Signs with Smart Connectivity for Better Road Safety

IOT Based Project

Project report by

AJAI BALAJI P	- 111519106003
DHEEPAK KUMAAR A S	- 111519106026
GOWSHIK RAM S	- 111519106041
VAMSI KRISHNA D	- 111519106030

Date: 23 November 2022
Team ID: PNT2022TMID14814

Bachelor of Engineering
In
R.M.D. Engineering College
Kavaraipettai

Academic Year: 2022-2023

TABLE OF CONTENTS

1. INTRODUCTION:

Traffic has recently become a big issue for the people of India. As a result, it wastes valuable time, fuel, and electricity. The Internet of Things (IOT) is a network of electrical appliances, cars, physical devices, and other items that are integrated with electronics, actuators, sensors, software, and connectivity, allowing these objects to connect and share data. Each object is uniquely identified by its embedded computing system, but it may interact with the existing Internet infrastructure.

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:

The individual traffic signals are connected with traffic control system to perform network wide traffic operation. These control systems contain a central computer, a communication network, and intersection traffic signals. Coordination of control system can be implemented through different techniques like time-base, hardwired interconnection method. Coordination between traffic signals and agencies requires the development of data sharing and traffic signal control agreements. A traffic-signal system has only one purpose i.e. to deliver signal timings to the driver. The system provides features that improve the traffic engineer's ability to achieve this goal. These are primarily access features. They provide access to the intersection signal controller for maintenance and operations. The more complete and convenient the access, the more efficient the operator will be and the more effective the system. In addition to control the traffic signals, modern technology also provide surveillance capabilities, including different kinds of video surveillance and traffic detection.

2.2 Reference:

1.https:/www.hindawi.com/journals/jat/2022/58296007/

2. https:/www.powerbulbs.com/us/blog/2020/01/yellow-or-whiter-light

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 and Proposed Solution:

