

# **SIGNS WITH SMART CONNECTIVITY FOR BETTER ROAD SAFETY**

**NALAIYA THIRAN PROJECT BASED LEARNING  
ON  
PROFESSIONAL READINESS FOR  
INNOVATION, EMPLOYABILITY AND  
ENTREPRENEURSHIP**

## **A PROJECT REPORT**

AJITH RAJA. M	912019104004
JEROME IDHAYA MICHAEL. J	912019104013
NAVEEN KUMAR. G	912019104022
SURYAPRAKASH. R	912019104028

## **INDUSTRY MENTOR**

VANATHI.D

**BACHELOR OF ENGINEERING  
IN  
COMPUTER SCIENCE AND ENGINEERING**

**PANDIAN SARASWATHI YADAV ENGINEERING COLLEGE**

An ISO Certified Institution

Approved by AICTE, New Delhi

&

Affiliated by Anna University, Chennai

NOVEMBER-2022

# **ABSTRACT**

In present Systems the road signs and the speed limits are Static. But the road signs can be changed in some cases. We can consider some cases when there are some road diversions due to heavy traffic or due to accidents then we can change the road signs accordingly if they are digitalized .This project proposes a system which has digital sign boards on which the signs can be changed dynamically. If there is rainfall then the roads will be slippery and the speed limit would be decreased. There is a web app through which you can enter the data of the road diversions, accident prone areas and the information sign boards can be entered through web app. This data is retrieved and displayed on the sign boards accordingly.

## TABLE OF CONTENTS

Chapter	Topic	Page.No
1.	<b>INTRODUCTION</b>	<b>6</b>
	Project Overview	6
	Purpose	6
2.	<b>LITERATURE SURVEY</b>	<b>7</b>
	Existing problem	7
	References .	9
	Problem Statement Definition	9
3.	<b>IDEATION &amp; PROPOSED SOLUTION</b>	<b>10</b>
	Empathy Map Canvas	10
	Ideation & Brainstorming	10
	Proposed Solution	11
4.	<b>REQUIREMENT ANALYSIS</b>	<b>12</b>
	Functional requirement	12
	Non-Functional requirements	12
5.	<b>PROJECT DESIGN</b>	<b>13</b>
	Data Flow Diagrams	13
	Solution & Technical Architecture	13
	User Stories	14
6.	<b>PROJECT PLANNING &amp; SCHEDULING</b>	<b>15</b>
	Sprint Planning & Estimation	15
	Sprint Delivery Schedule	16
	Reports from JIRA	18
7.	<b>CODING &amp; SOLUTIONING (Explain the features added in the project along with code)</b>	<b>19</b>
	Feature 1	19
	Feature 2	20

<b>8.</b>	<b>TESTING</b>	<b>21</b>
	Test Cases	21
	User Acceptance Testing	31
<b>9.</b>	<b>ADVANTAGES</b>	<b>32</b>
<b>10.</b>	<b>CONCLUSION</b>	<b>32</b>
<b>11.</b>	<b>FUTURE SCOPE</b>	<b>33</b>
<b>12.</b>	<b>APPENDIX</b>	<b>34</b>
	Source Code	42
	GitHub & Project Demo Link	43

## CHAPTER – 1

### INTRODUCTION

In its Global Status Report on Road Safety, the World Health Organization (WHO) noted that the worldwide total number of road traffic deaths has plateaued at million per year, with tens of million either injured or disabled . Different initiatives, such as the United Nations’ initiative for the Decade of Action for Road Safety, have led to improvements in road safety policies and enforcements. However, the WHO notes that the progress has been slow and has maintained the call for urgent action to reduce these figures

WHO describes different measures that can be implemented with minimal economic impacts in its “Save LIVES: Road Safety Technical Package”. A cornerstone of these steps is realizing economic systems for “monitoring road safety by strengthening data systems”. Meanwhile, a key theme in the package is motivating the adoption of a Safe System approach, which is a holistic approach to road safety that parts from traditional management solutions by emphasizing safety-by-design.

### PROJECT OVERVIEW

- To replace the static signboards, smart connected sign boards are used.
- These smart connected sign boards get the speed limitations from a web app using weather API and update automatically.
- Based on the weather changes the speed may increase or decrease.
- Based on the traffic and fatal situations the diversion signs are displayed.
- Guide(Schools), Warning and Service(Hospitals, Restaurant) signs are also displayed accordingly.
- Different modes of operations can be selected with the help of buttons.

### PURPOSE

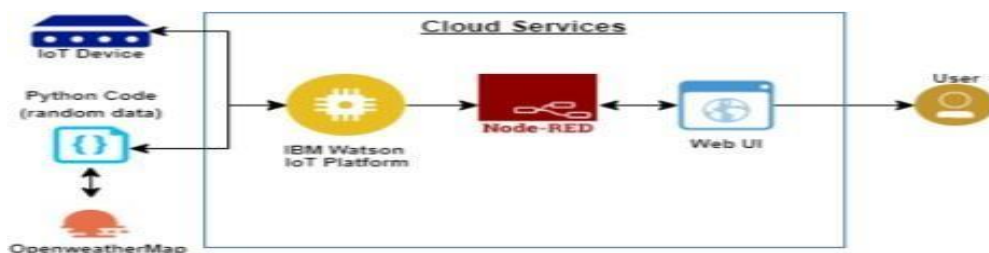


Fig : 1.2 - Architecture

## **CHAPTER -2**

### **LITERATURE SURVEY**

#### **EXISTING PROBLEM**

An IoT Architecture for Assessing Road Safety in Smart Cities (2018)  
The Safe System (SS) approach to road safety emphasizes safety-by-design through ensuring safe vehicles, road networks, and road users.

With a strong motivation from the World Health Organization (WHO),

this approach is increasingly adopted worldwide. Considerations in SS, however, are made for the medium-to-long term. Our interest in this work is to complement the approach with a short-to-medium term dynamic assessment of road safety. Toward this end, we introduce a novel, cost-effective Internet of Things (IoT) architecture that facilitates the realization of a robust and dynamic computational core in assessing the safety of a road network and its elements. In doing so, we introduce a new, meaningful, and scalable metric for assessing road safety. We also showcase the use of machine learning in the design of the metric computation core through a novel application of Hidden Markov Models (HMMs). Finally, the impact of the proposed architecture is demonstrated through an application to safety-based route planning. An IoT Architecture for Assessing the Safety of a Dynamic Road Transport System Assessment Elements.

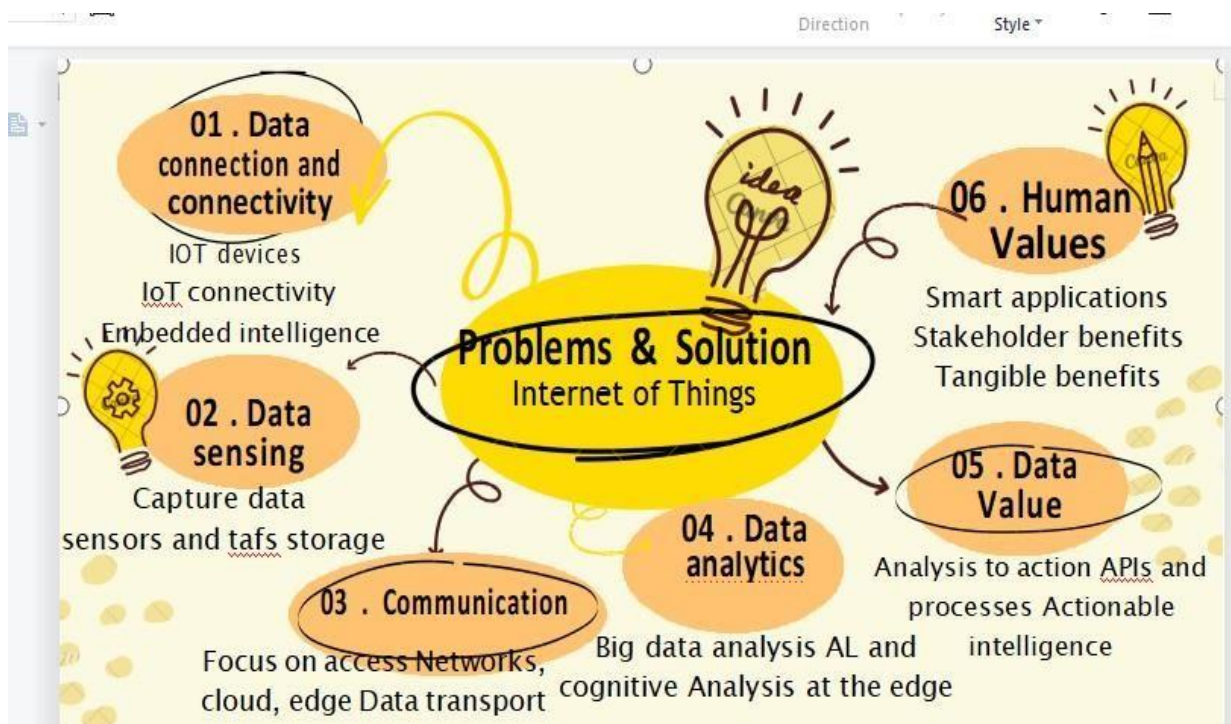
The way the SS approach comprises the three elements of safe vehicle, safe road, and safe driver facilitates a hierarchical safety assessment approach whereby the safety of the individual elements can provide a collective indicator of safety for the road network, as illustrated in Figure 3. In turn, this indicator can be concatenated from the assessment of individual road segments,

to routes, to the road network. It is possible to consider a meaningful safety metric based on the live (or real-time) status of the road. For example, the safety level of a certain segment/road depends on the aggregate safety of vehicles currently traversing it, combined with the number of potholes and/or the wetness or how slippery is the road, in addition to safety/alertness of the drivers on the road. In designing our architecture, we exploit three important dependencies. The first is between the SS elements, e.g., how well a car can handle a certain road, or how some drivers exhibit safer behavior in instances of higher visibility. The second dependency is in between consecutive segment/roads, especially in terms of traversing vehicles and drivers. The third dependency is like the second but is established in time. Abrupt changes in safety levels can thus be viewed as an anomaly (outlier) or inferred as indicator to a substantial change in the road context. Safety-Based Route Planning Route planning has become widely used in both personal and commercial use, resulting in an increasing dependence on its reliability. Various applications employ efficient algorithms for route planning [43]. Trip time and cost, e.g., for tolls, have been the typical metrics for route planning applications, but other metrics, however, have been utilized, e.g., for fuel emission/consumption or energy requirements of electric vehicles.

