

Internet Of Things

In partial fulfillment for the completion of

Signs with Smart Connectivity For Better Road Safety

SUBMITTED BY

TEAM LEADER:

RAMKUMAR C

TEAM MEMBER:

PRABUDEVA R

BARANI K

SURYA S

DATE	19 November 2022
Team ID	PNT2022TMID29179

BACHELOR OF ENGINEERING

ELECTRONICS AND COMMUNICATION ENGINEERING

MAILAM ENGINEERING COLLEGE,

MAILAM.

**During the academic year
2022-2023 (Odd Semester)**

TABLE OF CONTENTS

S. No	TITLE	PAGE NO
1.	Introduction	3
2.	Literature survey	4
3.	Ideation & Proposed Solution	5
4.	Requirement Analysis	8
5.	Project Design	10
6.	Project Planning & Scheduling	15
7.	Requirements	17
8	Coding & Solutioning	19
9.	Testing	31
10.	Results	42
11.	Advantages & Disadvantages	43
12.	Conclusion	44
13.	Future Scope	45
14.	Appendix	47

1.INTRODUCTION:

Machine learning is one of the key enabling technologies for autonomous vehicles. An autonomous vehicle can learn how to recognize the surroundings and can base its strategic decisions on the information learnt. It is only a matter of time for autonomous driving to replace of human drivers completely. However, for the time being, there are still important, yet not completely addressed, challenges for autonomous driving. Road-sign classification is one of these challenges. Varying weather conditions, changing lighting throughout the day and occlusion are known to pose challenges to road-sign recognition/classification in real-time applications.

1.1 Project Overview:

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.

By using the Weather API we can get the weather reports based on which we can set the speed limit to particular area. 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. There are three switches through which you can switch the display to different modes.

1.2 Purpose:

Due to this heavy traffic, the number of road accidents are increased which is a major issue. Our project helps to decrease the number of road accidents using smart connected sign boards using Internet Of things (IOT).

2. LITERATURE SURVEY:

2.1 Existing System:

In present Systems the road signs and the speed limits are Static. But the road sign 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. Software used • Arduino IDE • Embedded C

2.2 References :

- [1] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
- [2] J. Jin, K. Fu, and C. Zhang, "Traffic sign recognition with hinge loss trained convolutional neural networks," IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 5, pp. 1991–2000, 2014.
- [3] A. Gonzalez, L. M. Bergasa, and J. J. Yebes, "Text detection and ' recognition on traffic panels from street-level imagery using visual appearance," IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 1, pp. 228–238, 2014.
- [4] J. Greenhalgh and M. Mirmehdi, "Recognizing text-based traffic signs," IEEE Transactions on Intelligent Transportation Systems, vol. 16, no. 3, pp. 1360–1369, 2015

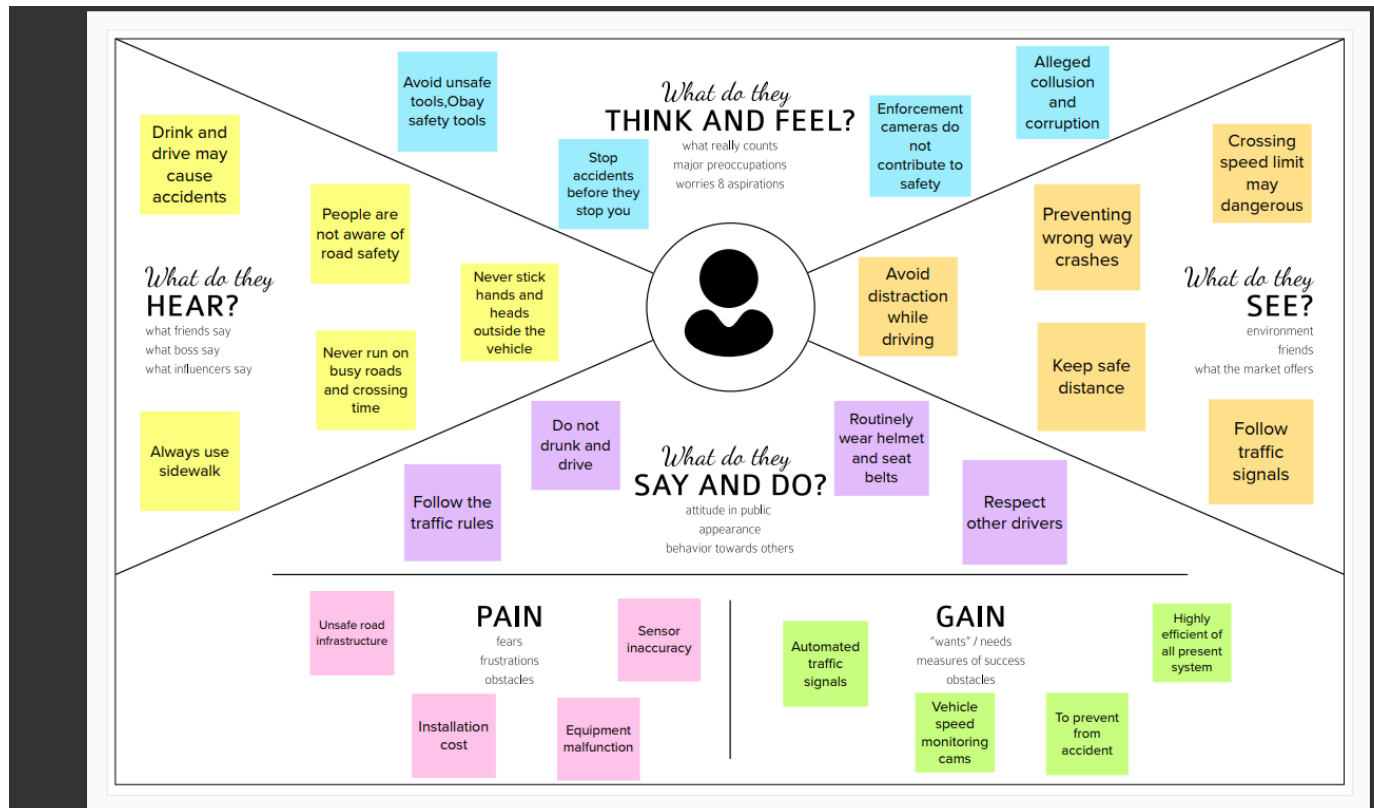
2.3 Problem Statement Definition :

This project will replace static signs with smart signs that can adjust speed restrictions based on the weather and climate, display detour instructions in the event of an accident, and display alert messages in the event of hospitals, schools, or roadworks.

3. IDEATION & PROPOSED SOLUTION :

3.1 Empathy Map Canvas :

Fig 3.1 Empathy Map



3.2 Ideation & Brainstorming :

Ideation Phase
Brainstorm & Idea Prioritization Template

Date	1 OCTOBER 2022
Team ID	PNT2022TMI029179
Project Name	Signs With Smart Connectivity for Better Road Safety
Maximum Marks	4 Marks

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

Brainstorm & idea prioritization

Use this template in your next brainstorming session as your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 40 minutes project
- Team's objective
- 3-4 people recommended

[View resources & details](#)

Before you collaborate

A state-of-the-art presentation guide is coming up with the content. Please don't put your head in the box yet.

[Go to slides](#)

Team gathering

Define who should participate in the session and why. Also, define the session's duration and format.

Set the goal

How would the problem and the solution be designed? Brainstorming session.

Keep track of the brainstorming

Use the following questions to track the progress and productivity.

[Open editor](#)

Define your problem statement

What problem are you trying to solve? (How good is the problem you're trying to solve? The better the focus of your brainstorm, the better.)

[Go to slides](#)

PROBLEM

In present system the road signs are static but it should change some climatic changes and fatal situations.

Key notes of your brainstorming

To create project and presentation

[View slides](#)

[Download all slides](#)

[View content](#)

[View details](#)

[Download to excel](#)

Fig 3.2 Ideation & Brainstorming

3.3 Proposed Solution :

S. No	Parameter	Description
1.	Problem Statement (Problem to be solved)	Signs with smart connectivity for Better Road Safety are intended to control speed, increase safety, and display the latest weather information.
2.	Idea / Solution description	Replacing traditional roadside signage with IoT-enabled smart ones. Smart signs are built using LED and the Internet of Things.

3.	Novelty / Uniqueness	Due to the use of LEDs, they can be seen from a distance. It is possible to view them from a distance since LEDs are used. These specifics were obtained from a weather-tracking app. Additionally, it gives information about nearby places like hospitals, schools, etc. so that customers may make decisions based on that knowledge, such as speeding.
4.	Social Impact / Customer Satisfaction	On the department of road safety, these are clearly felt. The avoidance of accidents can be achieved by imposing a user-set speed limit.
5.	Business Model (Revenue Model)	The government's implementation of these for common citizens is an excellent endeavor to increase public awareness. This can be funded separately by the government, which lays the groundwork for a safer environment.
6.	Scalability of the Solution	Because it is more visible than conventional signals, it has a better chance of reducing danger and could even save many lives.