3.1 Empathy Map Canvas:

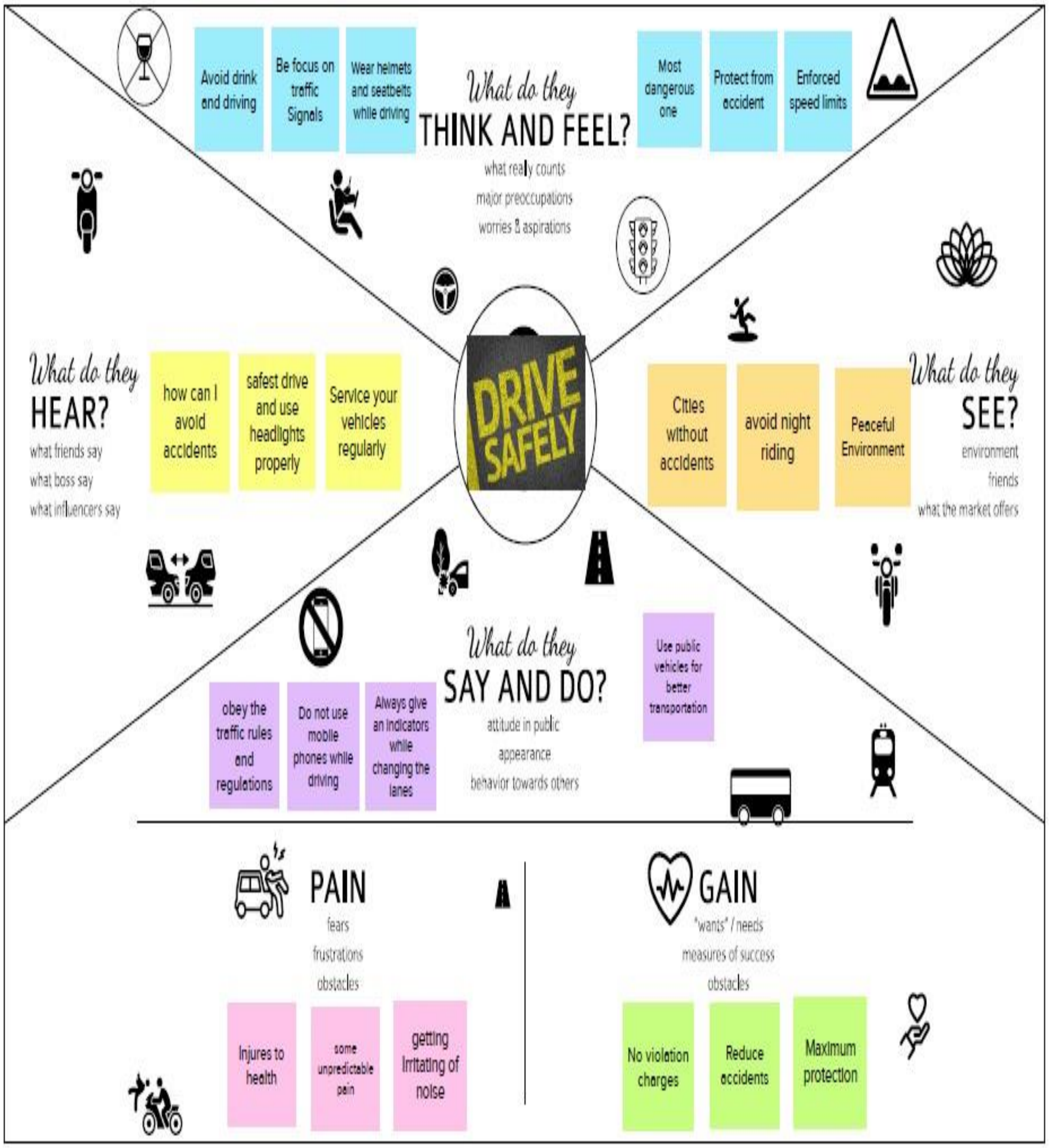


Fig 3.1 Empathy Map

3.2 Ideation and Brainstorming:

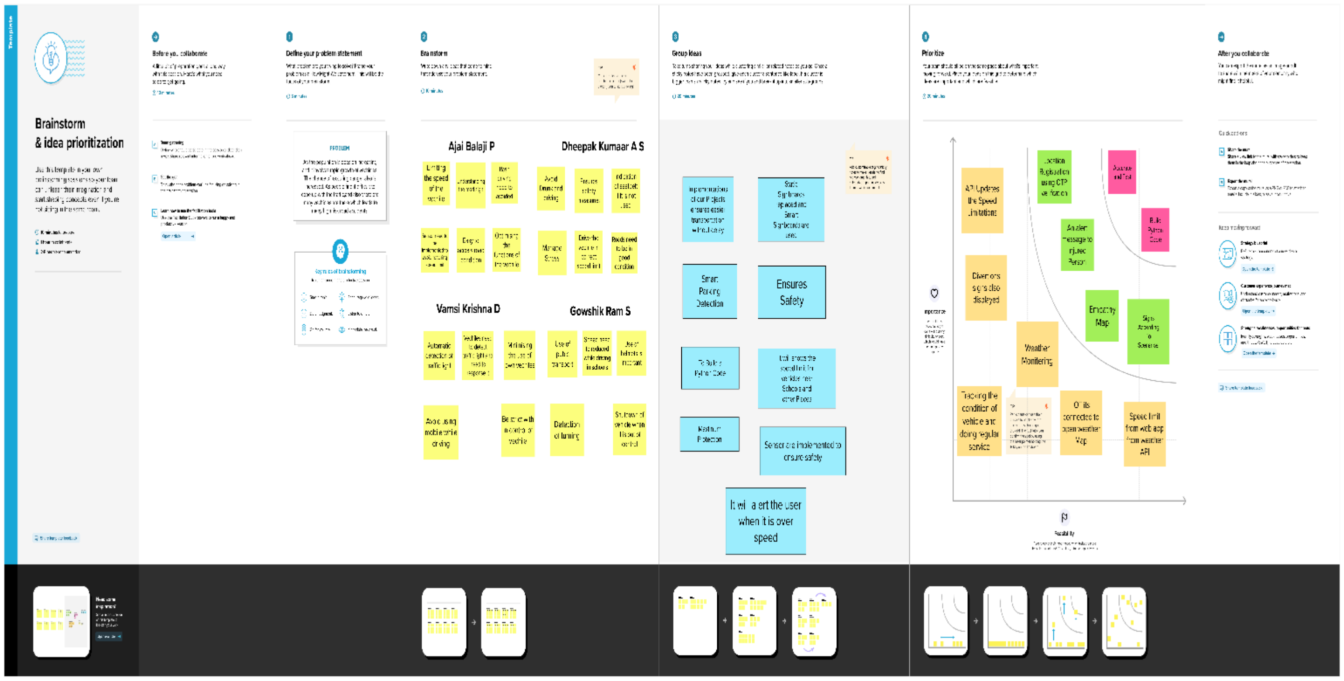


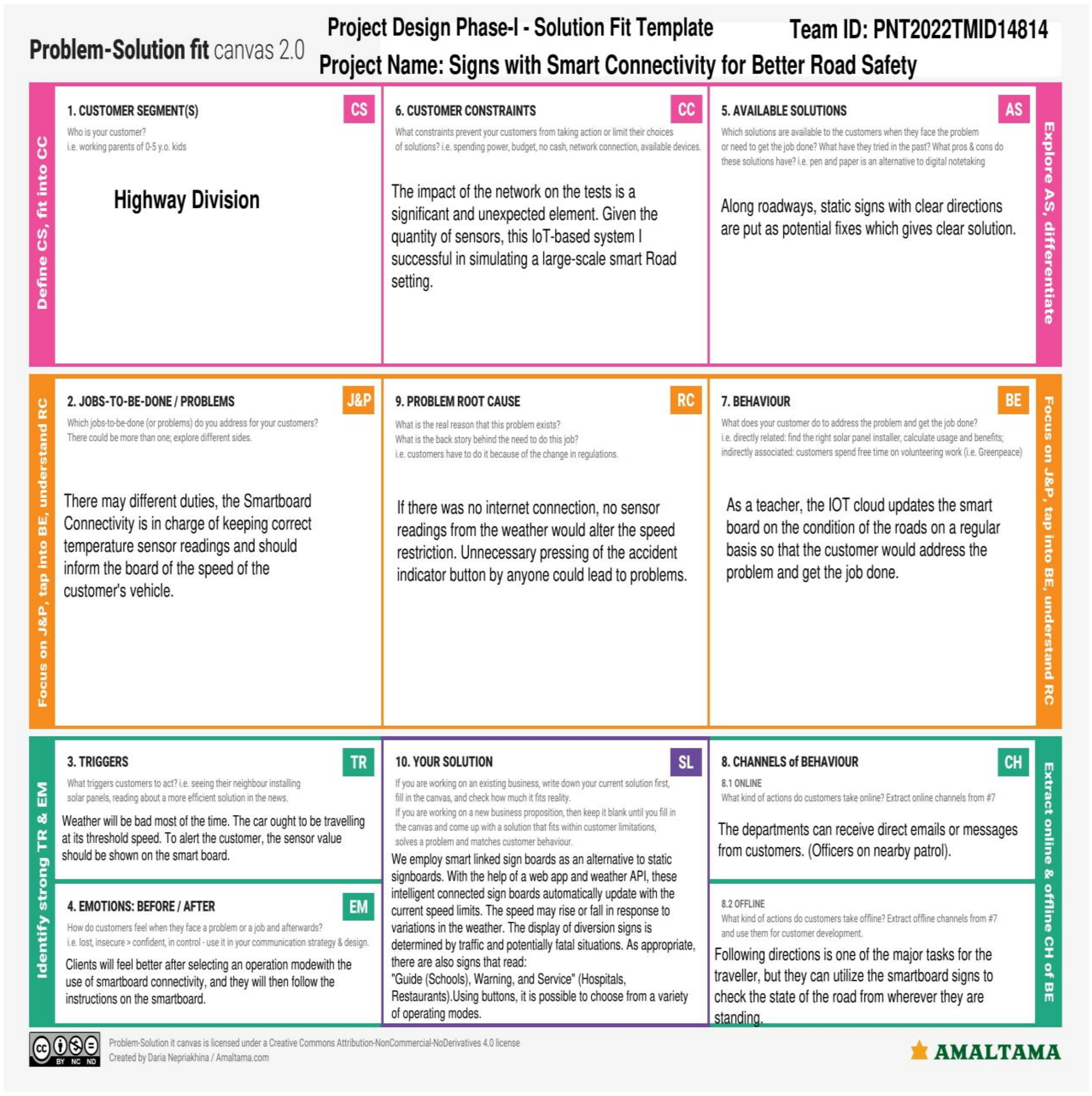
Fig 3.2 Ideation and Brainstorming

3.3 Proposed solution:

S.No.	**Parameter**	**Description**
1.	Problem Statement (Problem to be solved)	➢ To prevent the road accidents from happening using IOT.
2.	Idea / Solution description	➢ By Preparing smart signs using IOT instead of regular signs hung on the road. ➢ Smart signs are built with IOT and LED are used.
3.	Novelty / Uniqueness	➢ Since LED'S are used it will be visible. ➢ The smart signs consist of temperature, humidity, wind speed. ➢ This information is received from weather monitoring app. ➢ It also gives information about nearby Places such as hospitals, schools, etc. so that the users can decide their speeding according to that information.
4.	Social Impact / Customer Satisfaction	➢ These create a noticeable impact on the road safety department. ➢ By deciding a speed limit for the user, there is significant chance in reducing the accidents.
5.	Business Model (Revenue Model)	➢ By executing these for commoners by the government, it is great initiative in creating an awareness among the people.

		➢ A separate budget can be allotted for this by the government, which paves a way for a safer environment.
6.	Scalability of the Solution	➢ It has greater chance in reducing the risk for the people as it is more visible than the normal signs, which saves a lot of lives at stake.

3.4 Problem Solution Fit:



Problem-Solution fit canvas 2.0