## Reference

1. **R. Bhandari, R. Bhaskaran, and N. Venkata**, “Full Stop: Tracking unsafe stopping behavior of buses,” in Proceedings of the In Conference on Communication Systems Networks, pp. 65–72, 2018.
2. **N. Arbabzadeh and M. Jafari**, “A Data-Driven Approach for Driving Safety Risk Prediction Using Driver Behavior and Roadway Information Data,” IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 2, pp. 446–460, 2018
3. **. H . L . C h u , V . R a m a n , J . S h e n , R . R o y C h o u d h u r y , A . K a s a l , a n V . B a h l**, “Poster: You driving? Talk to you later,” in Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys’11 and Co- located Workshops, USA, July

## 2.3 Problem statement definition





# CHAPTER -3

## IDEATION & PROPOSED SOLUTION

### Empathy Map Canvas

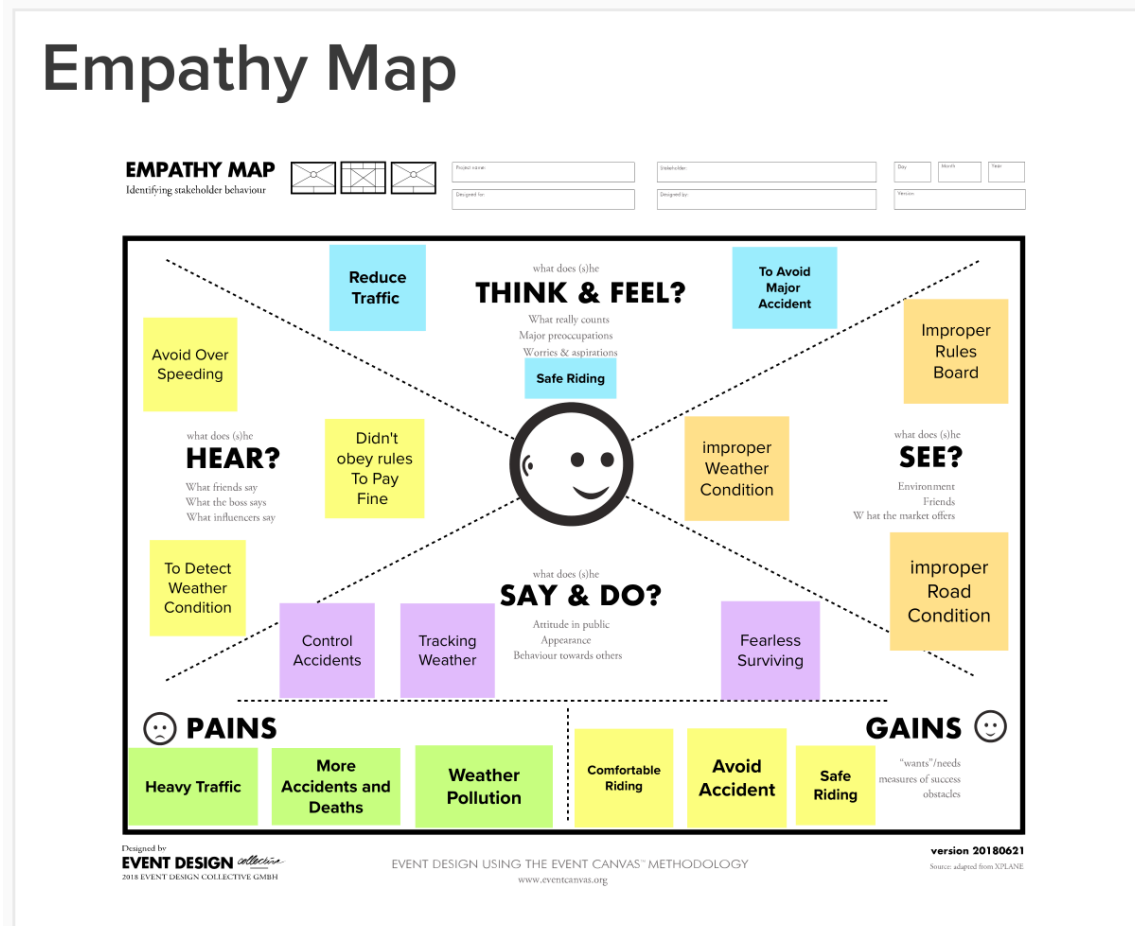


Fig : 3.1- Empathy Map Canvas

## Ideation & Brainstorming

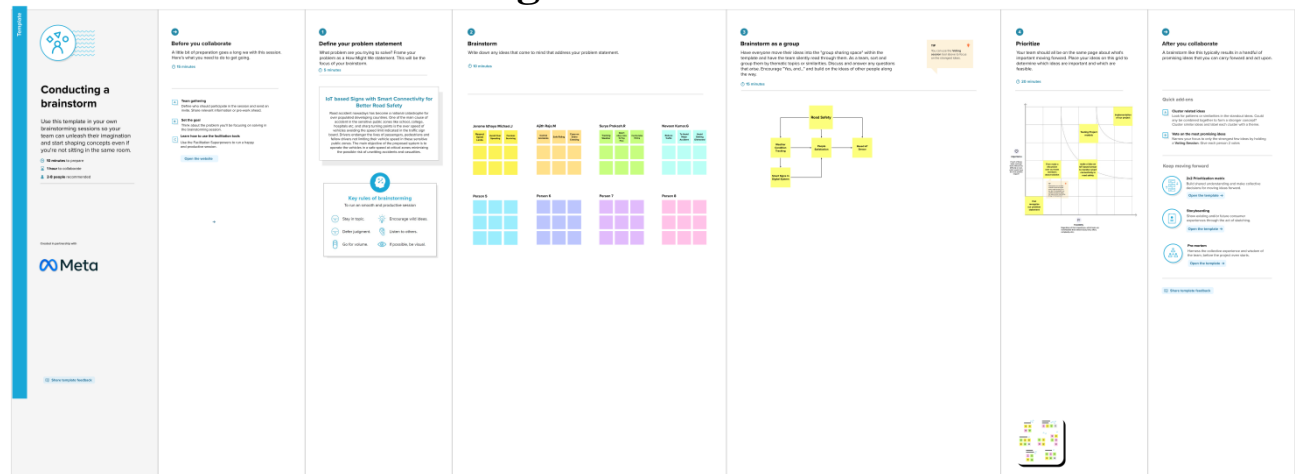


Fig : 3.2-Ideation & Brainstorming

## PROPOSED SOLUTION

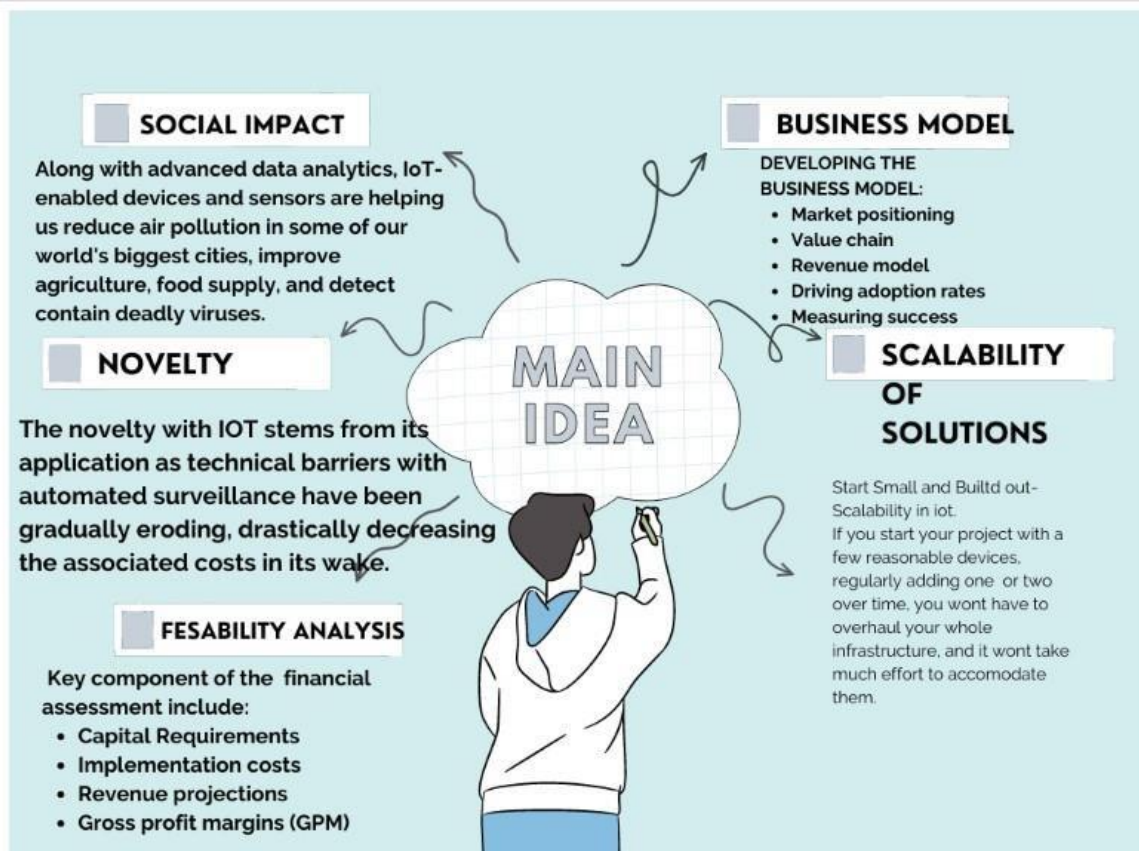


Fig : 3.3- PROPOSED SOLUTION

## CHAPTER -4

# REQUIREMENT ANALYSIS

## Functional Requirements

Functional Requirements: Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User tracking	Tacking through driving behavior. Tracking through digital process.
FR-2	Weather	Using open weather map
FR-3	Application programming interface	Open API keys.
FR-4	Sensor	Stand-alone-safety sensor GPS sensor

