic signals in urban and sub urban areas. This paper utilizes the concept of Internet of Things to monitor and update information about traffic. Here, ultrasonic sensors are used to monitor the traffic density. From the sensor data, the vehicle count is calculated and given to Arduino Uno microcontroller board which process the sensor data. The traffic density analysis is done by Raspberry Pi3 module, and the analysis results are updated to the cloud by means of wifi module. Based on the analysis results, the traffic lights are controlled by the system. The proposed system provides detailed traffic density analysis and traffic light control.

2.3 Problem Statement Definition

Problem statement 1:

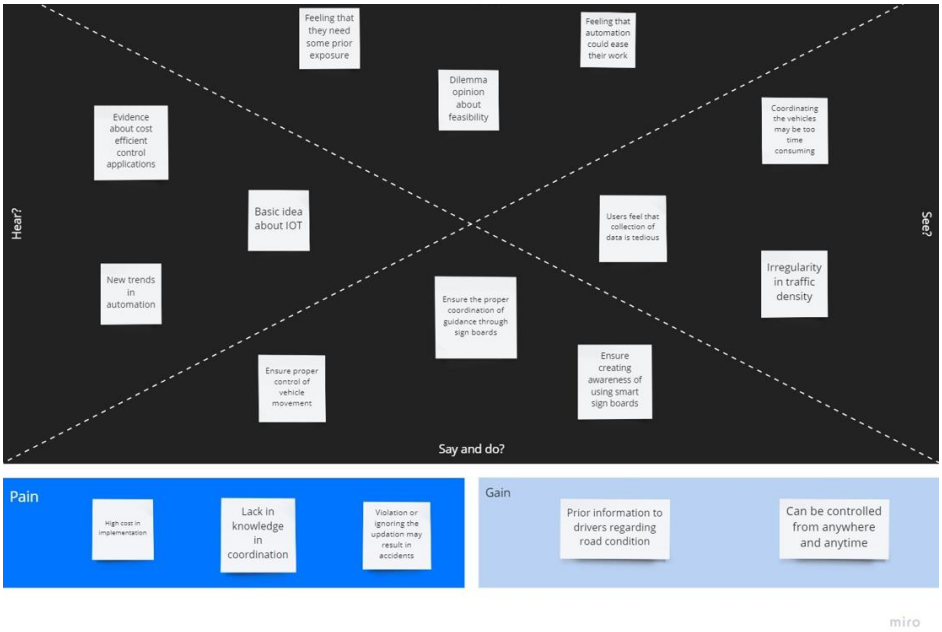
Problem statement 2:

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Traveller.	Reach destination earlier.	I got stuck intraffic.	I was unaware about the traffic density and road condition.	Impatient.
PS-2	Traveller.	Reach hill station.	I got delayed.	I was unaware about the weatherconditions.	Impatient.

CHAPTER 3

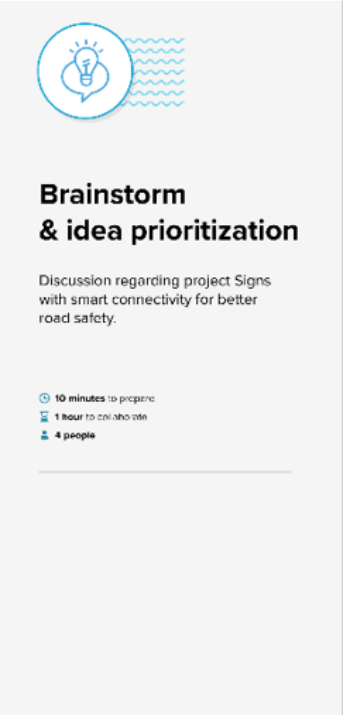
IDEATION & PROPOSED SOLUTION

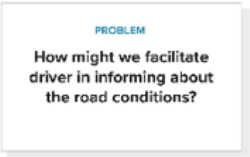
3.1 Empathy Map Canvas



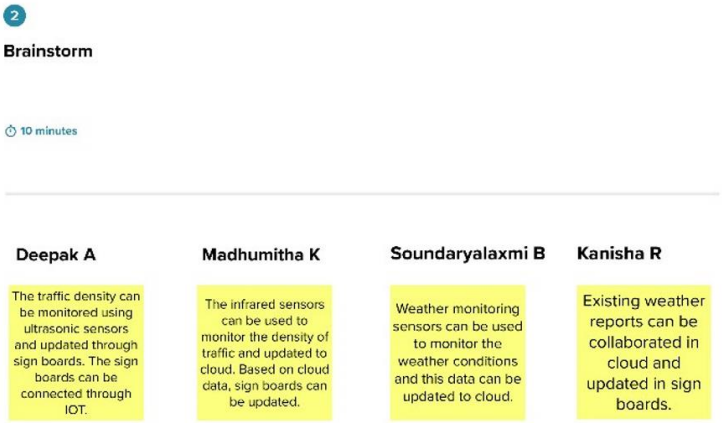
3.2 Ideation & Brainstorming

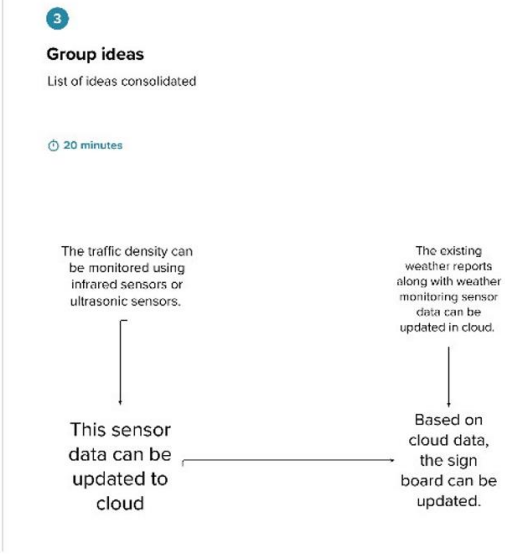
Step-1: Team Gathering, Collaboration and Select the Problem Statement