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMID14814

Project Name: Signs with Smart Connectivity for Better Road Safety

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

Highway Division

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

The impact of the network on the tests is a significant and unexpected element. Given the quantity of sensors, this IoT-based system I successful in simulating a large-scale smart Road setting.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Along roadways, static signs with clear directions are put as potential fixes which gives clear solution.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

There may different duties, the Smartboard Connectivity is in charge of keeping correct temperature sensor readings and should inform the board of the speed of the customer's vehicle.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

If there was no internet connection, no sensor readings from the weather would alter the speed restriction. Unnecessary pressing of the accident indicator button by anyone could lead to problems.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

As a teacher, the IOT cloud updates the smart board on the condition of the roads on a regular basis so that the customer would address the problem and get the job done.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

Weather will be bad most of the time. The car ought to be travelling at its threshold speed. To alert the customer, the sensor value should be shown on the smart board.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Clients will feel better after selecting an operation modewith the use of smartboard connectivity, and they will then follow the instructions on the smartboard.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

We employ smart linked sign boards as an alternative to static signboards. With the help of a web app and weather API, these intelligent connected sign boards automatically update with the current speed limits. The speed may rise or fall in response to variations in the weather. The display of diversion signs is determined by traffic and potentially fatal situations. As appropriate, there are also signs that read:
"Guide (Schools), Warning, and Service" (Hospitals, Restaurants).Using buttons, it is possible to choose from a variety of operating modes.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

The departments can receive direct emails or messages from customers. (Officers on nearby patrol).

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

Following directions is one of the major tasks for the traveller, but they can utilize the smartboard signs to check the state of the road from wherever they are standing.

Extract online & offline CH of BE

AMALTAMA

Fig 3.4 Proposed Solution

4. Requirement Analysis:

4.1 Functional Requirements:

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 linkedsign 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 lengthof 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 with in the network, collapsing to be the real-time 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 Graph:

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.