Table : 4.1

## Non - Functional Requirements

Non-functional Requirements: Following are the non-functional requirements of the proposed solution

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Specifies how systems should operate for the customer/end-user. i.e., how many clicks to get to certain place?
NFR-2	Reliability	Defines the systems availability and the tolerance for failure. i.e., what's the target uptime?
NFR-3	Performance	Focuses on the systems speed, efficiency, and workload. i.e., how fast does the system respond?
NFR-4	Availability	It is a metric that measures the probability that systems not failed or undergoing a repair action when it needs to be used.
NFR-5	Scalability	Ensures the system can respond to changes in demand. i.e., how will the system pull on additional resources?
NFR-6	Security	Focuses on how the system is kept secure, store data, and responds to attacks. i.e., What are the security protocols of the site?

## CHAPTER - 5

### PROJECT DESIGN

#### Data Flow Diagrams

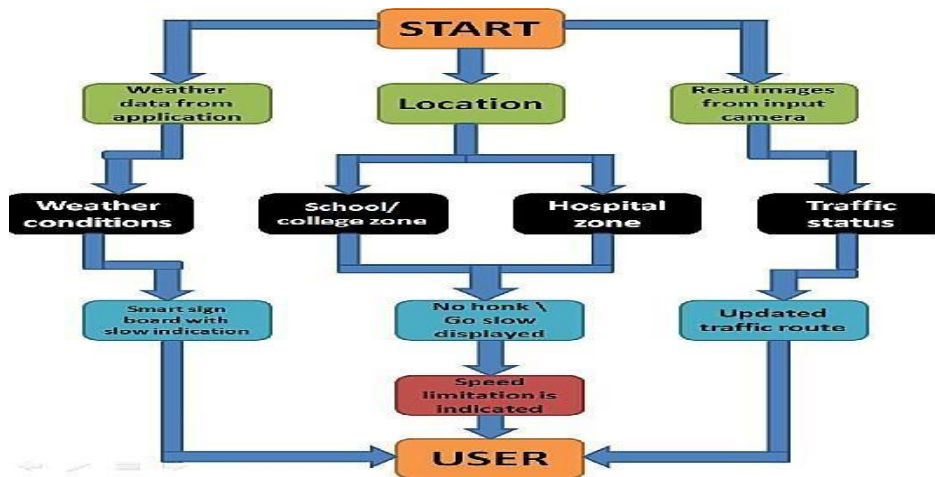


Fig : 5.1-Data Flow Diagrams

#### Solution & Technical Architecture



Fig : 5.2-Solution & Technical Architecture

## User Stories

### What is an IoT architecture?

An IoT architecture is a mix of hardware and software components that interact together to make up a smart cyber-digital system. Interoperating with one another, these components make up a base for an IoT solution to be built upon. Before we dive into the details, let's get things straight: there's no one-size-fits-all approach to designing an IoT architecture. Still, the basic layout stays largely the same no matter the solution.

### A standard IoT architecture: what's under the hood?

Common data-driven IoT applications rely on a standard IoT architecture spanning four layers:

- Device layer
- Network layer
- Service and application support layer
- Application layer

Recently, however, more and more connected systems have started shifting focus toward edge processing, which has led to an additional layer being added to a traditional four-tier architecture. The share of activities performed at the edge depends on a particular implementation but it commonly spans enabling connectivity, as well as filtering, aggregating, securing, and processing the incoming data.

## CHAPTER - 6

### PROJECT PLANNING & SCHEDULING

#### Sprint Planning & Estimation

##### **MILESTONE:**

##### **IBM Cloud Services: (Aug 22-Sep02 )**

Among all the IOT product development stages cloud services is an important stage for building the best IOT product. The development team is responsible for building web and mobile based applications for control in the functionality of products in real time.

##### **Open Weather Map :(Sep 05-Sep 10)**

The Open Weather Map is a service that provides weather data, including current weather data, forecasts, and historical data to the developers of webservices and mobile applications. We analyzed the behavior of the metrics for the open weather map model.

##### **Node-Red:(Oct 1-Oct 11)**

Node-Red is a programming tool for wiring together hardware devices, API and online services in new and interesting ways. It provides a browser-Based editor that makes it easy to wire together flows using the wider angel of nodes in the palette that can be deployed to its run time in a single -click.

##### **Python Script:(Sep 20-Sep 27)**

The primary objective of running python on an IOT device that popsupinmind is grabbing the Arduino UNO from the table. Python is pre-install edit the operating system, and the only objective left for us is to write the coding script.

##### **Sensor :(Sep 10-Sep 17)**

The Navigational sensor provides a precise geo-spatial orientation of the vehicle as well as trends in driving behavior. The ODAWS algorithm is used to interpret sensor data and offer real -time notifications to the driver, boosting road safety.

## Product Hardware Identification:

Product Hardware Identification is one of the most important parts of IoT product development stages. The development team with great and in-depth knowledge of diverse types of IoT boards, sensors and connector devices will get a huge success in IoT product development.

## Application:

A traffic signal is used as an instructing device that indicates the road user to act according to the displayed sign. Sensors installed in strategic locations can use IoT technology to collect data on congestion, moving vehicles away from these locations. IoT Big Data solution can analyze this information, determine alternative routes, and improve traffic signaling to reduce congestion.

## Final Deliverables : ( Oct 25-Nov 15)

Our project Signs with smart connectivity for better road safety in the domain of internet of things (IOT) will soon prove its potential in vehicle maintenance, navigation, monitoring leading to improve transportation on the given sprint delivery plan by using our followed task and assignments like Arduino UNO, IBM cloud services, Open weather map, Node-red, Python IDLE, sensor.

## Sprint Delivery Schedule

### Product Backlog, Sprint Schedule, and Estimation (4 Marks)

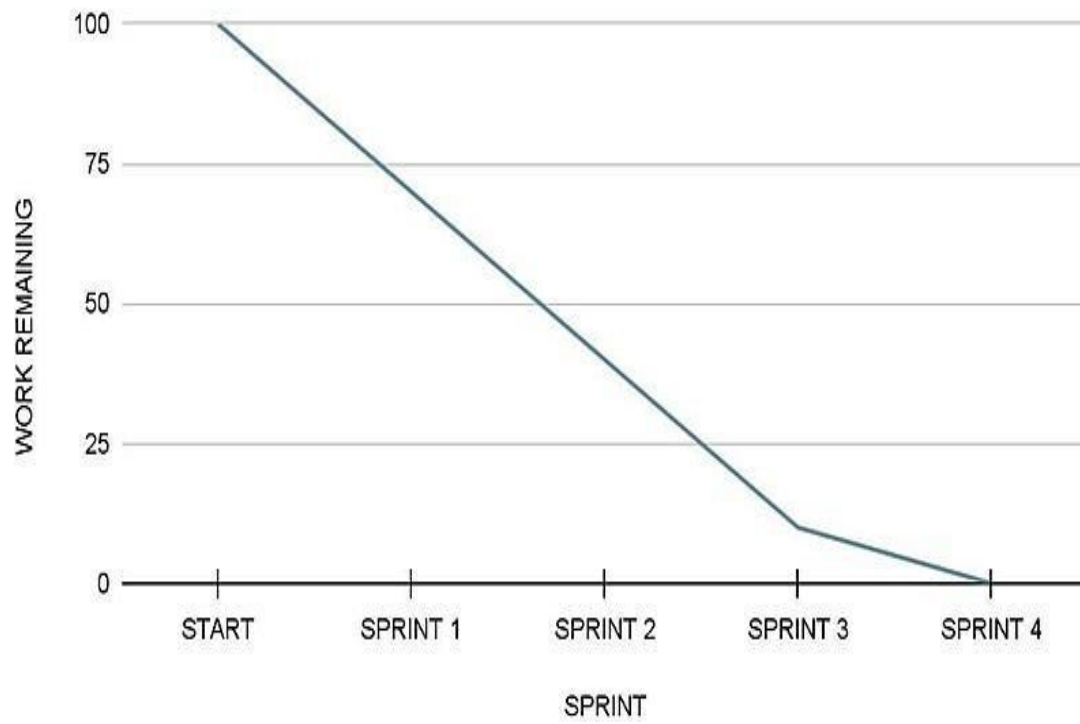
Sprint	Functional Requirement	User Story Number	User Story /Task	Story Points	Priority	Team Members
Sprint-1	Resources Initialization	USN-1	Create and initialize accounts in various public APIs like Open Weather API.	1	Low	Ajith Raja.M Jerome Idhaya Michael. J Naveen Kumar. G Suryaprakash. R