3.4 Problem Solution Fit :

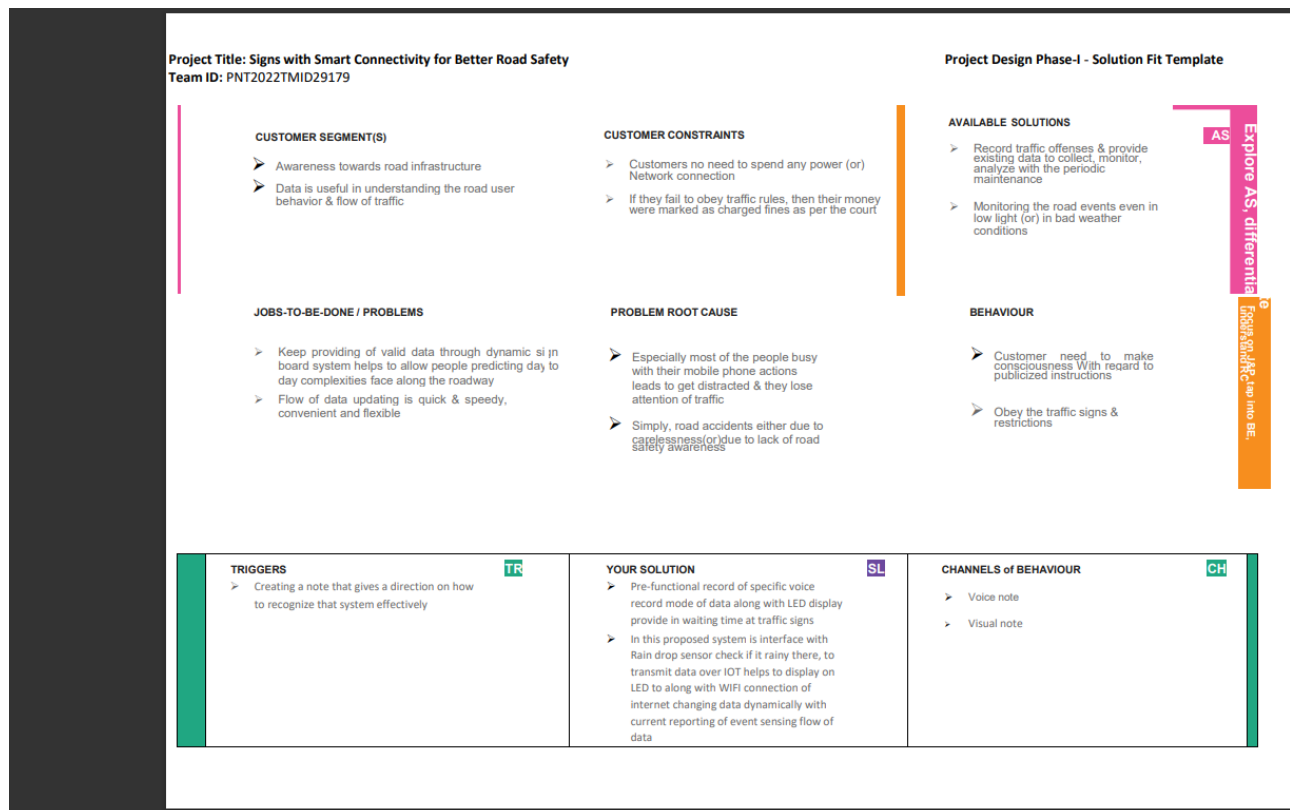


Fig 3.4 Problem Solution Fit

4. REQUIREMENT ANALYSIS :

4.1 Functional requirement:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Requirements	Static signboards will be replaced with smart linked sign boards that meet all criteria.
FR-2	User Registration	User Registration can be done through a Website or Gmail

FR-3	User Confirmation	Phone Confirmation Email confirmation OTP authentication
FR-4	Payments options	Bank Transfers
FR-5	Product Delivery and installation	The installation fee will be depend upon the length of the road.
FR-6	Product Feedback	Will be shared through a website via Gmail

4.2 Non-Functional requirement:

Following are the Non-Functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Will provide the clear product instructions and a self- explanatory product which is simple to use.
NFR-2	Security	Cloud data must be contained within the network, collapsing to be the realtime avoidance should be avoided, and the board will be monitored constantly.
NFR-3	Reliability	Hardware will be frequently tested.
NFR-4	Performance	The smart board must provide a better user experience and deliver the accuracy output.
NFR-5	Availability	All of the functions and the user demands will be provided, depend upon the customer needs.

NFR-6	Scalability	The product is based on road safety and should cover the entire highway system.
-------	--------------------	---

5. PROJECT DESIGN :

5.1 Data Flow Diagram :

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Data flow Diagram- Signs with smart connectivity for better road safety:

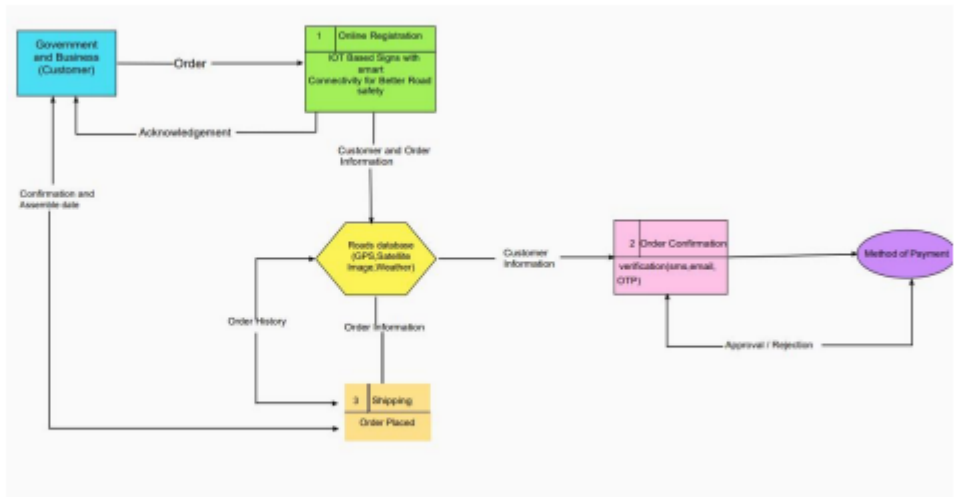


Fig 5.1 Data Flow Diagram - Signs with smart connectivity for better road safety

5.2 Solution & Technical Architecture :

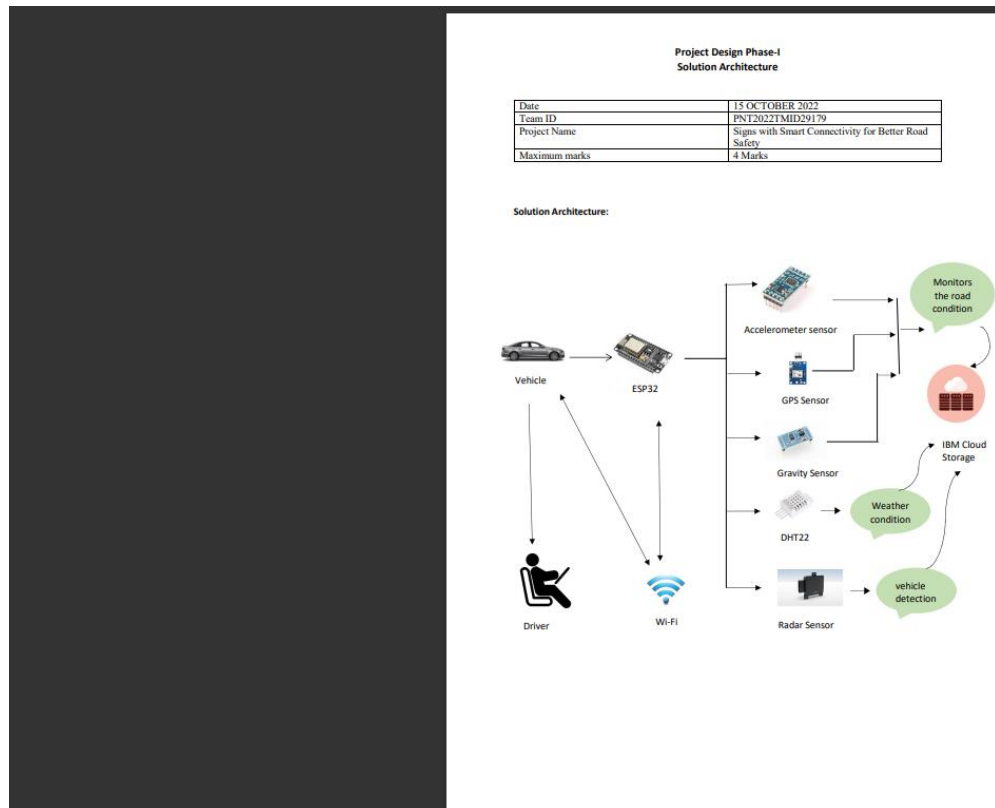


Fig 5.2 Solution & Technical Architecture - Signs with smart connectivity for better road safety

GUIDELINES:

- 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.