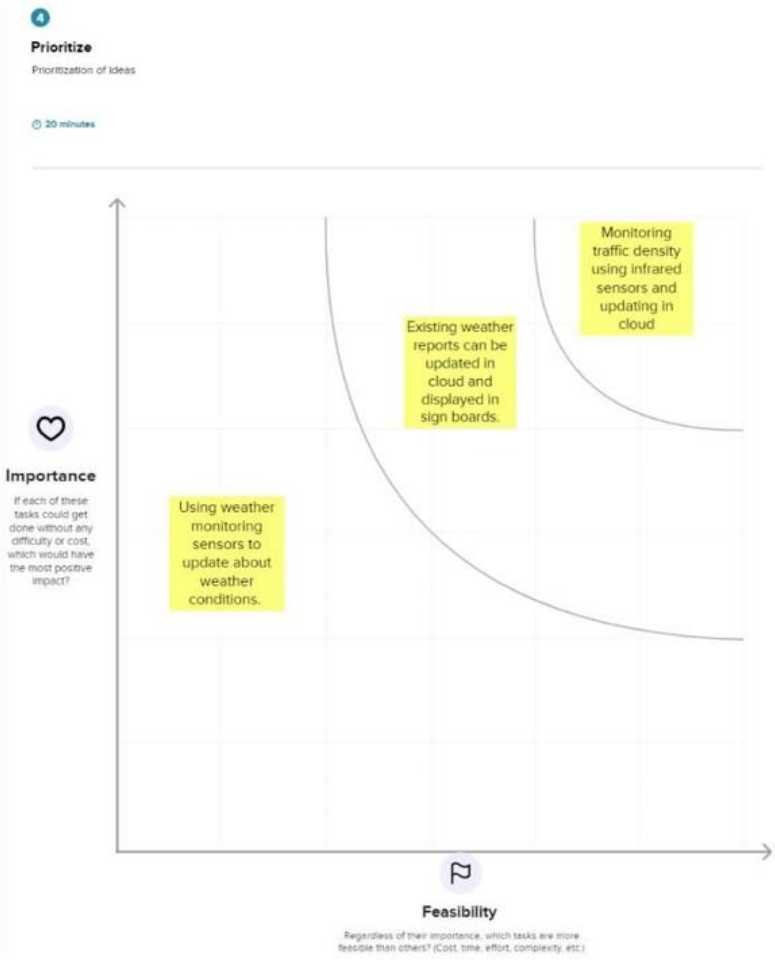


Step-2: Brainstorm, Idea Listing and Grouping





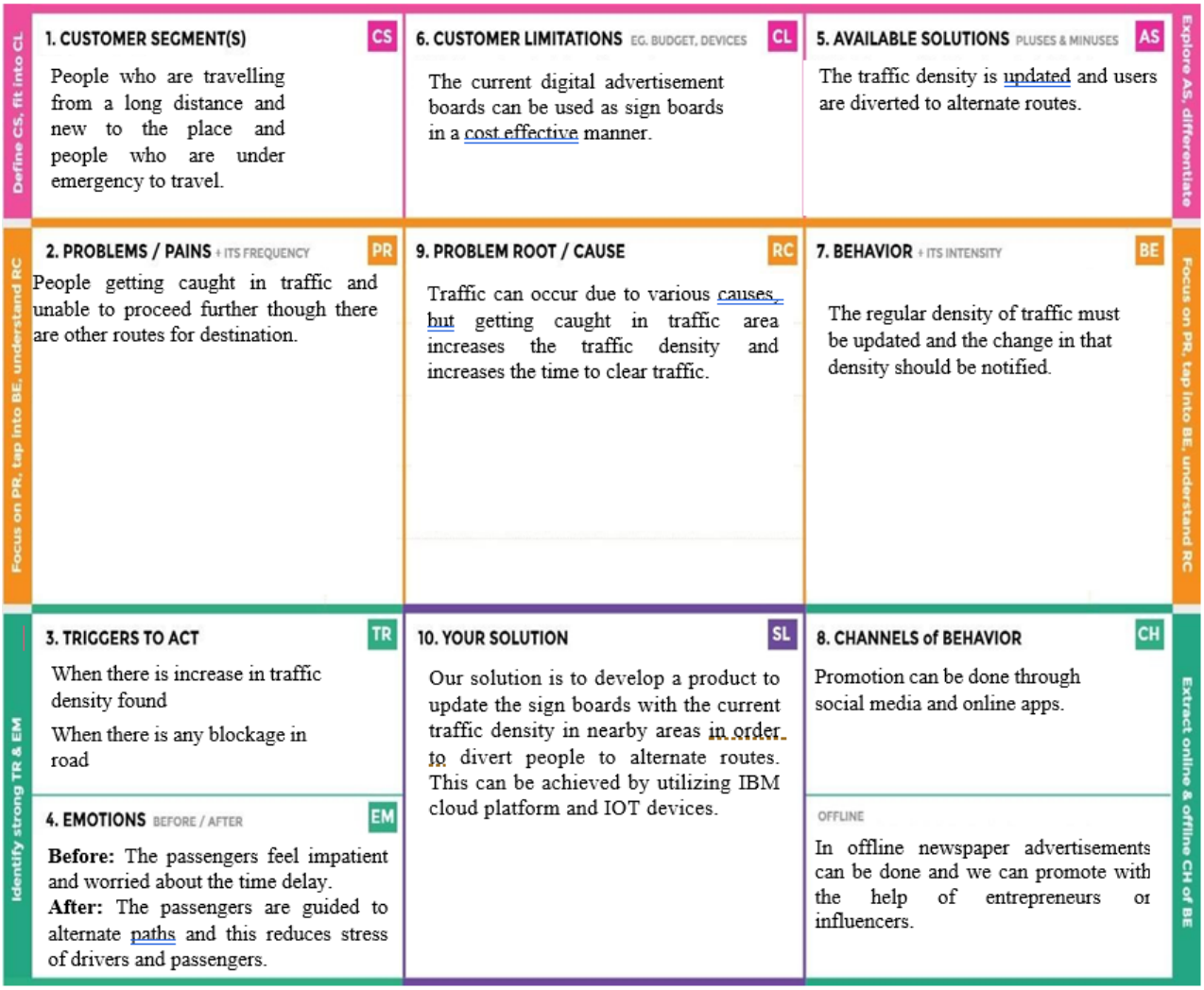
Step-3: Idea Prioritization



3.3 Proposed Solution

S.NO	Parameter	Description
1.	Problem Statement (Problem to be solved)	• Traffic density increases in unexpected manner and people are getting caught in traffic unaware about the traffic. • Also, emergency vehicles like ambulance, fire control vehicles etc. suffer due to traffic.
2.	Idea / Solution description	• The smart sign boards which indicate the traffic density in nearby junction and providing informationabout alternate route helps in avoiding further accumulation of vehicle and facilitates the drivers. • This is achieved by using IOT device, IOT cloud platform. The sensor data is updated to IOT cloud platform, based on data the sign boards are updated.