Sprint-1	Local Server/Software Run	USN-2	Write a Python program that outputs results given the inputs like weather and location	1	Medium	Ajith Raja.M Jerome Idhaya Michael. J Naveen Kumar. G Suryaprakash. R
Sprint-2	Push the server/software to cloud	USN-3	Push the code from Sprint 1 to cloud so it can be accessed from anywhere	2	Medium	Ajith Raja.M Jerome Idhaya Michael. J Naveen Kumar. G Suryaprakash. R
Sprint-3	Hardware initialization	USN-4	Integrate the hardware to be able to access the cloud functions and provide inputs to the same.	2	High	Ajith Raja.M Jerome Idhaya Michael. J Naveen Kumar. G Suryaprakash. R
Sprint-4	UI/UX Optimization	USN-5	Optimize all the shortcomings and provide	2	Medium	Ajith Raja.M Jerome Idhaya Michael. J Naveen Kumar. G Suryaprakash. R



## Reports From JIRA

### Balance Work



# CHAPTER - 7

## CODING & SOLUTIONING

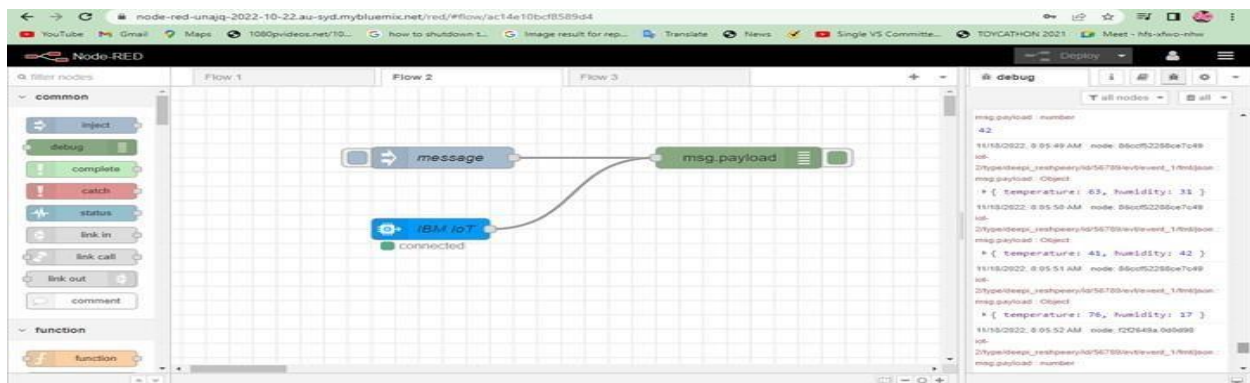
### Feature - 1

#### i) Python

```
new 1.py - H:\new\new 1.py (3.11.0)
File Edit Format Run Options Window Help

import requests
api_data = "https://api.openweathermap.org/data/2.5/weather?lat=9.9333&lon=78.1167&appid=5b470269921e324575ff30d836ad5aef"
rec=requests.get(url=api_data)
data= rec.json()
print(data)
temp = data['main']['temp']
print("\nTemperature is : ", temp)
humidity = data['main']['humidity']
print("Humidity is : ", humidity)
```

#### ii) NODE-RED



#### OUTPUT :

2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 63, humidity: 17 }

11/18/2022, 8:21:57 AMnode: 86ccf52288ce7c49iot-2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 46, humidity: 18 }

11/18/2022, 8:21:57 AMnode: 86ccf52288ce7c49iot-2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 34, humidity: 27 }

11/18/2022, 8:21:57 AMnode: 86ccf52288ce7c49iot-2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 83, humidity: 18 }

11/18/2022, 8:21:57 AMnode: 86ccf52288ce7c49iot-2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 97, humidity: 34 }

11/18/2022, 8:23:15 AMnode: 86ccf52288ce7c49iot-2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 61, humidity: 5 }

## Feature 2

### WOKWI

In this Wokwi platform the circuit is designed to determined the measure of temperature and humidity by the usage component of DHT22

The screenshot shows the Wokwi platform interface. On the left, the sketch code for an ESP32 is displayed, which includes the DHT22 library and publishes temperature and humidity data to an MQTT topic. On the right, the simulation window shows the circuit with an ESP32 and a DHT22 sensor. The sensor's output is displayed as Temperature: 30.0°C and Humidity: 44.0%.

```

1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 #include "DHT.h"
4 #define DHTPIN 4
5 #define DHTTYPE DHT22
6 #define LED 5
7 DHT dht (DHTPIN, DHTTYPE);
8
9 void callback(char* topic, byte* payload, unsigned int payloadLength);
10 #define ORG "lx9rln"
11 #define DEVICE_TYPE "jerom"
12 #define DEVICE_ID "1234"
13 #define TOKEN "12345678"
14 String data;
15 float h, t;
16
17 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
18 char publishTopic[] = "iot-2/evt/Data/fmt/json";
19 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
20 char authMethod[] = "use-token-auth";
21 char token[] = TOKEN;
22 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
23
24 //-----
25 WiFiClient wifiClient;
26 PubSubClient client(server, 1883, callback, wifiClient);
27 void setup()
28 {
29   Serial.begin(115200);
30   dht.begin();
31   pinMode(LED, OUTPUT);
32   delay(10);
33   Serial.println();
34   wifiConnect();
35   mqttConnect();
36 }
37
38 void loop()
39 {
40
41   h = dht.readHumidity();
42   t = dht.readTemperature();
43   Serial.print("temperature:");
44   ...

```

## IBM CLOUD

The screenshot shows the IBM Watson IoT Platform dashboard. The 'Browse' tab is selected, and a search for '1234' has been performed. The device '1234' is listed as 'Connected' with a status of 'jerom'. The 'Recent Events' tab is selected, showing a list of events received from the device.

Event	Value	Format	Last Received
Data	{"temperature":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":30,"humidity":28.5}	json	a few seconds ago

# CHAPTER – 8

## TESTING

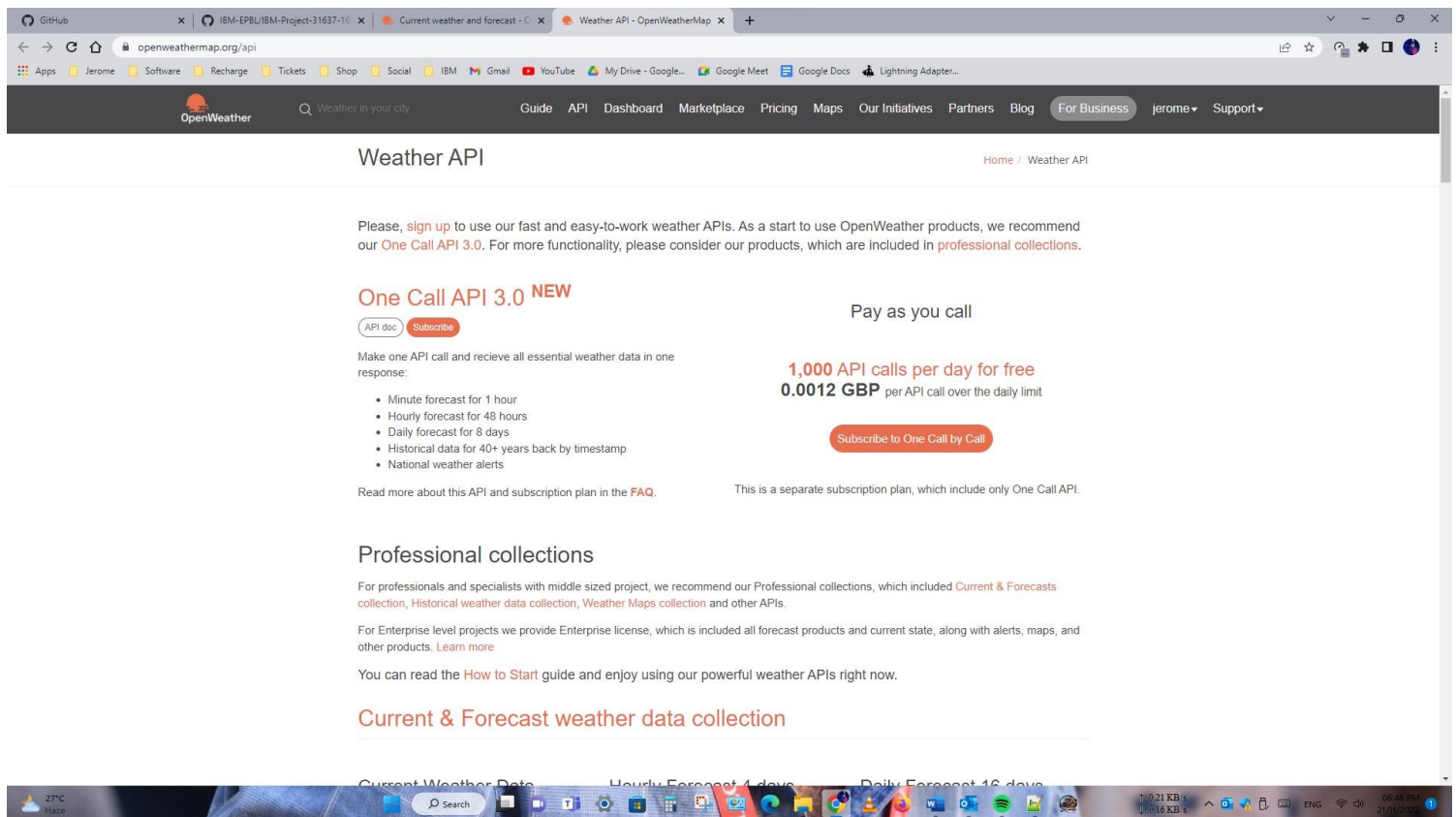
### TEST CASES

#### PROJECT DEVELOPMENT PHASES

##### SPRINT 01

##### SPRINT GOALS:

1. Create and initialize accounts in various public APIs like Open Weather API.
2. Write a Python program that outputs results given the inputs like weather and location.