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Security Implementations	Strong security system that anyone without login credentials and hackers are not allowed to enter the network.	Firewall, Firebase, cyber resiliency strategy
2.	Scalable Architecture	Easy to expand the operating range by increasing the bandwidth of the network.	IoT, internet.
3.	Availability	Available anytime and everywhere 24/7 as long as the user is signed into the network.	IBM Cloud
4.	Performance	Supports a large number of users to access the technology simultaneously.	IBM Cloud

5.3 User stories :

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	Access my account / dashboard	High	Sprint-1
Weather	openweathermap	USN-2	As a user, I want to check the weather of that location	Get the weather of that location	High	Sprint-1
IoT devices	Automation	USN-3	As a user, I want to use IoT devices for automation purposes	Get the work done without manual effort	High	Sprint-2
Python code	Random data	USN-4	As a user, I want to give some input to the devices for performing some action to complete the tasks very easily	Get the data workflow	Medium	Sprint-1
IBM Cloud	Cloud services	USN-5	As a user, I want to deploy <u>these application</u> for public version	Useful for all domain users	High	Sprint-1
Node-Red	Integration	USN-6	As a user, I want to integrate the applications with hardware	To precise for linear workflow	Medium	Sprint-3
Web UI	Interaction	USN-7	As a user, I want to interact with the digital products	To interact with the users	Medium	Sprint-2

Data validation	Checking accuracy	USN-8	As a user, I can check the ability and accuracy of the model in obtaining the required information	Check the capability of the model	High	Sprint-2
Data extraction	Obtaining the data	USN-9	As a user, I can retrieve the result data from the application for data storage for further uses	Download the result in the form of data	High	Sprint-3

6. PROJECT PLANNING & SCHEDULING :

6.1 Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Resources Initialization	Create and initialize accounts in various public APIs like OpenWeatherMap API.	1	LOW	RAMKUMAR C PRABUDEVA R BARANI K SURYA S
Sprint-1	Local Server/Software Run	Write a Python program that outputs results given the inputs like weather and location.	1	MEDIUM	RAMKUMAR C PRABUDEVA R BARANI K SURYA S
Sprint-2	Push the server/software to cloud	Push the code from Sprint 1 to cloud so it can be accessed from anywhere	2	MEDIUM	RAMKUMAR C PRABUDEVA R BARANI K SURYA S
Sprint-3	Hardware initialization	Integrate the hardware to be able to access the cloud functions and provide inputs to the same.	2	HIGH	RAMKUMAR C PRABUDEVA R BARANI K SURYA S
Sprint-4	UI/UX Optimization & Debugging	Optimize all the shortcomings and provide better user experience.	2	LOW	RAMKUMAR C PRABUDEVA R BARANI K SURYA S

6.2 Sprint Delivery Schedule:

Sprint	Total Story Point	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	31 Oct 2022
Sprint3	20	6 Days	07 Nov 2022	12 Nov 2022	20	07 Nov 2022
Sprint4	20	6 Days	14 Nov 2022	19 Nov 2022	20	14 Nov 2022

7. REQUIREMENTS :

The hardware and software requirements of the project are:

7.1 Software Requirements :

- Arduino IDE
- Weather API
- IBM Cloud Platform

7.2 Hardware Requirements :

1.NODE MCU(12-E):

New Node Mcu Lua ESP8266 CH340G ESP-12E Wireless WIFI Internet Development Board
ESP12E is a WIFI enabled Arduino-alike development board, Which can dramatically reduce the redundant work for configuring and manipulating hardware. Code like arduino, but interactively in Lua script.



Fig 7.2.1 NODE MCU(12-E)

2.PUSH BUTTON:

A Pushbutton switch is a switch featuring a button you push to open and close a circuit. Depending on the series, Pushbutton switches could perform momentary or maintained functions.

E-Switch carries numerous momentary and maintained pushbutton switches that can be non-illuminated or illuminated, with multiple LED colors and lens colors, with up to IP67 ratings for protection against dust and moisture.



Fig 7.2.2 Push Button

3.OLED DISPLAY:

OLED displays are made by placing a series of organic thin films between two conductors. When an electrical current is applied, a bright light is emitted. A simple design - which brings with it many advantages over other display technologies.

OLEDs enable emissive displays - which means that each pixel is controlled individually and emits its own light (unlike LCDs in which the light comes from a backlighting unit). OLED displays feature great image quality - bright colors, fast motion and most importantly - very high contrast.

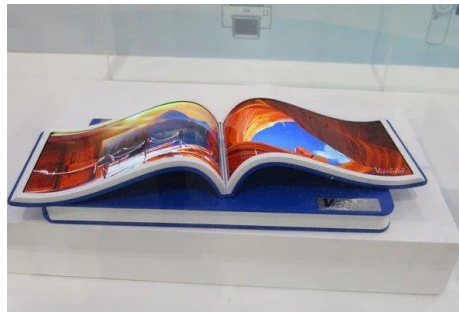


Fig 7.2.3 OLED Display

8. CODING & SOLUTIONING :

8.1 Feature 1:

```
#include <ESP8266WiFi.h> #include
<PubSubClient.h> const char* ssid =
"SB-IOT1"; const char* password =
"sb@iot11";
String command1,command2;
#define ORG "bhip5y"
#define DEVICE_TYPE "Vamsi"
```

```

#define DEVICE_ID "8500"

#define TOKEN "8500913778" String command; char server[] = ORG
".messaging.internetofthings.ibmcloud.com"; char topic[] = "iot-
2/cmd/home/fmt/String"; char authMethod[] = "use-token-auth";
char token[] = TOKEN; char clientId[] = "d:" ORG ":" DEVICE_TYPE ":"
DEVICE_ID;

//Serial.println(clientId);

#include <Wire.h>

#include <Adafruit_SSD1306.h>

#include <Adafruit_GFX.h>

#define SSD1306_LCDHEIGHT 64

// OLED display TWI address

#define OLED_ADDR 0x3C

Adafruit_SSD1306 display(-1);

#if (SSD1306_LCDHEIGHT != 64)

#error("Height incorrect, please fix Adafruit_SSD1306.h!");

#endif void callback(char* topic, byte* payload, unsigned int
payloadLength);

WiFiClient wifiClient;

PubSubClient client(server, 1883, callback, wifiClient); void

setup() {

    display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR);

    Serial.begin(115200);

    Serial.println();

    pinMode(D1,OUTPUT);

    wifiConnect();

    mqttConnect();

}

```

```

void loop() {
    if (!client.loop()) {
        mqttConnect();
    }
    delay(100);
}

void wifiConnect() {
    Serial.print("Connecting to "); Serial.print(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.print("\nWiFi connected, IP address: "); Serial.println(WiFi.localIP());
}

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

void initManagedDevice() {
    if (client.subscribe(topic)) {

```

```

    Serial.println("subscribe to cmd OK");

} else {

    Serial.println("subscribe to cmd FAILED");

}

}

void callback(char* topic, byte* payload, unsigned int payloadLength) {

    Serial.print("callback invoked for topic: "); Serial.println(topic);

    for (int i = 0; i < payloadLength; i++) {

        //Serial.println((char)payload[i]);

        command += (char)payload[i];

    }

    Serial.println(command);

    command1=getValue(command,',',0);

    command2=getValue(command,',',1); if(command1=="1"){

        display.clearDisplay();

        // display a line of text

        display.setTextSize(1);

        display.setTextColor(WHITE);

        display.setCursor(0,10);

        display.print(command);

        // update display with all of the above graphics

        display.display();

    }

    command =""; command1

    =""; command2="";

}

String getValue(String data, char separator, int index)

{

```

```

int found = 0;

int strIndex[] = { 0, -1 };

int maxIndex = data.length() - 1;

for (int i = 0; i <= maxIndex && found <= index; i++) {

    if (data.charAt(i) == separator || i == maxIndex) {

        found++;

        strIndex[0] = strIndex[1] + 1;

        strIndex[1] = (i == maxIndex) ? i+1 : i;

    }

}

return found > index ? data.substring(strIndex[0], strIndex[1]) : "";

}

```

8.2 Feature 2 :

```

#include <ESP8266WiFi.h> #include
<PubSubClient.h> const char* ssid =
"SB-IOT1"; const char* password =
"sb@iot11";

String command1,command2;

#define ORG "bhip5y"

#define DEVICE_TYPE "Vamsi"

#define DEVICE_ID "8500"

#define TOKEN "8500913778" String command; char server[] = ORG
".messaging.internetofthings.ibmcloud.com"; char topic[] = "iot-
2/cmd/home/fmt/String"; char authMethod[] = "use-token-auth";