3.	Novelty / Uniqueness	• Here, instead of updating in website, the traffic information is directly displayed in sign boards, facilitating people who can't be able to access internet.
4.	Social Impact / CustomerSatisfaction	• This system is very much useful for emergency vehicles in order to avoid disasters. • This also facilitates travellers who are new to the particular area.
5.	Business Model (RevenueModel)	• In business point of view, these sign boards can also beused as advertising boards. • Though there is a huge investment, the demand for guidance and notification is high and this could encourage vendors to develop this kind of models.
6.	Scalability of the Solution	• It is expected to reduce further vehicle accumulation. • This also ensures to help in some emergency situations.

3.4 Problem Solution fit



CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional requirement

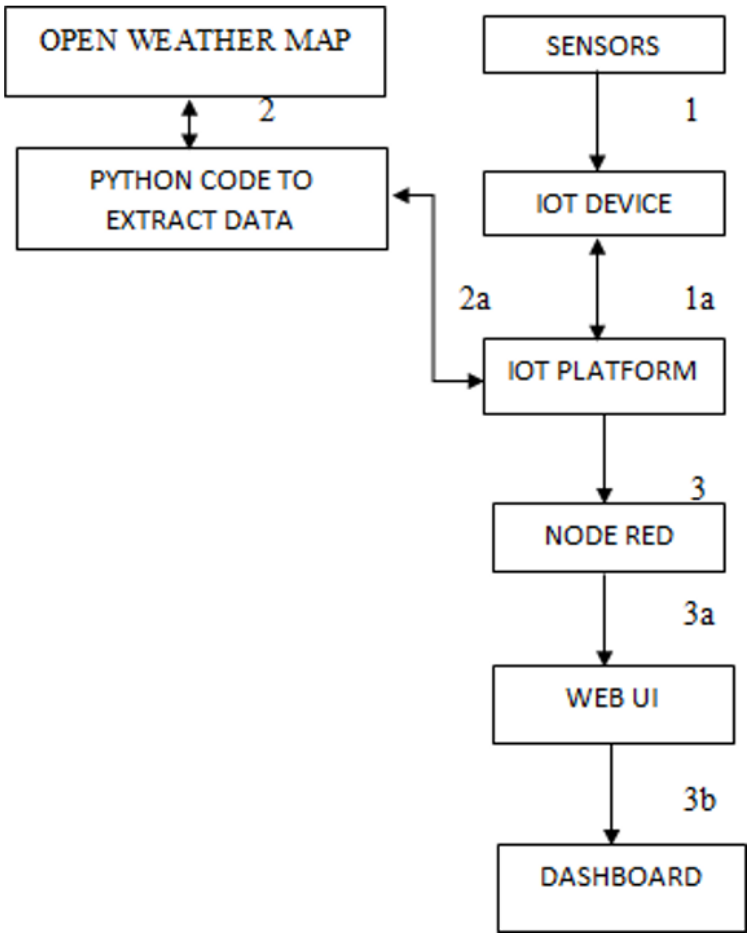
Following are the functional requirements of the proposed solution.