## MADURAI – WEATHER API

```
{ "coord": { "lon": 78.1167, "lat": 9.9333 }, "weather": [ { "id": 721, "main": "Haze", "description": "haze", "icon": "50n" } ], "base": "stations", "main": { "temp": 300.16, "feels_like": 301.94, "temp_min": 300.16, "temp_max": 300.16, "pressure": 1008, "humidity": 69 }, "visibility": 3000, "wind": { "speed": 1.03, "deg": 0 }, "clouds": { "all": 40 }, "dt": 1669036491, "sys": { "type": 1, "id": 9216, "country": "IN", "sunrise": 1668991463, "sunset": 1669033354 }, "timezone": 19800, "id": 1264521, "name": "Madurai", "cod": 200 }
```

## SPRINT 01

<https://api.openweathermap.org/data/2.5/weather?lat=9.9333&lon=78.1167&appid=5b470269921e324575ff30d836ad5aef>

## PYTHON CODE

```
import requests
api_data="https://api.openweathermap.org/data/2.5/weather?lat=9.9333&lon=78.1167&appid=5b470269921e324575ff30d836ad5aef"
rec=requests.get(url=api_data)
data= rec.json()
print(data)
temp = data['main']['temp']
print("\nTemperature is : ", temp)
humidity = data['main']['humidity']
print("Humidity is : ", humidity)

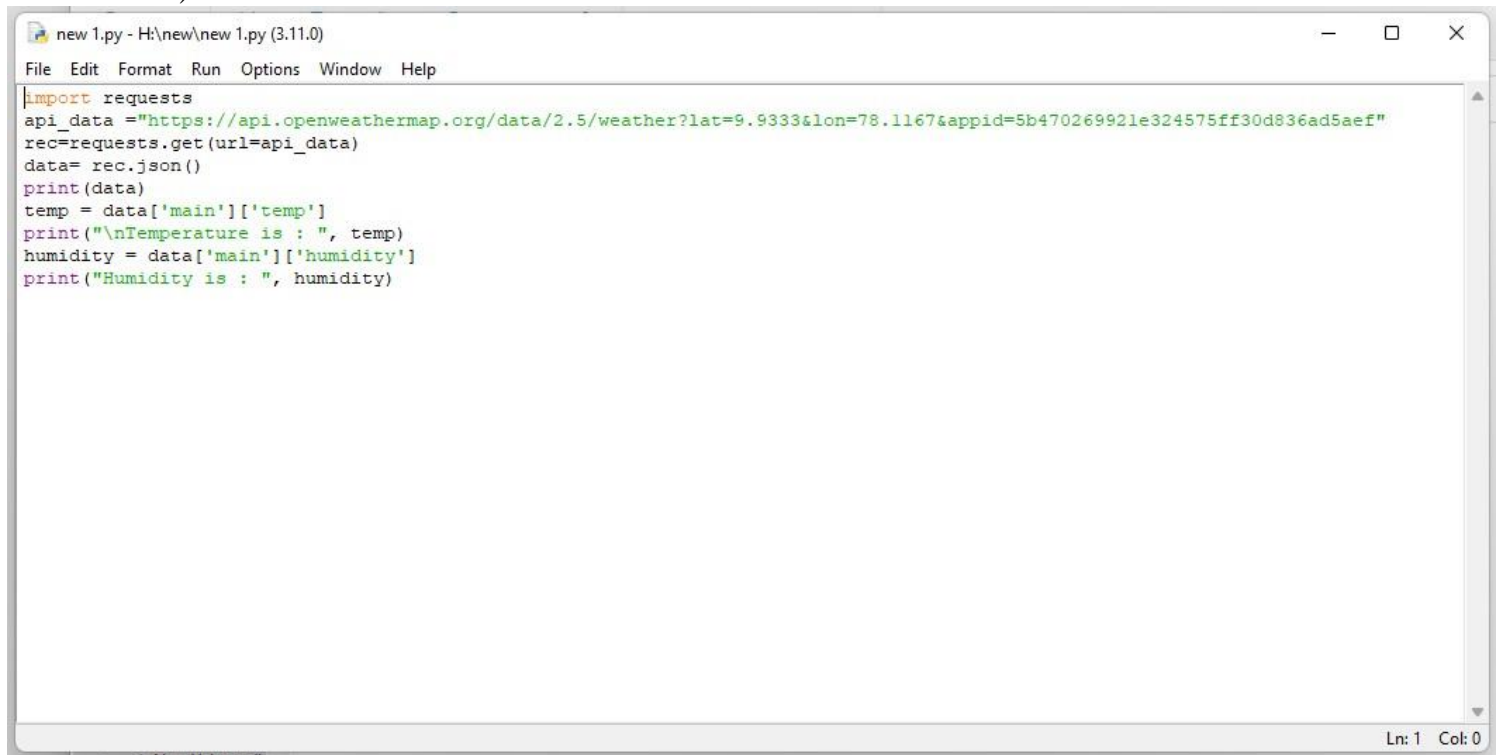
//This code execution will be done in sprint 02
```

## Sprint - 02

### Sprint goal:

Push the code from Sprint 1 to cloud so it can be accessed from anywhere

- 1) To get the data of current weather condition
- 2)

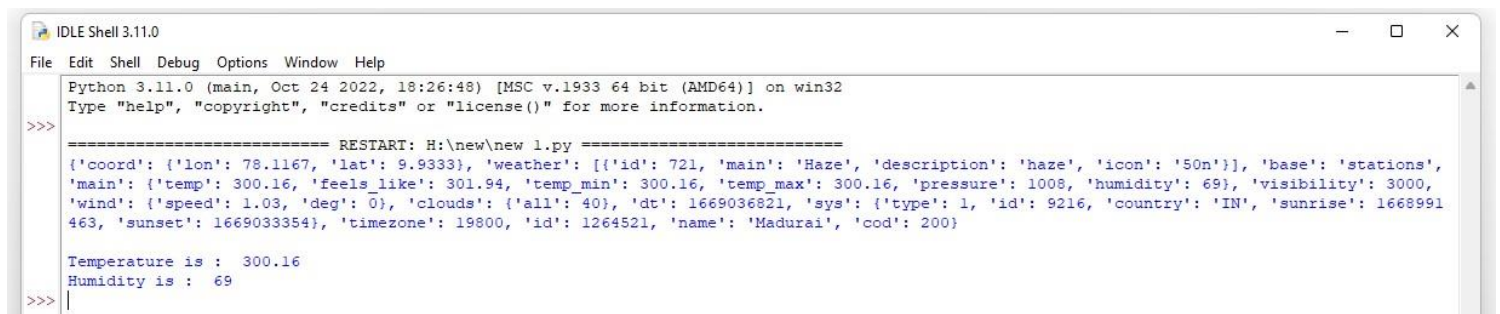


```
new 1.py - H:\new\new 1.py (3.11.0)
File Edit Format Run Options Window Help

import requests
api_data = "https://api.openweathermap.org/data/2.5/weather?lat=9.9333&lon=78.1167&appid=5b470269921e324575ff30d836ad5aef"
rec=requests.get(url=api_data)
data= rec.json()
print(data)
temp = data['main']['temp']
print("\nTemperature is : ", temp)
humidity = data['main']['humidity']
print("Humidity is : ", humidity)
```

Ln: 1 Col: 0

### OUTPUT:



```
IDLE Shell 3.11.0
File Edit Shell Debug Options Window Help

Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
===== RESTART: H:\new\new 1.py =====
{'coord': {'lon': 78.1167, 'lat': 9.9333}, 'weather': [{'id': 721, 'main': 'Haze', 'description': 'haze', 'icon': '50n'}], 'base': 'stations',
'main': {'temp': 300.16, 'feels_like': 301.94, 'temp_min': 300.16, 'temp_max': 300.16, 'pressure': 1008, 'humidity': 69, 'visibility': 3000,
'wind': {'speed': 1.03, 'deg': 0}, 'clouds': {'all': 40}, 'dt': 1669036821, 'sys': {'type': 1, 'id': 9216, 'country': 'IN', 'sunrise': 1668991
463, 'sunset': 1669033354}, 'timezone': 19800, 'id': 1264521, 'name': 'Madurai', 'cod': 200}

Temperature is : 300.16
Humidity is : 69
>>>
```

### 3) Get the data in the IBM cloud

```
new 2.py - H:\new\new 2.py (3.11.0)
File Edit Format Run Options Window Help

#IBM Watson IOT Platform
#pip install wiotp-sdk
import wiotp.sdk.device
import time
import random
myConfig = {
    "identity": {
        "orgId": "lx9rlh",
        "typeId": "jerom",
        "deviceId": "1234"
    },
    "auth": {
        "token": "12345678"
    }
}

def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']

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

while True:
    temp=random.randint(-20,125)
    hum=random.randint(0,100)
    myData={'temperature':temp, 'humidity':hum}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
    time.sleep(2)
client.disconnect()
```

Ln: 1 Col: 0

### OUTPUT:

Browse

Action

Device Types

Interfaces

Add Device +

Identity	Device Information	Recent Events	State	Logs
The recent events listed show the live stream of data that is coming and going from this device.				
Event	Value	Format	Last Received	
event_1	{"temperature":71,"humidity":2}	json	a few seconds ago	
event_1	{"temperature":18,"humidity":15}	json	a few seconds ago	
event_1	{"temperature":19,"humidity":31}	json	a few seconds ago	
event_1	{"temperature":95,"humidity":48}	json	a few seconds ago	
event_1	{"temperature":90,"humidity":2}	json	a few seconds ago	



## SPRINT 03

### SPRINT GOAL:

Integrate the hardware to be able to access the cloud functions and provide inputs to the same.