```



```

char token[] = TOKEN; char clientId[] = "d:" ORG ":" DEVICE_TYPE ":"
DEVICE_ID;

//Serial.println(clientId);

#include <Wire.h>

#include <Adafruit_SSD1306.h>

#include <Adafruit_GFX.h>

#define SSD1306_LCDHEIGHT 64

// OLED display TWI address

#define OLED_ADDR 0x3C

Adafruit_SSD1306 display(-1);

#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif

void callback(char* topic, byte* payload, unsigned int payloadLength);

WiFiClient wifiClient;

PubSubClient client(server, 1883, callback, wifiClient); void

setup() {
    display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR);

    Serial.begin(115200);

    Serial.println();

    pinMode(D1,OUTPUT);

    wifiConnect();

    mqttConnect();
}

void loop() {
    if (!client.loop()) {

```

```

    mqttConnect();
}
delay(100);
}

void wifiConnect() {
    Serial.print("Connecting to "); Serial.print(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.print("\nWiFi connected, IP address: "); Serial.println(WiFi.localIP()); }

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

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

```

```

    Serial.println("subscribe to cmd FAILED");
}
}

void callback(char* topic, byte* payload, unsigned int payloadLength) {

    Serial.print("callback invoked for topic: "); Serial.println(topic);

    for (int i = 0; i < payloadLength; i++) {

        //Serial.println((char)payload[i]);

        command += (char)payload[i];

    }

    Serial.println(command);

    command1=getValue(command,',',0);

    command2=getValue(command,',',1); if(command1=="3"){

        display.clearDisplay();

        // display a line of text

        display.setTextSize(1);

        display.setTextColor(WHITE);

        display.setCursor(0,10);

        display.print(command2);

        // update display with all of the above graphics

        display.display();

    }

    command = ""; command1

    = ""; command2="";

}

String getValue(String data, char separator, int index)

{

    int found = 0;

    int strIndex[] = { 0, -1 };

```

```

int maxIndex = data.length() - 1;

for (int i = 0; i <= maxIndex && found <= index; i++) {

    if (data.charAt(i) == separator || i == maxIndex) {

        found++;

        strIndex[0] = strIndex[1] + 1;

        strIndex[1] = (i == maxIndex) ? i+1 : i;

    }

}

return found > index ? data.substring(strIndex[0], strIndex[1]) : "";

```

9. TESTING :

9.1 Test cases :

Wokwi Simulation : <https://wokwi.com/projects/>

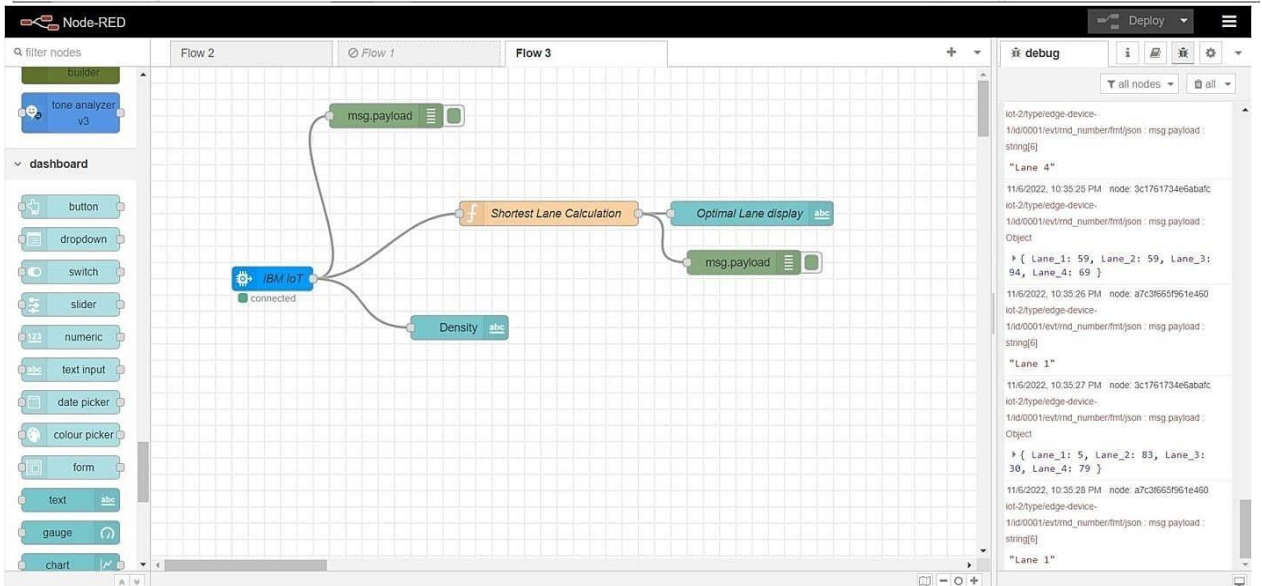
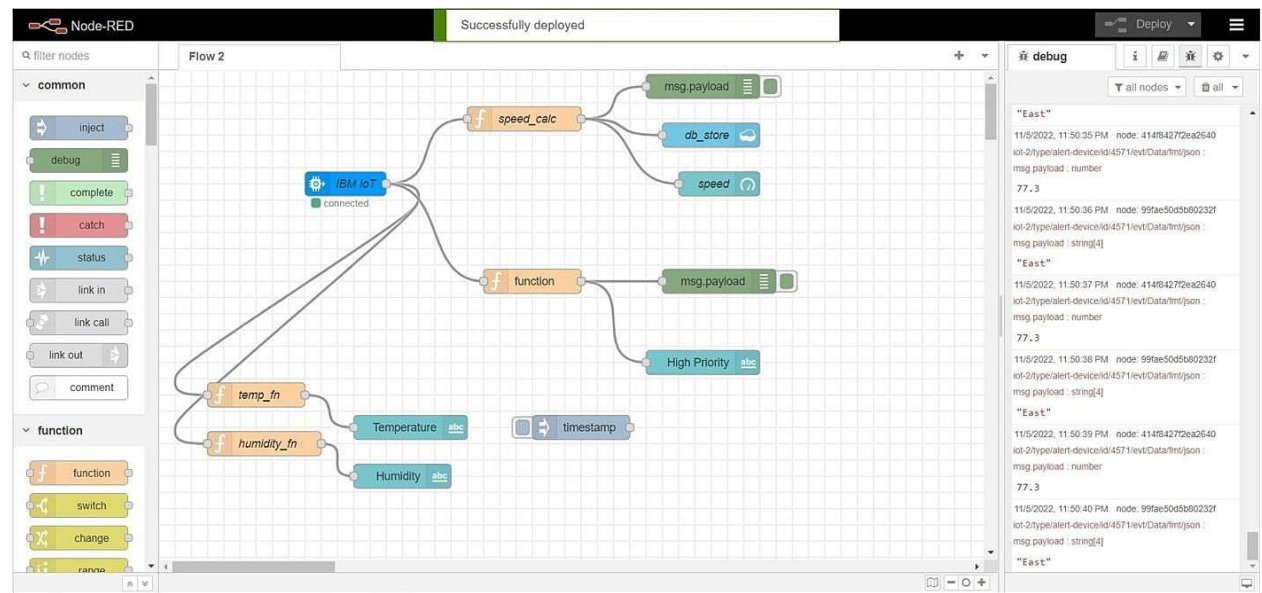
The screenshot displays the Wokwi simulation interface. On the left, the Arduino IDE sketch is open, showing code for an ESP8266 module connected to a DHT11 sensor. The code includes libraries for WiFi, PubSubClient, and DHT. It defines the DHT pin and type, creates a DHT instance, and sets up a callback function for MQTT. The code also includes credentials for IBM Watson IoT Platform and a client ID. On the right, the simulation window shows a 3D model of the ESP8266 module and DHT11 sensor. Below the model, the simulation results are displayed, showing the current temperature and humidity readings, and the status of the MQTT connection.

```

temp:37.40
humidity:86.00
Sending payload:
{"temp":37.40,"humidity":86.00,"North":0,"South":0,"East":0,"West":0}
Publish ok
Reconnecting client to psh4py.messaging.internetofthings.ibmcloud.com
.....