FR no	Functional requirement	Sub requirement
FR 1	Website registration	Registration in weather forecast website through official mail
FR 2	Login confirmation	Confirmation via email
FR 3	Sensor integration	Placing sensors in adequate locations and connecting to IOTdevice
FR 4	Parameter input	Getting input from sensors and weather forecast platform
FR 5	Processing of input	Data conversion and comparison with threshold values.
FR 6	Actuation based on processing	Actuating events based on processing of input
FR7	Results	The actuation results are displayed in dashboard

CHAPTER 5

PROJECT DESIGN

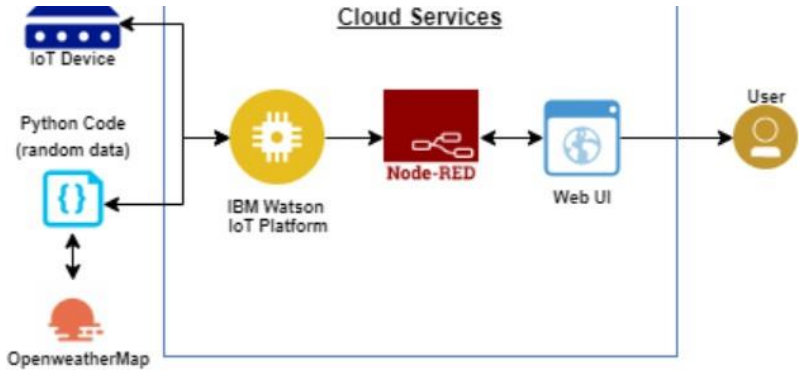
5.1 Data Flow Diagrams



- The traffic data from array of sensors are collected and given to IOT device. 1a. The sensor data is processed and given to IBM Watson platform.
- The weather information is obtained from open weather map.
  - The required weather data is extracted using python code.
- The IOT Watson platform will process the data collected and give the signs or message to be displayed.
  - Node Red will interface the IOT platform data to Web UI.
  - Wed design to display signs or message is designed in Web Ui and the output is displayed in dashboard.

5.2 Solution & Technical Architecture

Solution Architecture:



- IoT Device: This device acts as sensor which sends data to the cloud about the traffic.
- Python Code: Generates random data based on the weather data received from the open weather map platform.
- Open weather Map: This works as an API which provides global weather for any geographical location.
- IBM Watson IoT Platform: It works as a cloud service which receives data from both the IoT device and Python code and configures the Web UI based on the received data.
- Node-RED: It works as a connector between the cloud and the web UI to transfer the received sensor data to web UI.
- Web UI: Based on the received data from Node RED the required message or symbol is displayed.
- User: Based on the data the user can decide regarding the follow through action.

Technology Stack (Architecture & Stack):

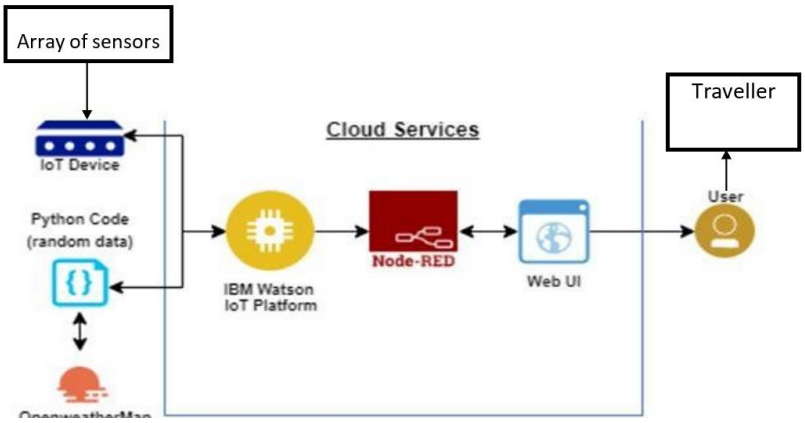


Table-1: Components & Technologies:

S.No	Component	Description	Technology
1.	Sensors	Collection of data about traffic	Ultrasonic sensors, IR sensors etc.
2.	Application logic 1	Data collection from weather forecast website	Python
3.	Application logic 2	Data processing	IBM Watson
4.	Website	Data display	Web Ui

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Creating python code to extract data, setting up IoT devices, programming IoT devices, using a dashboard to display signs	IBM Watson, NodeRed, Web UI
2.	Scalable Architecture	The application can load as many branches as there are logged in	IBM Watson, Web UI
3.	Availability	Application is offered 24/7	IBM Watson, Node Red, Web UI
4.	Performance	Frequent updation	IOT Device, IBM Watson, Node Red, Web UI

5.3 User Stories

User type	Functional requirement	User story number	User story	Acceptance criteria	Priority	Release
Customer (Traffic departme nt)	Data collection	USN-1	The user collect data fromarray of sensors.	Sensor data integrated with themodule.	High	Sprint 1
	Data processing	USN-2	The data collected is retrieved for processing.	The processed data will be passedfrom IOT Watson platform.	High	Sprint 1
	Website registration	USN-3	The user has to register an account in weather forecast website to get weather data.	New user account created for eachbranch.	High	Sprint 2
	Data collection	USN-4	The weather data isextracted from the website using python code.	The weather data integrated withthe module.	High	Sprint 2
	Data processing	USN-5	The data collected is retrieved for processing.	The processed data will be passed from IOT Watson platform.	High	Sprint 2
Customer (traveller, driver)	Data display	USN-6	Based on processed data, output signs or messagewill be displayed.	The user can see the signs or alert message in dashboard	High	Sprint 3

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

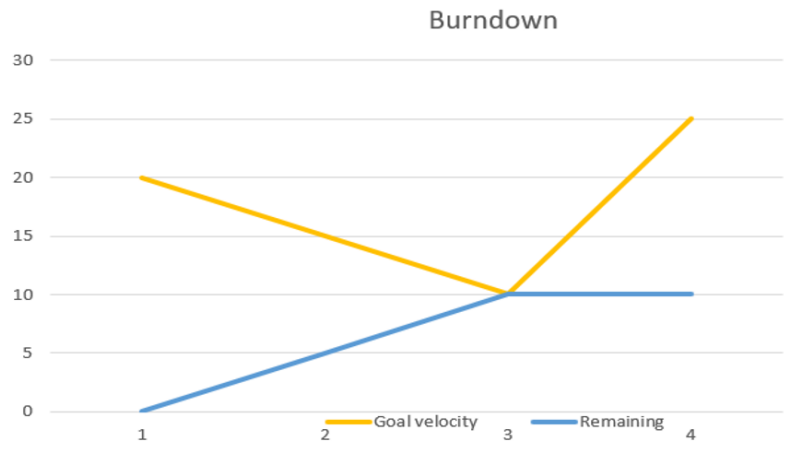
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	31 Oct 2022	20	04 Nov 2022
Sprint-2	20	6 Days	31 Oct 2022	07 Nov 2022	15	08 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	17 Nov 2022	10	18 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	25	19 Nov 2022