### POGRAM 01:

**AIM:** To find the Temperature and Humidity DHT22 and ESP32

**PLATFORM:** WOKWI

The screenshot shows the Wokwi IDE interface. On the left, the sketch code is displayed, which includes headers for WiFi, PubSubClient, and DHT. It defines a server, topics, and a callback function. The setup function initializes the DHT22 sensor and connects to the WiFi network. The loop function reads the temperature and humidity from the DHT22 and prints them to the serial monitor.

On the right, the simulation window shows a visual representation of the ESP32 and DHT22 sensor connected. Below the simulation, the console output shows the following data:

```
temperature:30.00
Humidity:44.00
Sending payload: {"temperature":30.00,"humidity":44.00}
Publish ok
temperature:30.00
Humidity:44.00
Sending payload: {"temperature":30.00,"humidity":44.00}
Publish ok
temperature:30.00
Humidity:44.00
Sending payload: {"temperature":30.00,"humidity":44.00}
Publish ok
temperature:30.00
Humidity:44.00
Sending payload: {"temperature":30.00,"humidity":44.00}
Publish ok
temperature:30.00
Humidity:44.00
Sending payload: {"temperature":30.00,"humidity":44.00}
Publish ok
temperature:30.00
Humidity:44.00
Sending payload: {"temperature":30.00,"humidity":44.00}
Publish ok
```

### OUTPUT IN THE IBM CLOUD

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present, and a 'Device Simulator' toggle is visible.

The main content area displays a table of devices. The table has columns for Device ID, Status, Device Type, Class ID, Date Added, Descriptive Location, Added By, and Device Class. One device is listed with ID 1234, status 'Connected', and type 'jerom'.

Below the table, a detailed view of the device is shown, including a 'Recent Events' tab. The events are listed in a table with columns for Event, Value, Format, and Last Received. The events show a stream of temperature and humidity data being received from the device.

Event	Value	Format	Last Received
Data	{"temperature":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":30,"humidity":28.5}	json	a few seconds ago

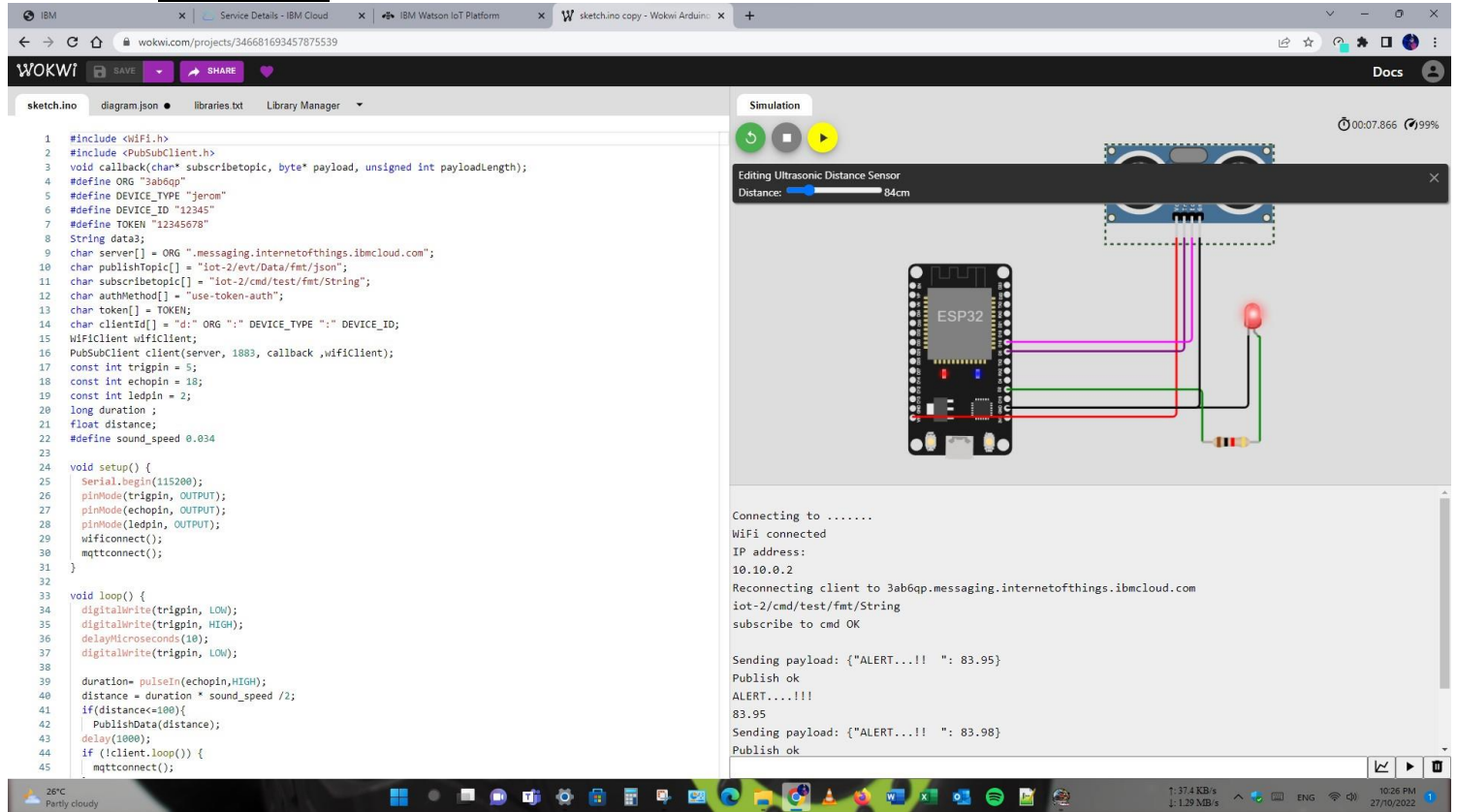
**Output link :** <https://wokwi.com/projects/3489495259220547>



## PROGRAM 02

**AIM:** Write code and connection in Wowki for ultrasonic sensor. Whenever distance is less than 100 cms send “Alert” to IBM cloud and display in device recent events by using ESP32