```

Node Red :



Edit function node

Delete Cancel Done

Properties

Name Shortest Lane Calculation

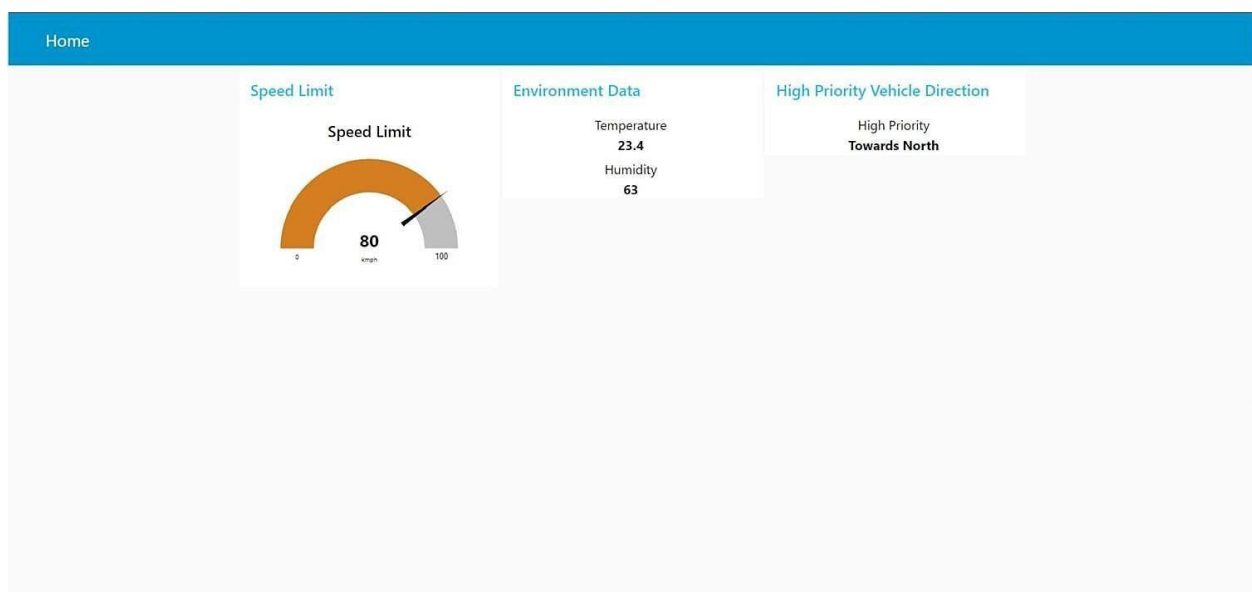
Setup On Start **On Message** On Stop

```

1 var l1 = msg.payload.Lane_1;
2 var l2 = msg.payload.Lane_2;
3 var l3 = msg.payload.Lane_3;
4 var l4 = msg.payload.Lane_4;
5
6 mini = Math.min(l1,l2,l3,l4);
7
8 res = "-";
9
10 switch(mini) {
11   case l1: res = "Lane 1"; break;
12   case l2: res = "Lane 2"; break;
13   case l3: res = "Lane 3"; break;
14   case l4: res = "Lane 4"; break;
15 }
16
17 msg.payload = res;
18
19 return msg;

```

Node Red Web UI :



The image shows a Node-RED dashboard on the left and a Wokwi simulation environment on the right.

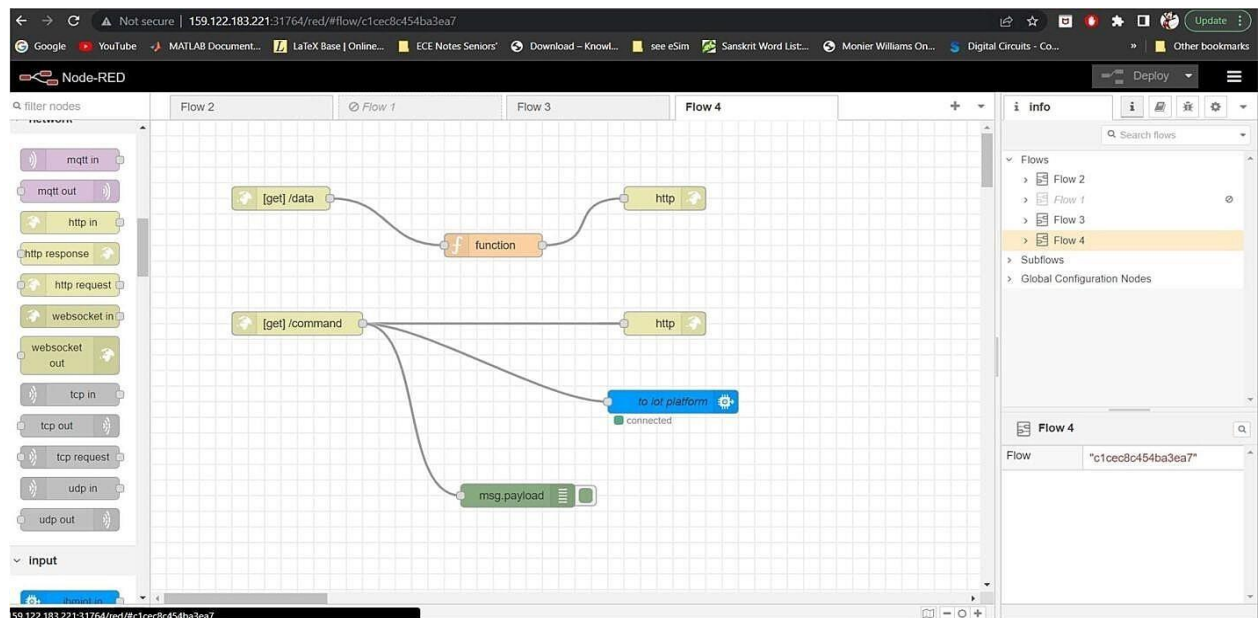
Node-RED Dashboard:

- Speed Limit:** A gauge showing a value of 69.7 mph, with a scale from 0 to 100.
- Environment Data:**
 - Temperature: 14.7
 - Humidity: 88
 - High Priority Vehicle Direction: High Priority Towards East

Wokwi Simulation:

- Sketch:** A sketch of an ESP32 microcontroller connected to a DHT22 temperature and humidity sensor.
- Simulation:** The simulation is running, showing the sensor's output. The temperature is 14.7°C and the humidity is 88.0%.
- Code:** The sketch code is visible, showing the setup and loop functions. The loop function publishes the sensor data to a topic.

Node Red - Connect with MIT APP Inventor :



Edit function node

Delete

Cancel

Done

Properties

Name

Name

Setup

On Start

On Message

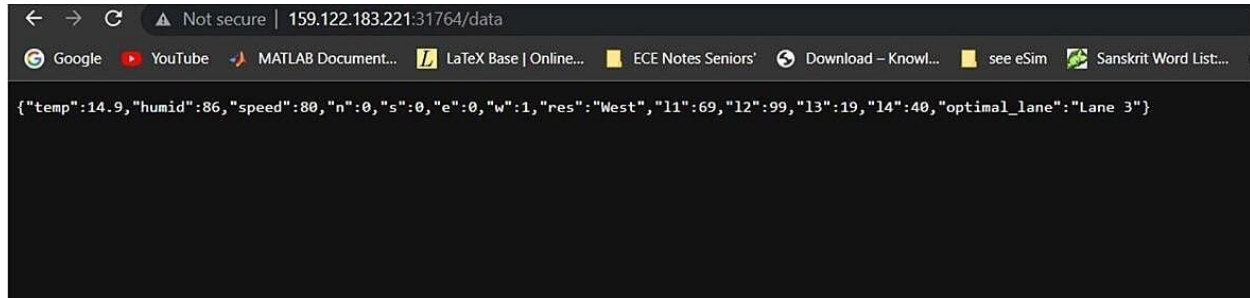
On Stop

```

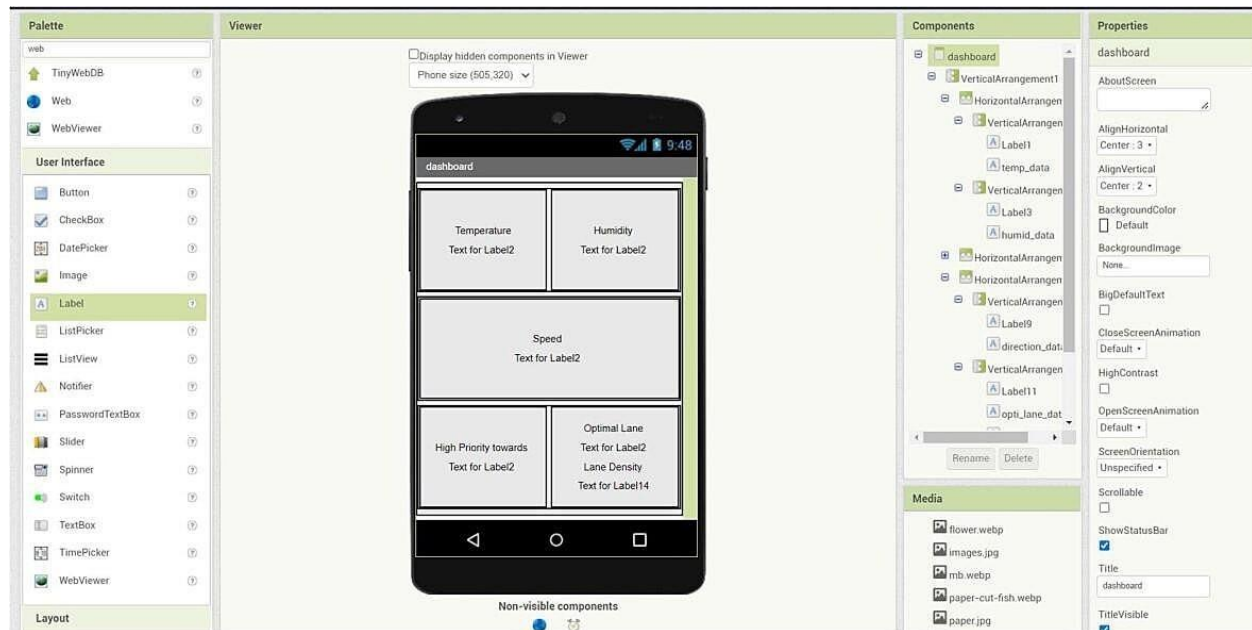
1 msg.payload = {
2   "temp":global.get("temp"),
3   "humid":global.get("humid"),
4   "speed":global.get("speed"),
5   "n":global.get("n"),
6   "s":global.get("s"),
7   "e":global.get("e"),
8   "w":global.get("w"),
9   "res":global.get("res"),
10  "l1":global.get("l1"),
11  "l2":global.get("l2"),
12  "l3":global.get("l3"),
13  "l4":global.get("l4"),
14  "optimal_lane":global.get("optimal_lane")
15 };
16 ^
17
18 return msg;