Velocity:

$$AV = \frac{\text{Sprint Duration}}{\text{Velocity}} = \frac{20 + 15 + 10 + 25}{4} = 17.5$$

Burndown Chart:

Sprint	Goal	Done	Goal velocity	Remaining
1	20	0	20	0
2	20	0	15	5
3	20	0	10	10
4	20	0	25	10



6.2 Sprint Delivery Schedule

Sprint	Functional requirement	User story number	User story	Priority	Team Members
Sprint 1	Data collection	USN-1	The user collect data from array ofsensors.	High	Deepak Appa Rao, Madhumitha K, Soundaryalaxmi B, Kanisha R
Sprint 1	Data processing	USN-2	The data collected is retrieved for processing.	High	Deepak Appa Rao, Madhumitha K, Soundaryalaxmi B, Kanisha R
Sprint 2	Website registration	USN-3	The user has to register an account in weather forecast website to get weather data.	High	Deepak Appa Rao, Madhumitha K, Soundaryalaxmi B, Kanisha R
Sprint 2	Data collection	USN-4	The weather data isextracted from thewebsite using python code.	High	Deepak Appa Rao, Madhumitha K, Soundaryalaxmi B, Kanisha R
Sprint 3	Data processing	USN-5	The data collected is retrieved for processing.	High	Deepak Appa Rao, Madhumitha K, Soundaryalaxmi B, Kanisha R
Sprint 4	Data display	USN-6	Based on processed data, output signs or message will be displayed.	High	Deepak Appa Rao, Madhumitha K, Soundaryalaxmi B, Kanisha R

CHAPTER 7

CODING & SOLUTIONING

7.1 Feature 1

Sprint 1:

Random data generation for temperature, humidity, nearby schools, hospitals and signs in case of diversion. The data are randomly generated and used to update the climatic conditions and also the road conditions.

Code:

```
import wiotp.sdk.device
import time
import random
import ibmiotf.application
import ibmiotf.device
import requests,json
myConfig = {
    #Configuration
    "identity":{
        "orgId":"ne15x6",
        "typeId":"ESP32",
        "deviceId":"030800"
          },
     #APIKey
     "auth":{
        "token":"26082022"
         }
        }
```

```
 def myCommandCallback(cmd):
    print("MessagereceivedfromIBMIoTPlatform:%s"%
    cmd.data['command'])m=cmd.data['command']

 client=wiotp.sdk.device.DeviceClient(config=myConfig,
 logHandlers=None)client.connect()

 #OpenWeatherMap Credentials
 BASE_URL="https://api.openweathermap.org/data/2.5/weather?"


 CITY="Coimbatore"
 URL=BASE_URL+"q="+"coimbatore"+"&appid="+"2f8de56c1b8fa2fe34b779ad4
 0c92609"

 while True:
    response=requests.get(U
    RL) if
    response.status_code==2
    00:
data=response.json()
main = data['main']
temperature = main['temp']
humidity = main['humidity']
report=data['visibility']
```

```
#messge part
msg=random.randint(0,5)
        if msg==1:
           message="SLOWDOWN,SCHOOL
           ZONE"
        elif msg==3:
           message="SLOWDOWN,HOSPITAL
           ZONE"
else:
   message=""
#Sign part
sign=random.randint(0,5)if
sign==1:
   signMsg="Right Diversion >"
elif sign==3:
   signMsg="Left Diversion <"
elif sign==5:
   signmsg="U Turn"
else:
           signMsg=""
     else:
        print("Error in the HTTP request")
     myData={'Temperature':temperature,
     'Message':message,'Sign':signMsg}
```

```
    client.publishEvent(eventId="status",msgFormat="json",data=myData,qos=0,
 onPublish=None)
    print("Published data Successfully: %s",
    myData)
    client.commandCallback=myCommandCa
    llback time.sleep(5)
 client.disconnect()
```

7.2 Feature 2

Sprint 2:

This code is used to extract temperature, humidity and Rain updates from Open weather map website.

Code:

i) weather.py

```
import requests as reqs
def get(myLocation,APIKEY):
   apiURL
="https://api.openweathermap.org/data/2.5/weather?q={myLocation}&appid={A
PIKEY}"
    responseJSON
    =(reqs.get(apiURL)).json()
    returnObject = {
      "temperature" : responseJSON['main']['temp'] - 273.15,
```

```
      "weather" :
[responseJSON['weather'][_]['main'].lower() for _ in
range(len(responseJSON['weather']))],

      "visibility" : responseJSON['visibility']/100, # visibility in percentage
where 10km is 100%and 0km is 0%

       }
    if("rain" in responseJSON):

       returnObject["rain"] = [responseJSON["rain"][key] for key in

       responseJSON["rain"]]return(returnObject)
```