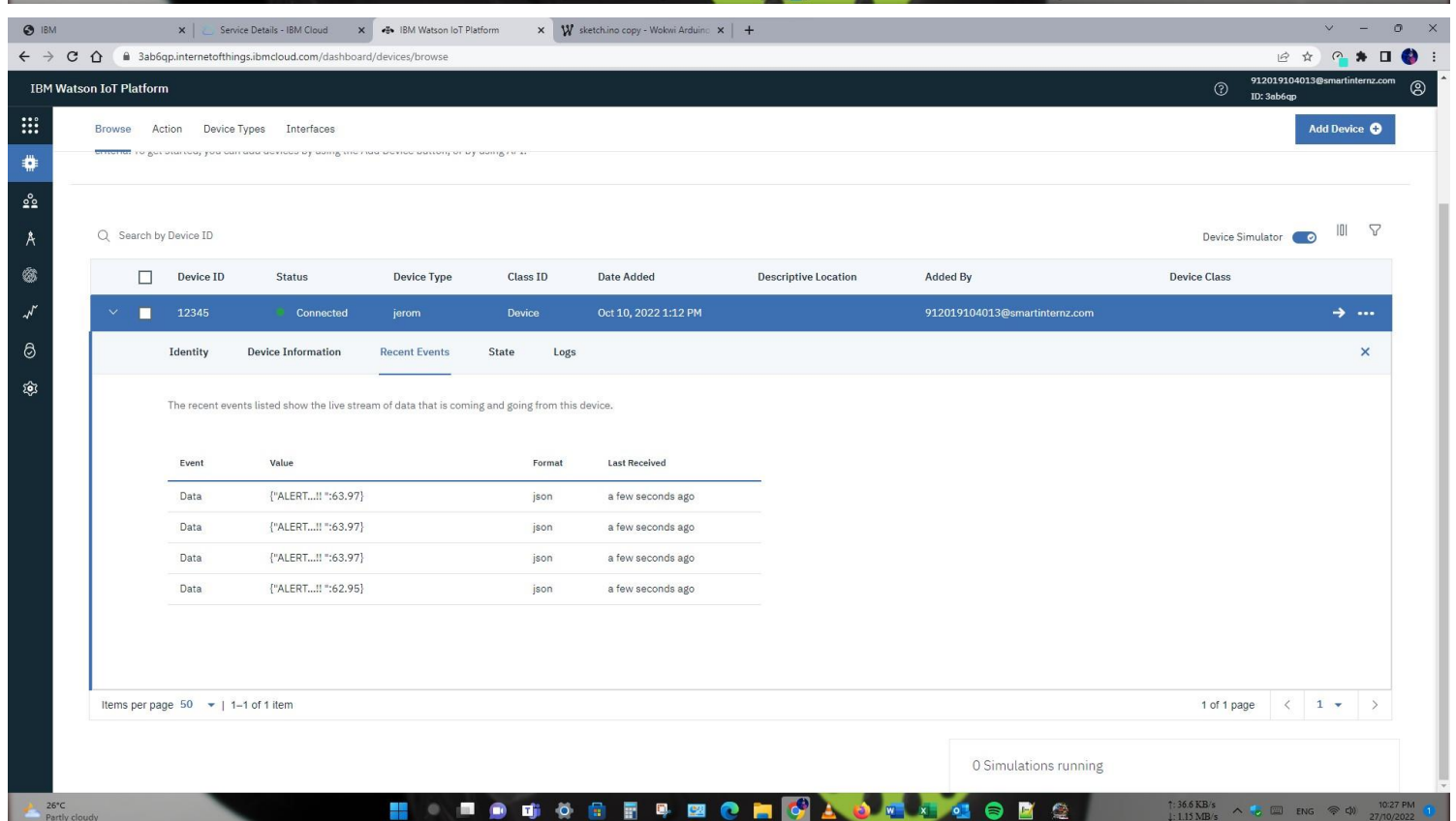
### PLATFORM: WOKWI



The screenshot shows the Wokwi IDE interface. On the left, the Arduino sketch is displayed, which includes the necessary libraries and code for connecting to the IBM Watson IoT Platform and publishing data. The code defines a callback function, sets up the WiFi and MQTT client, and implements a loop that triggers an LED and publishes an alert when the ultrasonic sensor detects a distance less than 100 cm.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadlength);
4 #define ORG "3ab6qp"
5 #define DEVICE_TYPE "jerom"
6 #define DEVICE_ID "12345"
7 #define TOKEN "12345678"
8 String data3;
9 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
10 char publishTopic[] = "iot-2/evt/data/fmt/json";
11 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
12 char authMethod[] = "use-token-auth";
13 char token[] = TOKEN;
14 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
15 WiFiClient wifiClient;
16 PubSubClient client(server, 1883, callback, wifiClient);
17 const int trigpin = 5;
18 const int echopin = 18;
19 const int ledpin = 2;
20 long duration;
21 float distance;
22 #define sound_speed 0.034
23
24 void setup() {
25   Serial.begin(115200);
26   pinMode(trigpin, OUTPUT);
27   pinMode(echopin, OUTPUT);
28   pinMode(ledpin, OUTPUT);
29   wifiConnect();
30   mqttConnect();
31 }
32
33 void loop() {
34   digitalWrite(trigpin, LOW);
35   digitalWrite(trigpin, HIGH);
36   delayMicroseconds(10);
37   digitalWrite(trigpin, LOW);
38
39   duration = pulseIn(echopin, HIGH);
40   distance = duration * sound_speed / 2;
41   if(distance < 100){
42     PublishData(distance);
43     delay(1000);
44     if (!client.loop()) {
45       mqttConnect();
46     }
47   }
48 }
```

The simulation window on the right shows an ESP32 microcontroller connected to an ultrasonic sensor and an LED. The console output indicates that the device is connected to the WiFi network, has an IP address of 10.10.0.2, and is successfully publishing data to the IBM Watson IoT Platform. The payload sent is {"ALERT...!! ": 83.95}.



The screenshot shows the IBM Watson IoT Platform dashboard. The device details for device ID 12345 are displayed, showing it is connected and has a status of "Connected". The "Recent Events" tab is selected, showing a list of events published by the device. The events are listed in a table with columns for Event, Value, Format, and Last Received.

Event	Value	Format	Last Received
Data	{"ALERT...!! ":63.97}	json	a few seconds ago
Data	{"ALERT...!! ":63.97}	json	a few seconds ago
Data	{"ALERT...!! ":63.97}	json	a few seconds ago
Data	{"ALERT...!! ":62.95}	json	a few seconds ago

The dashboard also shows the device's identity, device information, and state logs. The bottom of the dashboard indicates that 0 simulations are running.

**PYTHON CODE:**

```
const int trigpin = 5;
const int echopin = 18;
const int ledpin = 2;
long duration ;
float distance;
#define sound_speed 0.034
void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    pinMode(trigpin, OUTPUT);
    pinMode(echopin, OUTPUT);
    pinMode(ledpin, OUTPUT);
    wificonnect();
    mqttconnect();
}
void loop() {
    digitalWrite(trigpin, LOW);
    digitalWrite(trigpin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigpin, LOW);
    duration= pulseIn(echopin,HIGH);
    distance = duration * sound_speed /2;
    if(distance<=100){
        PublishData(distance);
        delay(1000);
        if (!client.loop()) {
            mqttconnect();
        }
        digitalWrite(ledpin, HIGH);
        Serial.println("ALERT .. !!!");
        Serial.println(distance);
    }
    else
    {
        digitalWrite(ledpin, LOW);
    }
}
```

**Output link:** <https://wokwi.com/projects/349037081204359762>

## **SPRINT 04**

### **SPRINT GOAL:**

Optimize all the shortcomings and provide better user experience.

### **Process 01:**

**Aim:** Having a USER INTERFACE method to get the Light system in Road Safety

**Platform:** NODE-RED

### **OUTPUT:**

**User Interface (UI):**



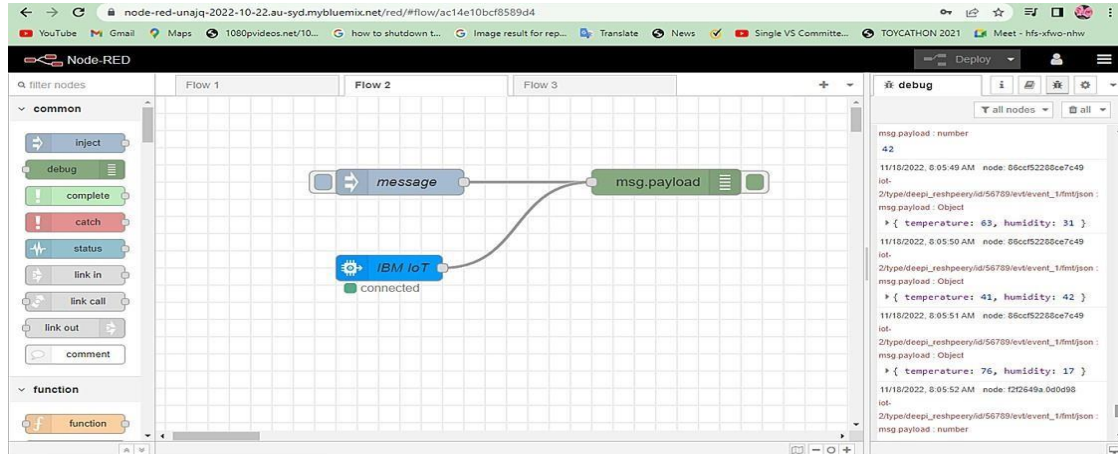
### **OUTPUT:**