```

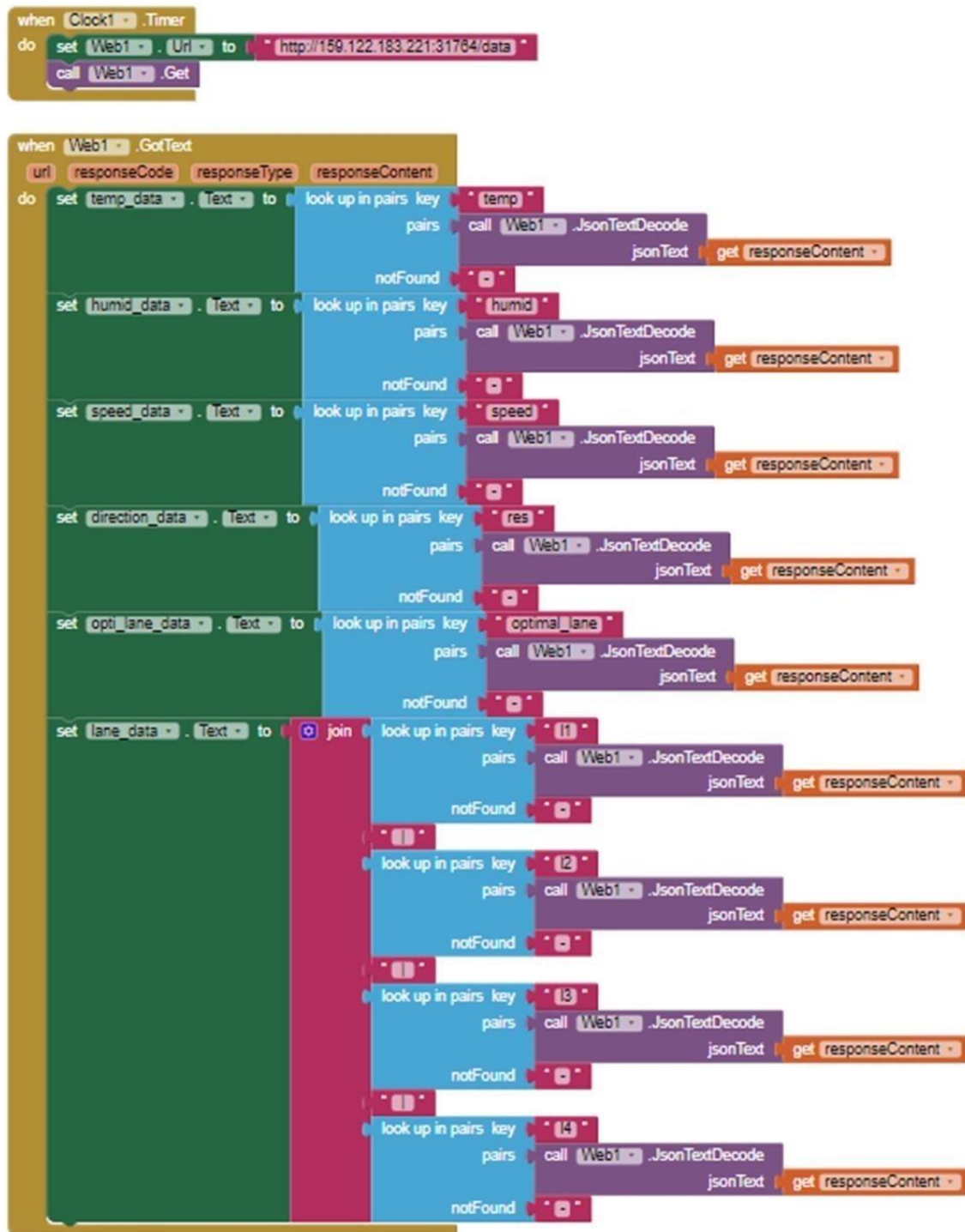

Output from Node red :



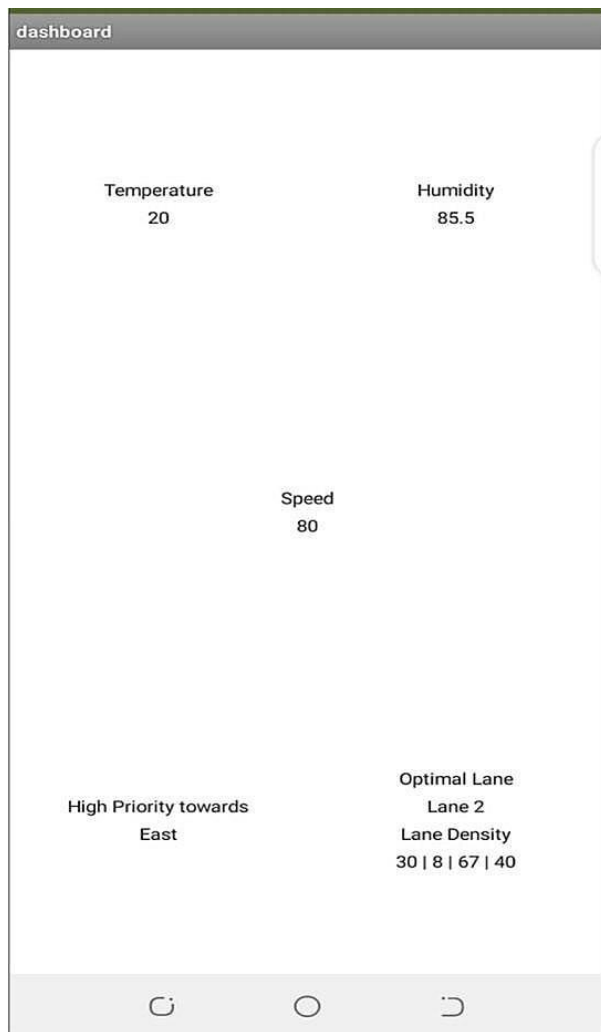
MIT App Inventor UI design:



MIT App Inventor Backend design:



(OUTPUT) Display from MIT App:



9.2 User Acceptance Testing :

Python Simulation :

```

RandomValues.py - E:/IBM/Others/Project Development Phase/Sprint 3/RandomValues.py (3.6.5)
File Edit Format Run Options Window Help

import wiotp.sdk.device
import time
import random
import ibmiotf.application
import ibmiotf.device
import requests, json

myConfig = {
    #Configuration
    "identity": {
        "orgId": "n6r19n",
        "typeId": "NodeMCU",
        "deviceId": "621319106312"
    },
    #API Key
    "auth": {
        "token": "9876543210"
    }
}

#Receiving callbacks from IBM IOT platform
def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data["command"])
    m=cmd.data["command"]

```

Import wiotp-sdk & ibmiotf :

```

Command Prompt

C:\Users\DHILEEP>pip install wiotp-sdk
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
Defaulting to user installation because normal site-packages is not writeable
Collecting wiotp-sdk
  Downloading wiotp-sdk-0.11.0.tar.gz (96 kB)
    | 96 kB 294 kB/s
  Preparing metadata (setup.py) ... done
Collecting iso8601>=0.1.12
  Downloading iso8601-1.1.0-py3-none-any.whl (9.9 kB)
Requirement already satisfied: pytz>=2018.9 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from wiotp-sdk) (2021.3)
Collecting pyyaml>=3.13
  Downloading PyYAML-6.0-cp36-cp36m-win_amd64.whl (153 kB)
    | 153 kB 2.2 MB/s
Requirement already satisfied: paho-mqtt>=1.5.0 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from wiotp-sdk) (1.6.1)
Requirement already satisfied: requests>=2.21.0 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from wiotp-sdk) (2.27.1)
Collecting requests-toolbelt>=0.8.0
  Downloading requests-toolbelt-0.10.1-py2.py3-none-any.whl (54 kB)
    | 54 kB 61 kB/s
Requirement already satisfied: charset-normalizer~>2.0.0 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from requests>=2.21.0->wiotp-sdk) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from requests>=2.21.0->wiotp-sdk) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from requests>=2.21.0->wiotp-sdk) (2022.9.24)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from requests>=2.21.0->wiotp-sdk) (1.26.12)
Using legacy 'setup.py install' for wiotp-sdk, since package 'wheel' is not installed.
Installing collected packages: requests-toolbelt, pyyaml, iso8601, wiotp-sdk
  Running setup.py install for wiotp-sdk ... done
Successfully installed iso8601-1.1.0 pyyaml-6.0 requests-toolbelt-0.10.1 wiotp-sdk-0.11.0