ii) brain.py

```
import weather
from datetime import

datetime as dt# IMPORT

SECTION ENDS

#

# UTILITY LOGIC SECTION STARTS
 def

    processConditions(myLocation,APIKEY,

    localityInfo):weatherData =

    weather.get(myLocation,APIKEY)

  finalSpeed = localityInfo["usualSpeedLimit"]if "rain" not in
weatherData elselocalityInfo["usualSpeedLimit"]/2

    finalSpeed = finalSpeed if weatherData["visibility"]>35

    else finalSpeed/2if(localityInfo["hospitalsNearby"]):

       # hospital
       zone
       doNotHo
       nk = True
    else:
```

```
        if(localityInfo["schools"]["schoolZon
          e"]==False):# neither school nor
          hospital zone
          doNotHon
        k = False
        else:
            # school zone
            now = [dt.now().hour,dt.now().minute]
            activeTime = [list(map(int,_.split(":"))) for _ in
            localityInfo["schools"]["activeTime"]]
           doNotHonk =
activeTime[0][0]<=now[0]<=activeTime [1][0] and
activeTime[0][1]<=now[1]<=activeTime[1][1]
            return({
              "speed" :
              finalSpeed,
              "doNotHonk" :
              doNotHonk
    })
```

iii) main.py

```
# IMPORT SECTION STARTS
impo
    rt
brain
# IMPORT SECTION ENDS
#..... # USER INPUT SECTION STARTS
myLocation = "Coimbatore,IN"
APIKEY =
"bf4a8d480ee05c00952bf65b78ae826b
```

"# USER INPUT SECTION ENDS

.....

MICRO-CONTROLLER CODE STARTS

print(brain.processConditions(myLocation,APIKEY

,localityInfo))

'''

MICRO CONTROLLER CODE WILL BE

ADDED IN SPRINT 2 AS PER OUR PLANNED

SPRINT SCHEDULE

'''

7.3 Feature 3

Sprint 4

This code explains how ultrasonic sensor is interfaced with ESP32 and connected to IBM Watson IOT platform.

Code:

```
#include <WiFi.h>
#include<PubSubClient.h>
void callback(char* subscribetopic,byte* payload, usigned int payloadLength);
//IBM credentials//
#define ORG "ne15x6"
#define DEVICE_TYPE "MP"
#define DEVICE_ID "0331"
#define TOKEN "03082000"
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback ,wifiClient);
//Main code//
#define ECHO_PIN 2
```

```
#define TRIG_PIN 3

void setup() {
  Serial.begin(115200);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

float readDistanceCM() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  int duration = pulseIn(ECHO_PIN, HIGH);
  return duration * 0.034 / 2;
}

void loop()
{
  float distance = readDistanceCM();
  if(distance<25)
  {
    Serial.println("Dense Traffic");
  }
  else
  {
    Serial.println("Clear Traffic");
  }
  delay(100);
}
```

CHAPTER 8

TESTING

8.1 Test Cases

This table shows the number of test cases that have passed, failed, and untested.

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	0	5
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Final Report Output	4	0	0	4

8.2 User Acceptance Testing

Defect Analysis:

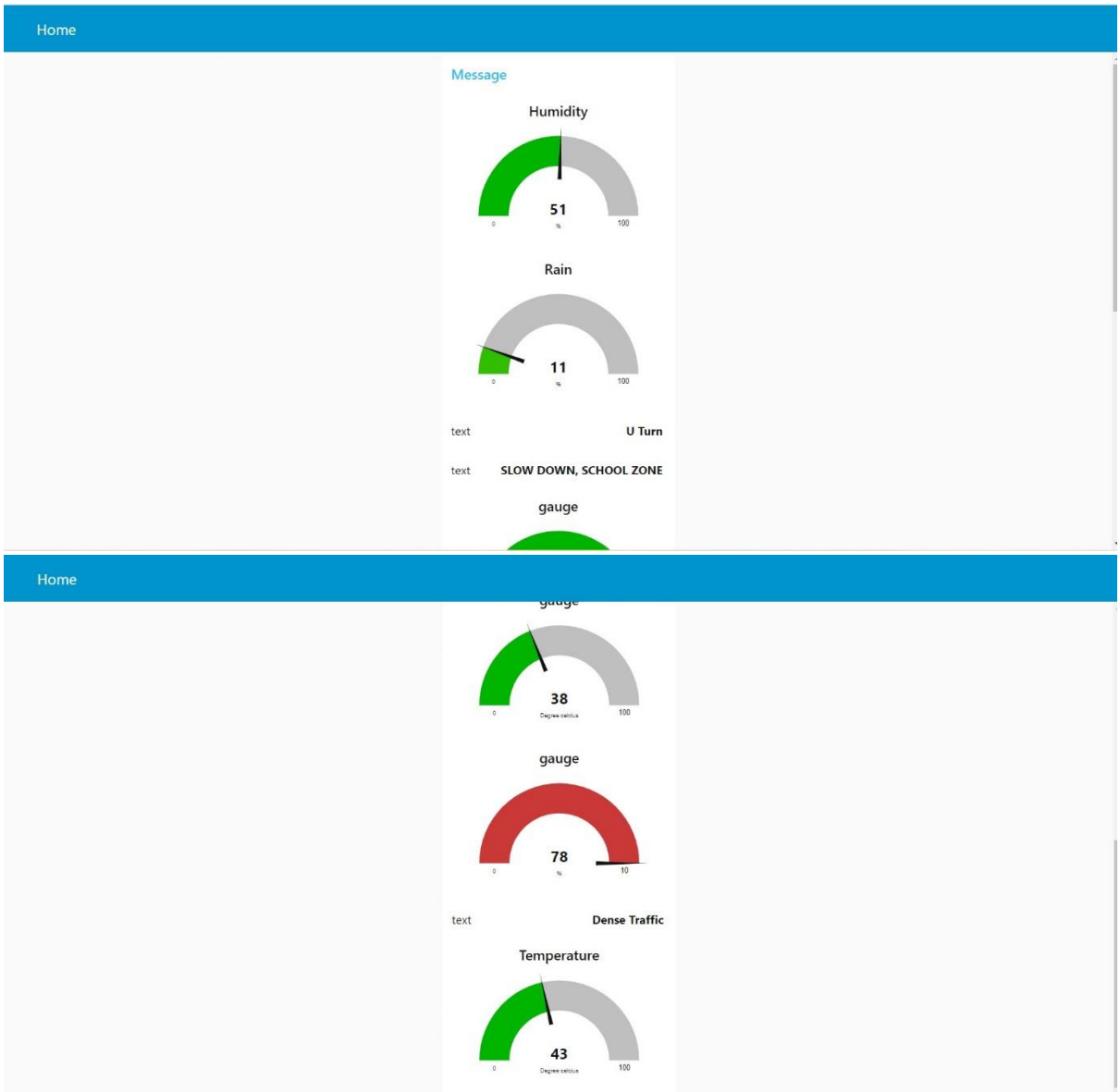
This table shows the number of resolved or closed bugs at each severity level, and how they were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Subtotal
By Design	5	4	2	11
Duplicate	1	0	3	4
External	1	1	0	2
Fixed	5	2	1	8
Not Reproduced	0	0	1	1
Skipped	0	0	1	1
Won't Fix	0	2	2	4
Totals	12	9	10	31