```
11/18/2022, 8:08:42 AMnode: f2f2649a.0d0d98msg.payload : Object
{ command: "ligthon" }
11/18/2022, 8:08:43 AMnode: f2f2649a.0d0d98msg.payload : Object
{ command: "LigthOFF" }
11/18/2022, 8:08:48 AMnode: f2f2649a.0d0d98msg.payload : Object
{ command: "ligthon" }
11/18/2022, 8:08:48 AMnode: f2f2649a.0d0d98msg.payload : Object
{ command: "ligthon" }
11/18/2022, 8:08:49 AMnode: f2f2649a.0d0d98msg.payload : Object
{ command: "LigthOFF" }
```

## Process 02 :

**Aim:** Having a Temperature and Humidity in Road Safety

**Platform:** NODE-RED



## OUTPUT :

2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 63, humidity: 17 }

11/18/2022, 8:21:57 AMnode: 86ccf52288ce7c49iot-2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 46, humidity: 18 }

11/18/2022, 8:21:57 AMnode: 86ccf52288ce7c49iot-2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 34, humidity: 27 }

11/18/2022, 8:21:57 AMnode: 86ccf52288ce7c49iot-2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 83, humidity: 18 }

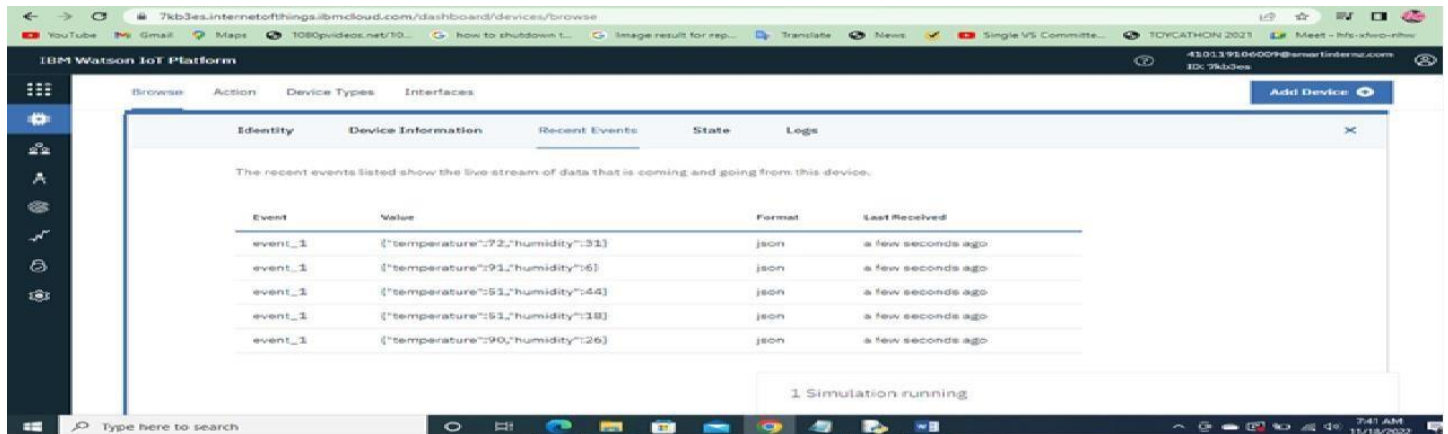
11/18/2022, 8:21:57 AMnode: 86ccf52288ce7c49iot-2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 97, humidity: 34 }

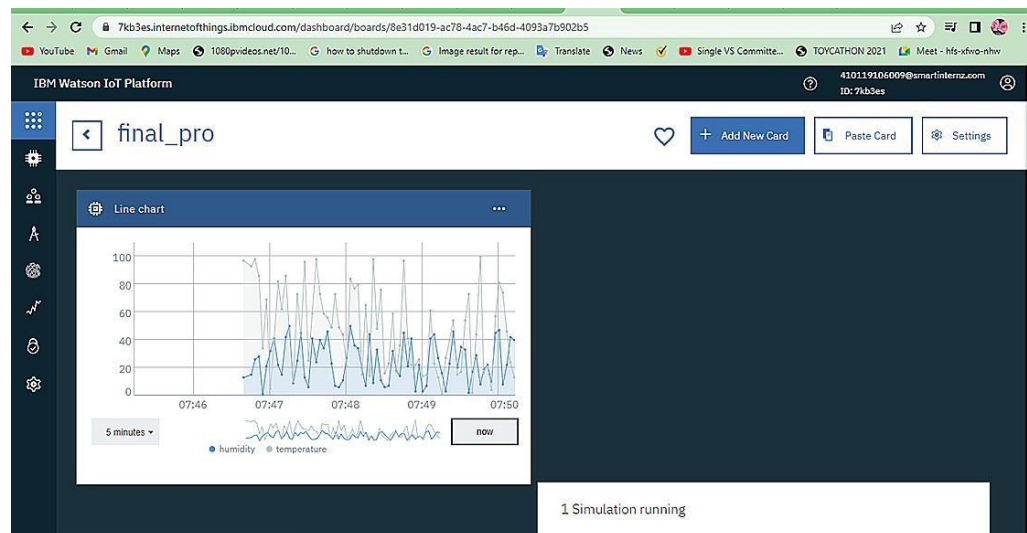
11/18/2022, 8:23:15 AMnode: 86ccf52288ce7c49iot-2/type/jerom/id/56789/evt/event\_1/fmt/json : msg.payload : Object

{ temperature: 61, humidity: 5 }

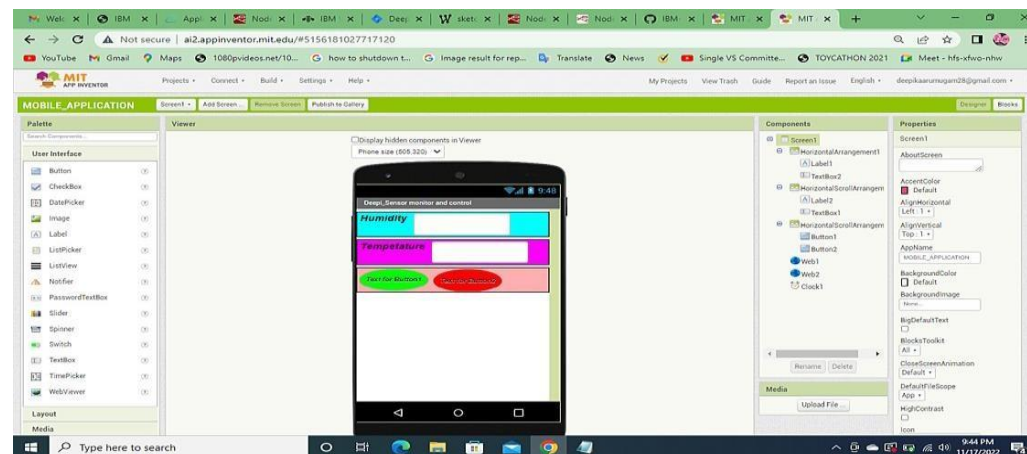
## NODE-RED to IBM cloud



## LINE CHART



By the use of MIT- APP inventor



# USER ACCEPTANCE TESTING

## 1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the[Product Name] project at the time of the release to User Acceptance Testing (UAT).

## 2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

## 3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

## **CHAPTER -9**

### **ADVANTAGES**

- They improve vehicle safety by providing real-time traffic information to the driver. Road signs play an important role in road safety.
- To be effective, road signs must be visible at a distance that enables drivers to take the necessary actions.
- Traffic signs provide valuable information to drivers and other road users.
- Being able to forecast and plan for the future when it comes to the local climate is a major advantage when it comes to planning tourism facilities.
- The transport sector can also benefit, as infrastructure can be set up to measure road surface conditions to improve traffic safety

## **CHAPTER -10**

### **CONCLUSION**

We hope to gain hands-on experience with the trending technologies of "Embedded System" and "Internet of Things" through this project. IoT-enabled industrial monitoring systems have become increasingly popular in a variety of industries because they improve safety standards by providing real-time monitoring of critical parameters such as temperature, humidity, and smoke, as well as alerting officials and workers regularly. The implementation is not only for safety reasons, but it also has the potential to increase industry yields. In our project, the Internet of Things (IoT) is used to collect data and communicate through the internet. We hope that our project will be beneficial enough to be implemented in industries across India, saving lives and property from accidents and risks that are often overlooked by industry personnel and users. Companies in the industrial and logistics sectors can better meet the new era of instant needs by utilizing the Industrial Internet of Things (IIoT)

## **CHAPTER 11**

### **FUTURE SCOPE**

The future of road safety is uncertain and definitely not the same for all regions of the world. Countries with a mature road safety approach and an ambition to make further progress are expected to move in the direction of a pro-active approach: a Safe System approach. It is reported that many LMIC, meanwhile, are on the brink of designing road safety strategies and implementing action plans. The international community is willing to support LMIC, but LMIC cannot simply copy successful HIC strategies because local circumstances differ. The principles of successful HIC strategies are applicable, but the priorities and action plans should take root in and align with local conditions



## CHAPTER 12

### APPENDIX

#### SOURCE – CODE

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQTT

void callback(char* subscribetopic, byte* payload, unsigned i
\
//-----credentials of IBM Accounts-----

#define ORG "7kb3es"//IBM ORGANITION ID
#define DEVICE_TYPE "jerom"//Device type mentioned #define DEVICE_ID
"56789"//Device ID mentioned in ibm watson #define TOKEN "CUsp45PBuBEsQeJ_9N"
//Token
String data3;

//----- Customise the above values -----
char server[] = ORG ".messaging.internetofthings.ibmcloud.com" char publishTopic[] = "iot-
2/evt/Data/fmt/json";// topic name char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd
R char authMethod[] = "use-token-auth";// authentication method char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//cl

//
WiFiClient wifiClient; // creating the instance for wificlien PubSubClient client(server,
1883, callback ,wifiClient); //ca wificredential
const int trigpin = 5;
const int echopin = 18; const int ledpin = 2;

long duration ; float distance;
```

```

#define sound_speed 0.034 void setup() {
// put your setup code here, to run once: Serial.begin(115200);
pinMode(trigpin, OUTPUT);

pinMode(echopin, OUTPUT); pinMode(ledpin, OUTPUT); wificonnect(); mqttconnect();
}

void loop() { digitalWrite(trigpin, LOW); digitalWrite(trigpin, HIGH);
delayMicroseconds(10); digitalWrite(trigpin, LOW);

duration= pulseIn(echopin,HIGH); distance = duration * sound_speed /2; if(distance<=100){
PublishData(distance); delay(1000);
if (!client.loop()) { mqttconnect();
}
digitalWrite(ledpin, HIGH); Serial.println("ALERT  !!!");
Serial.println(distance);
}
else
{
digitalWrite(ledpin, LOW);
}
// put your main code here, to run repeatedly: delay(10); // this speeds up the simulation
}

/*.....retrieving to Cloud...

void PublishData(float distance) { mqttconnect();//function call for connecting to ibm

// creating the String in in form JSon to update the data String payload = "{\"ALERT...!! \":
";
payload += distance; payload += "}";

Serial.print("Sending payload: "); Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str()))

```

```

Serial.println("Publish ok");// if it sucessfully upload publish failed
} else {
Serial.println("Publish failed");
}

}

void mqttconnect() {
if (!client.connected()) { Serial.print("Reconnecting client to "); Serial.println(server);
while (!!!client.connect(clientId, authMethod, token)) { Serial.print(".");
delay(500);
}

initManagedDevice(); Serial.println();
}
}

void wificonnect() //function defination for wificonnect
{
Serial.println(); Serial.print("Connecting to ");

WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credent while (WiFi.status() !=
WL_CONNECTED) {
delay(500); Serial.print(".");
}
Serial.println(""); Serial.println("WiFi connected"); Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void initManagedDevice() {
if (client.subscribe(subscribetopic)) { Serial.println((subscribetopic));
Serial.println("subscribe to cmd OK");
} else {
Serial.println("subscribe to cmd FAILED");
}
}
}

```

```

void callback(char* subscribetopic, byte* payload, unsigned i
{

Serial.print("callback invoked for topic: "); Serial.println(subscribetopic);
for (int i = 0; i < payloadLength; i++) {
//Serial.print((char)payload[i]); data3 += (char)payload[i];
}

Serial.println("data: " + data3); if(data3=="lighton")
{
Serial.println(data3);
}
else
{
Serial.println(data3);
}
data3="";
}

```

**LINKS:**

**GITHUB:** <https://github.com/IBM-EPBL/IBM-Project-31637-1660203738>

**DEMO LINK -**