```

```

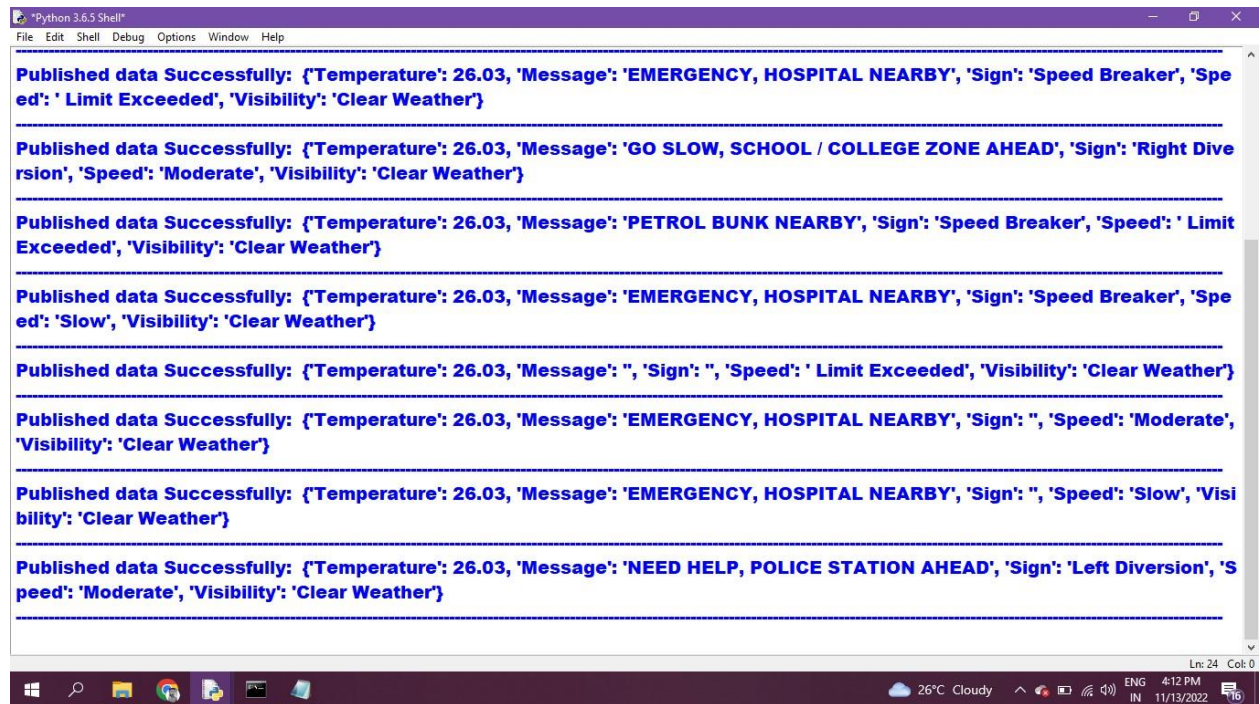
C:\Users\DHILEEP>pip install ibmiotf
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
Defaulting to user installation because normal site-packages is not writeable
Collecting ibmiotf
  Downloading ibmiotf-0.4.0.tar.gz (71 kB)
    | 71 kB 13 kB/s
  Preparing metadata (setup.py) ... done
Requirement already satisfied: iso8601>=0.1.12 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from ibmiotf) (1.1.0)
Requirement already satisfied: pytz>=2017.3 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from ibmiotf) (2021.3)
Requirement already satisfied: paho-mqtt>=1.3.1 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from ibmiotf) (1.6.1)
Requirement already satisfied: requests>=2.18.4 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from ibmiotf) (2.27.1)
Requirement already satisfied: requests_toolbelt>=0.8.0 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from ibmiotf) (0.10.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from requests>=2.18.4->ibmiotf) (2022.9.24)
Requirement already satisfied: idna<4,>=2.5 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from requests>=2.18.4->ibmiotf) (3.4)
Requirement already satisfied: charset-normalizer<=2.0.0 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from requests>=2.18.4->ibmiotf) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\dhileep\appdata\roaming\python\python36\site-packages (from requests>=2.18.4->ibmiotf) (1.26.12)
Using legacy 'setup.py install' for ibmiotf, since package 'wheel' is not installed.
Installing collected packages: ibmiotf
  Running setup.py install for ibmiotf ... done
Successfully installed ibmiotf-0.4.0

```

OpenWeatherMap - (Ex., Salem, IN) :

The screenshot shows the OpenWeatherMap website interface. At the top, there's a navigation bar with links like 'Weather in your city', 'Guide', 'API', 'Dashboard', 'Marketplace', 'Pricing', 'Maps', 'Our Initiatives', 'Partners', 'Blog', 'For Business', 'Nithy...', and 'Support'. Below this, a large orange banner contains a search bar with 'Salem, IN' entered and a 'Search' button. Under the banner, the weather for Salem, IN is displayed as 'overcast clouds' with a temperature of 26.9°C. Additional details include 'temperature from 26.9 to 26.9 °C, wind 3 m/s, clouds 94 %, 1009 hpa' and 'Geo coords [11.65, 78.1667]'. A section titled 'Search engine is very flexible. How it works:' follows, with a bullet point explaining that users should provide the city name, a comma, and a 2-letter country code (ISO3166) for precise results, with examples like 'London, GB' or 'New York, US'.

Python IDLE Output :



```

Python 3.6.5 Shell
File Edit Shell Debug Options Window Help

Published data Successfully: {'Temperature': 26.03, 'Message': 'EMERGENCY, HOSPITAL NEARBY', 'Sign': 'Speed Breaker', 'Speed': 'Limit Exceeded', 'Visibility': 'Clear Weather'}

Published data Successfully: {'Temperature': 26.03, 'Message': 'GO SLOW, SCHOOL / COLLEGE ZONE AHEAD', 'Sign': 'Right Diversion', 'Speed': 'Moderate', 'Visibility': 'Clear Weather'}

Published data Successfully: {'Temperature': 26.03, 'Message': 'PETROL BUNK NEARBY', 'Sign': 'Speed Breaker', 'Speed': 'Limit Exceeded', 'Visibility': 'Clear Weather'}

Published data Successfully: {'Temperature': 26.03, 'Message': 'EMERGENCY, HOSPITAL NEARBY', 'Sign': 'Speed Breaker', 'Speed': 'Slow', 'Visibility': 'Clear Weather'}

Published data Successfully: {'Temperature': 26.03, 'Message': '', 'Sign': '', 'Speed': 'Limit Exceeded', 'Visibility': 'Clear Weather'}

Published data Successfully: {'Temperature': 26.03, 'Message': 'EMERGENCY, HOSPITAL NEARBY', 'Sign': '', 'Speed': 'Moderate', 'Visibility': 'Clear Weather'}

Published data Successfully: {'Temperature': 26.03, 'Message': 'EMERGENCY, HOSPITAL NEARBY', 'Sign': '', 'Speed': 'Slow', 'Visibility': 'Clear Weather'}

Published data Successfully: {'Temperature': 26.03, 'Message': 'NEED HELP, POLICE STATION AHEAD', 'Sign': 'Left Diversion', 'Speed': 'Moderate', 'Visibility': 'Clear Weather'}

Ln: 24 Col: 0

```

10. RESULT:

10.1 Performance Metrics:

The result shows three switches through which you can switch the display to different modes.