CHAPTER 9

RESULTS

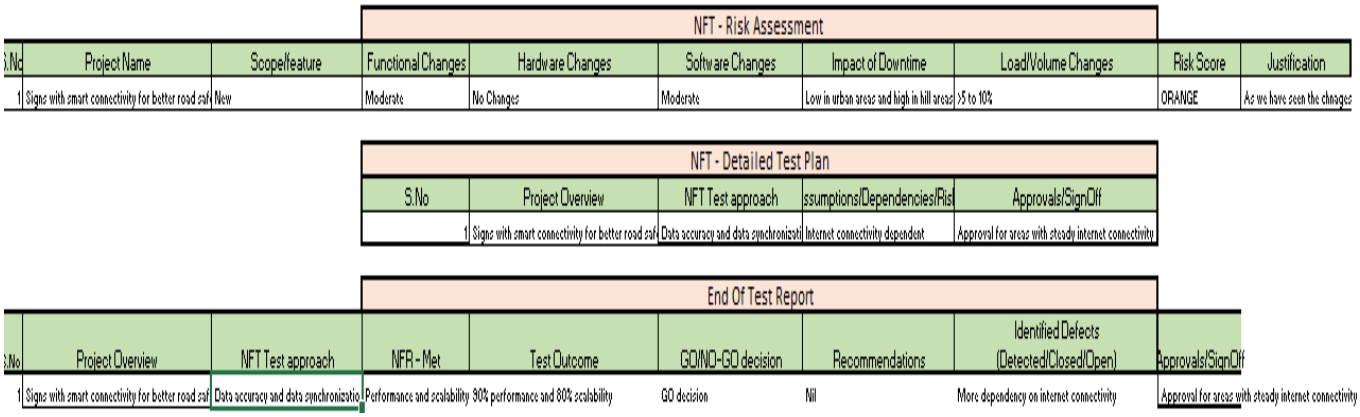
9.1 Web UI Results:



The final results from Node RED web UI page is shown above. The temperature, humidity and rain conditions along with school and hospitals zones are displayed. A random condition is considered and diversions for that condition is also displayed. Also, the traffic density data is also displayed.

9.2 Performance Metrics

The final output will be temperature, humidity, sign diversions and schools and hospital locations and traffic density. The performance metrics is good in terms of urban areas and poor in terms of rural areas where network connectivity is poor.



NFT - Risk Assessment

S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score	Justification
1	Signs with smart connectivity for better road saf	New	Moderate	No Changes	Moderate	Low in urban areas and high in hill areas	>5 to 10%	ORANGE	As we have seen the changes

NFT - Detailed Test Plan

S.No	Project Overview	NFT Test approach	ssumptions/Dependencies/Risk	Approvals/SignOff
1	Signs with smart connectivity for better road saf	Data accuracy and data synchronizati	Internet connectivity dependent	Approval for areas with steady internet connectivity

End Of Test Report

S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1	Signs with smart connectivity for better road saf	Data accuracy and data synchronizatio	Performance and scalability	90% performance and 80% scalability	GO decision	Nil	More dependency on internet connectivity	Approval for areas with steady internet connectivity

CHAPTER 10

ADVANTAGES & DISADVANTAGES

Advantages:

- Updating traffic density help drivers to take alternate routes.
- Updating Road conditions and weather condition help to alert drivers for safe journey.
- Updating traffic density help emergency vehicles to take alternate routes and reach destination earlier.

Disadvantages:

- The performance of this project will be good in urban areas with stable internet connectivity but will be poor in rural areas with poor internet connectivity.
- The traffic density data may not be accurate since the distance only measured.
- The digital sign boards may not be feasible in all locations.

CHAPTER 11

CONCLUSION

Traffic congestion has a negative impact on people's employment, education, and personal lives, as well as the country's progress. There are multiple issues that people must face owing to the increasing problems of Traffic Jams. Hence, use of sign boards helps the passengers stay updated about the traffic status of the road which saves them from any emergency situation at the right moment, reduces the impact of pollution in the environment, and provides an overall improvement in the quality of living.

CHAPTER 12

FUTURE SCOPE

- Cameras can be used in place of ultrasonic sensors to measure traffic density.
- The road conditions can be accurately updated by interfacing traffic department database to the cloud.
- Diversion routes and maps can be included for display.

CHAPTER 13

APPENDIX

Source Code

Sprint 1:

```
import wiotp.sdk.device
import time
import random
import ibmiotf.application
import ibmiotf.device
import requests,json
myConfig = {
    #Configuration
    "identity":{
        "orgId":"ne15x6",
        "typeId":"ESP32",
        "deviceId":"030800"
          },
     #APIKey
     "auth":{
        "token":"26082022"
         }
       }
def myCommandCallback(cmd):
print("MessagereceivedfromIBMIoTPlatform:%s"%
cmd.data['command'])m=cmd.data['command']

client=wiotp.sdk.device.DeviceClient(config=myConfig,
logHandlers=None)client.connect()
```

```
#OpenWeatherMap Credentials
BASE_URL="https://api.openweathermap.org/data/2.5/we
ather?"
CITY="Coimbatore"
URL=BASE_URL+"q="+"coimbatore"+"&appid="+"2f8de56c1b8fa2fe34
b779ad40c92609"
while True:
response=requests.g
et(URL) if
response.status_cod
e==200:
data=response
.json()main =
data['main']
temperature =
main['temp']
humidity =
main['humidity']
report=data['visibilit
y'] #messge part
msg=random.randin
t(0,5)
if msg==1:
message="SLOWDOWN,SCHOOL
ZONE"
elif msg==3:
```

```
message="SLOWDOWN,HOSPITAL
ZONE"
else:
message
=""
#Sign
part
sign=random.rand
int(0,5)if sign==1:
signMsg="Right
Diversion >"elif
sign==3:
signMsg="Left
Diversion <"elif
sign==5:
signmsg="U
Turn"else:
signMsg=""
else:
print("Error in the HTTP request")
myData={'Temperature':temperature,
'Message':message,'Sign':signMsg}
client.publishEvent(eventId="status",msgFormat="json",data=myData,qos
=0, onPublish=None)
print("Published data Successfully:
%s", myData)
```

client.commandCallback=myComm

andCallback time.sleep(5)

client.disconnect()

Sprint 2:

i) weather.py

import requests as reqs

def get(myLocation,APIKEY):
apiURL
="https://api.openweathermap.org/data/2.5/weather?q={myLocation}&a
ppid={APIKEY}"

responseJSON

=(reqs.get(apiURL)).json()

returnObject = {

"temperature" : responseJSON['main']['temp'] - 273.15,
"weather" :
[responseJSON['weather'][_]['main'].lower() for _
inrange(len(responseJSON['weather']))],

"visibility" : responseJSON['visibility']/100, # visibility in percentage
where 10km is 100%and 0km is 0%

}
if("rain" in responseJSON):

returnObject["rain"] = [responseJSON["rain"][key] for key in

responseJSON["rain"]]return(returnObject)

ii) brain.py

import weather
from datetime import

datetime as dt#

IMPORT SECTION

ENDS

#

```
# UTILITY LOGIC SECTION STARTS
def

processConditions(myLocation,APIKEY,l

ocalityInfo):weatherData =

weather.get(myLocation,APIKEY)

finalSpeed = localityInfo["usualSpeedLimit"]if "rain" not in
weatherData elselocalityInfo["usualSpeedLimit"]/2

finalSpeed = finalSpeed if weatherData["visibility"]>35

else finalSpeed/2if(localityInfo["hospitalsNearby"]):

# hospital

zone

doNotHonk

= True

else:

if(localityInfo["schools"]["schoolZone"

]==False):# neither school nor hospital

zone

doNotHonk =

Falseelse:

# school zone
now = [dt.now().hour,dt.now().minute]
activeTime = [list(map(int,_.split(":"))) for _ in
localityInfo["schools"]["activeTime"]]

doNotHonk = activeTime[0][0]<=now[0]<=activeTime
[1][0] andactiveTime[0][1]<=now[1]<=activeTime[1][1]

return({
"speed" : finalSpeed,

"doNotHonk" :

doNotHonk
```

```
})
```

iii) main.py

```
# IMPORT SECTION STARTS
im
po
 rt
 br
 ai
  n
# IMPORT SECTION ENDS
#..... # USER INPUT SECTION STARTS
myLocation = "Coimbatore,IN"
APIKEY =
"bf4a8d480ee05c00952bf65b78ae82
6b"# USER INPUT SECTION
ENDS
# .....
# MICRO-CONTROLLER CODE STARTS
print(brain.processConditions(myLocation,APIK
EY,localityInfo))
'''
MICRO CONTROLLER CODE WILL BE
ADDED IN SPRINT 2 AS PER OUR
PLANNED SPRINT SCHEDULE
'''
```

Sprint 4:

```
#include <WiFi.h>
#include<PubSubClient.h>
void callback(char* subscribetopic,byte* payload, usigned int payloadLength);
//IBM credentials//
#define ORG "ne15x6"
#define DEVICE_TYPE "MP"
#define DEVICE_ID "0331"
#define TOKEN "03082000"
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback ,wifiClient);
//Main code//
#define ECHO_PIN 2
#define TRIG_PIN 3

void setup() {
  Serial.begin(115200);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

float readDistanceCM() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  int duration = pulseIn(ECHO_PIN, HIGH);
  return duration * 0.034 / 2;
}

void loop()
{
  float distance = readDistanceCM();
  if(distance<25)
  {
```

```
  Serial.println("Dense Traffic");
 }
 else
 {
  Serial.println("Clear Traffic");
 }
 delay(100);
}
```

GitHub Link	https://github.com/IBM-EPBL/IBM-Project-13658-1659525186
Video presentation link	https://youtu.be/5-HmFBOS6dM