Mode1 : Displaying Speed Limit

Mode2 : Display of Diversions, Alerts of Accident prone area

Mode3 : Information sign boards

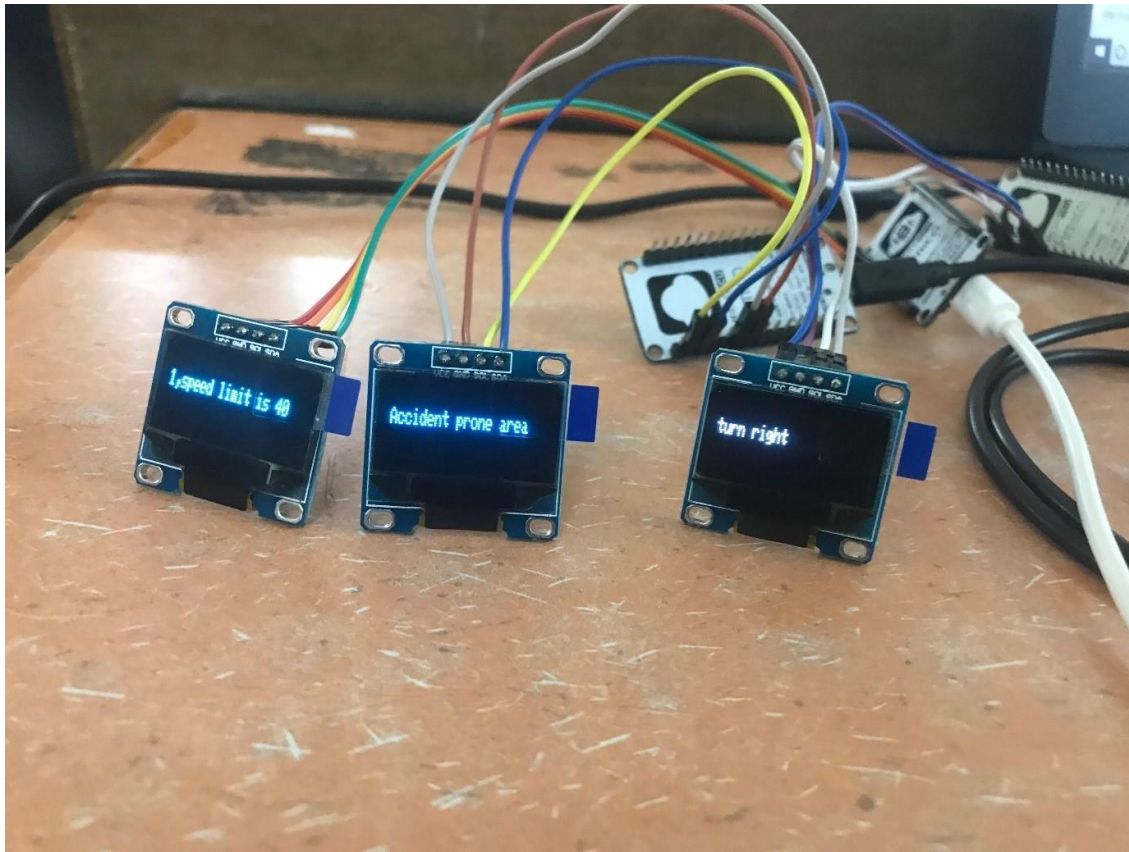


Fig 10 - This figure shows three OLEDs that we are displaying.

(1) The first OLED display shows speed limit using weather API.

(2)The second OLED display shows about the alerts of the accident prone area.

(3)The third OLED display shows the information sign boards

11. ADVANTAGES & DISADVANTAGES:

Advantages :

- Efficient Traffic Management ● Automated Toll and Ticketing
- Self-driving Cars
- Advanced Vehicle Tracking or Transportation Monitoring
- Enhanced Security of the Public Transport

Disadvantages :

- Property Damage
- Bodily Injury
- Cyber Risk

12. CONCLUSION:

Roads were previously only functional in nature. Highways are now built to be safe, longlasting, and easily accessible. The thought of a roadway being a vector for IOT networks or any other communication system was unthinkable and impractical. However, recent advances, such as the installation of digital sign boards along roadside, have provided a gateway that allows highways to function as data conveyors. Data such as road conditions and traffic patterns are now shown on sign boards. Wireless networks can use sensor technology to enable more detailed communications at higher levels. IoT systems could be used by state and local transportation departments to target road maintenance needs, traffic utilisation, weather conditions, and accident records.

13.FUTURE SCOPE:

1. *Solar powered roadways*

Photovoltaic cells are embedded within hexagonal panels made of tempered glass, which are used to pave roads. These panels contain LEDs, microprocessors, snow-melting heating devices and inductive charging capability for electric vehicles when driving. Glass is renewable and can be engineered to be stronger than steel, and to allow cars to stop safely even when traveling at high speeds. While this idea has gained widespread support, scalability is a challenge as it remains expensive.

2. *Smart Roads*

Specially engineered roadways fitted with smart features, including sensors that monitor and report changing road conditions, and WiFi transmitters that provide broadband services to vehicles, homes and businesses. The smart road can also charge electric cars as they drive.

3. *Glow in the dark roads*

Glowing markers painted onto existing roadway surfaces use a photo-luminescent powder that absorbs and stores daylight. The 500m long strips glow for 8 hours after dark. This technology is still in the testing phase, and the glow is not yet consistent, but it could be more cost-effective than traditional road lighting technologies.

4. *Interactive lights*

Road lights activated by motion sensors to illuminate a particular section of the road as cars approach. The lights dim once the car passes. Suited for roads with less traffic, interactive lights provide night visibility as needed and reduce energy wastage when there are no cars. One design, developed in Holland, uses the wind generated by passing vehicles to power lights.

5. *Electric priority lane for charging electric vehicles*

Embedded cables generate magnetic fields that charge electric vehicles while driving. A receiver coil in the vehicle picks up electromagnetic oscillations from a transmitter coil embedded in the road and converts them to AC, which can then power the car. Inductive charging technology already exists for static cars, but future wireless technology could charge batteries while in motion, providing distance range solutions for electric vehicles which travel longer journeys.

6. *Weather detection*

Networks of AI-integrated sensors detect weather conditions that impact road safety. Road Weather Information Systems (RWIS) in use today are limited because they only collect data from a small set of weather stations. A larger future network could use automated weather stations to collect atmospheric and weather data and instantly upload it to the cloud. Dynamic temperature-sensitive paint could be used to highlight invisible roadway conditions like black ice.

7. *Traffic detection*

Data that helps travelers plan their routes. Sensors lining highways monitor traffic flow and weight load, warn drivers of traffic jams, and automatically alert the authorities about accidents. Fiber-optic cables embedded in the road detect wear and tear, and communication between vehicles and roads can improve traffic management. For example, rapid flow technologies use artificial intelligence (AI) to manage traffic lights, which respond to each other and to cars. Traditional systems were pre-programmed to optimize flow around peak journey times, new technologies are able to process and optimize flows in real time.

14. APPENDIX :

Source code :

Code to print the random temperature, Road signs, Speed limit, Message:

(RandomValues.py)

```

import wiotp.sdk.device
import time import random
import ibmiotf.application
import ibmiotf.device
import requests, json

myConfig = {
    #Configuration
    "identity": {

        "orgId": "n6rl9n",
        "typeId": "NodeMCU",
        "deviceId": "621319106312"
    },
    #API Key "auth":
    {
        "token": "9876543210"
    }
}

#Receiving callbacks from IBM IOT platform

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()

#OpenWeatherMap Credentials

BASE_URL = "https://api.openweathermap.org/data/2.5/weather?"
CITY = "Salem, IN"
URL = BASE_URL + "q=" + CITY + "&units=metric"&"&appid=" +
"f58e4720c739a54c439aba9b05176839"

while True:
    response = requests.get(URL)
    if response.status_code == 200:

```

```

        data = response.json()
        main = data['main']
        temperature = main['temp']
humidity = main['humidity']
        pressure = main['pressure']
        report = data['visibility']

#messge part

msg=random.randint(0,5) if
msg==1:
    message="GO SLOW, SCHOOL ZONE AHEAD"
elif msg==2:
    message="NEED HELP, POLICE STATION AHEAD"
elif msg==3:
    message="EMERGENCY, HOSPITAL NEARBY" elif
msg==4:
    message="DINE IN, RESTAURENT AVAILABLE"
elif msg==5:
    message="PETROL BUNK NEARBY"
else:
    message=""

#Speed Limit part

speed=random.randint(0,150)
if speed>=100:
    speedMsg=" Limit Exceeded"
elif speed>=60 and speed<100:
    speedMsg="Moderate" else:
    speedMsg="Slow"
#Diversion part

sign=random.randint(0,5)
if sign==1:
    signMsg="Right Diversion" elif
sign==2:
    signMsg="Speed Breaker"
elif sign==3:
    signMsg="Left Diversion" elif
sign==4:
    signmsg="U Turn"

```

```

else:
    signMsg=""

#Visibility

if temperature < 24:
    visibility="Fog Ahead, Drive Slow"
elif temperature < 20:
    visibility="Bad Weather" else:
    visibility="Clear Weather" else:
    print("Error in the HTTP request")

myData={'Temperature':temperature, 'Message':message, 'Sign':signMsg, 'Speed':speedMsg,
'Visibility':visibility}
client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)

#PUBLISHING TO IOT WATSON

print("Published data Successfully: ", myData)
client.commandCallback = myCommandCallback
time.sleep(5)
client.disconnect()

```

GITHUB LINK:

<https://github.com/IBM-EPBL/IBM-Project-31649-1660203842>

VIDEO LINK:

<https://youtu.be/7-5xlqTAjAA>