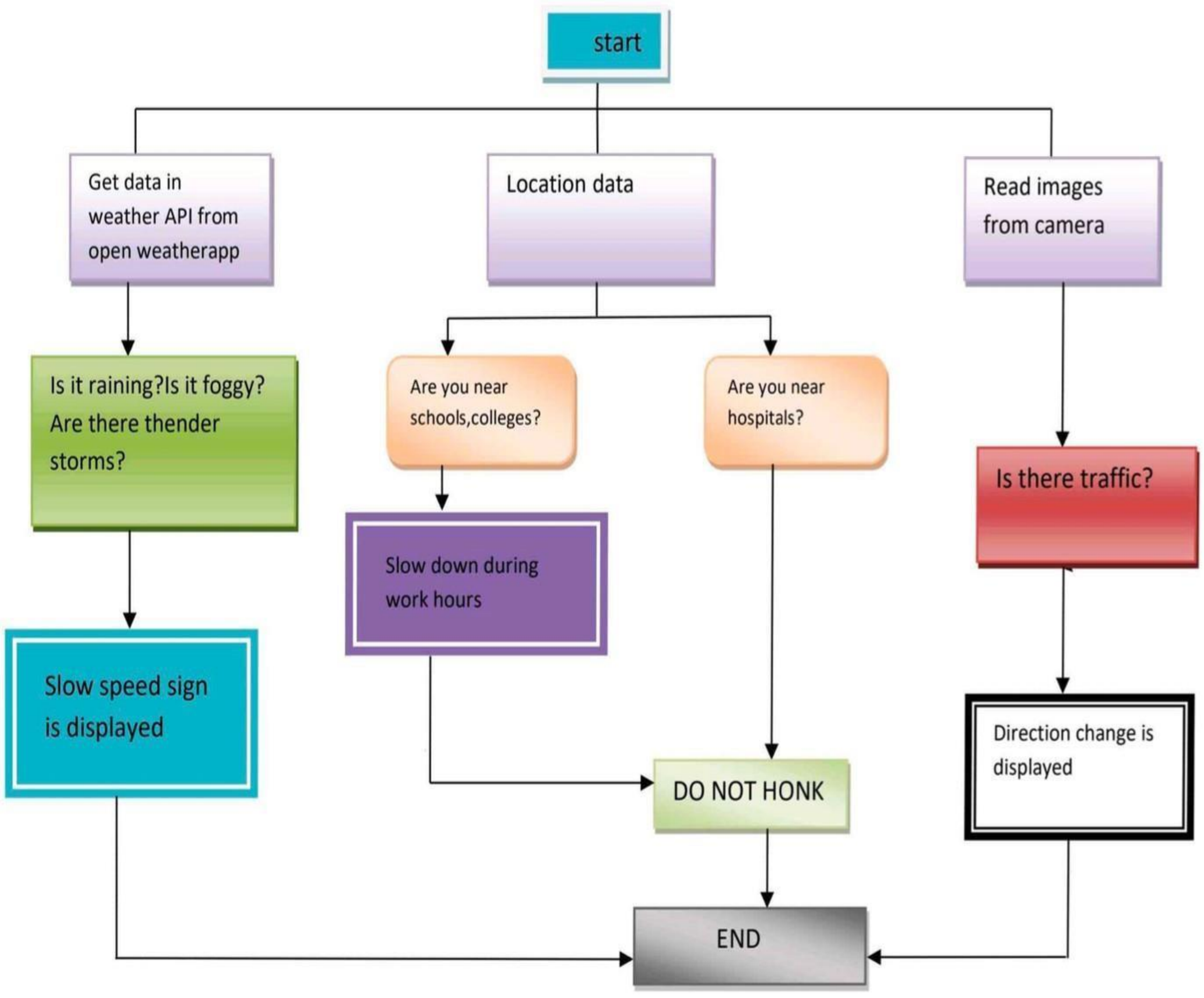
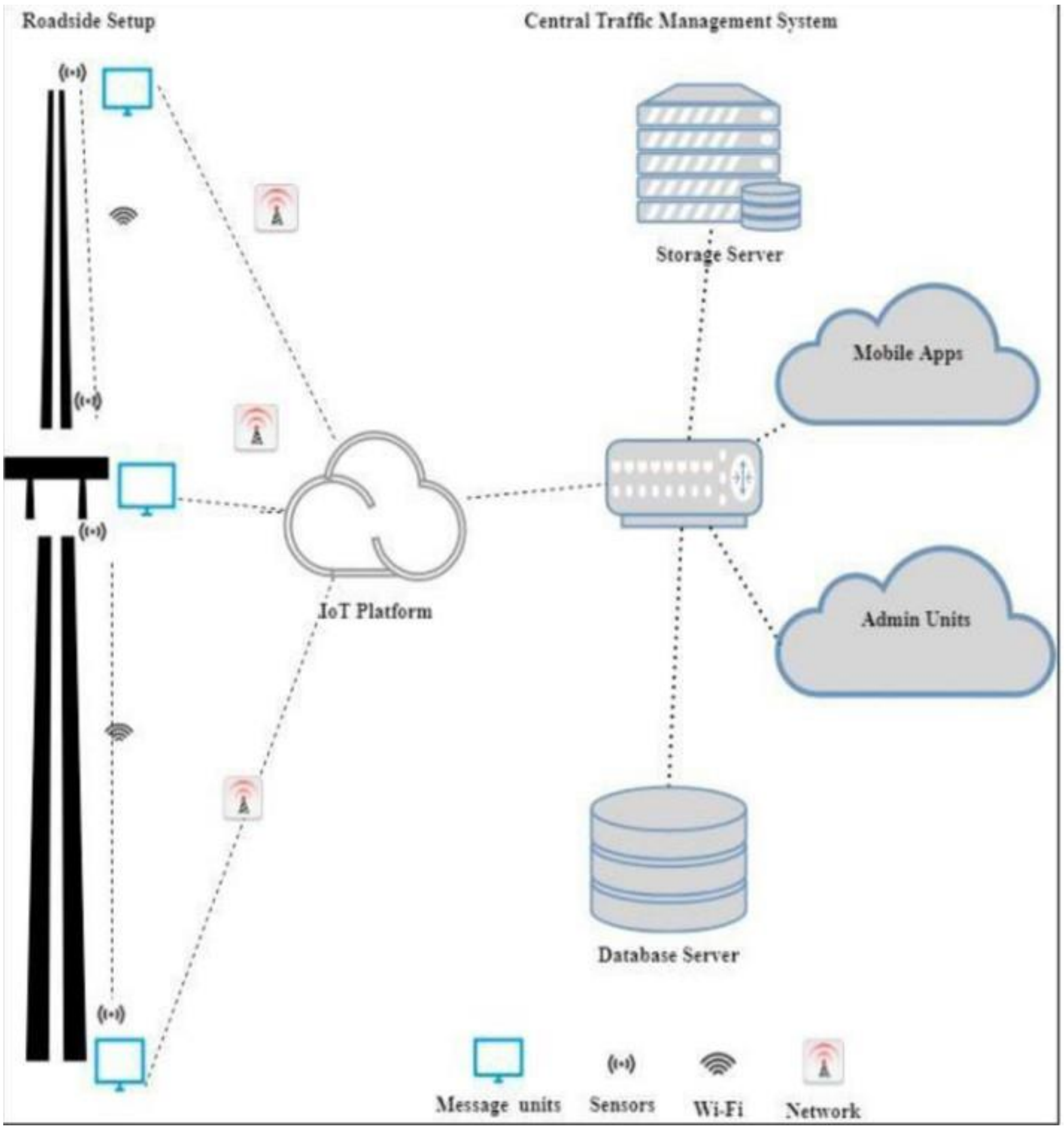

start
Get data in weather API from open weatherapp
Location data
Read images from camera
Is it raining?Is it foggy? Are there thender storms?
Are you near schools,colleges?
Are you near hospitals?
Is there traffic?
Slow down during work hours
Slow speed sign is displayed
Direction change is displayed
DO NOT HONK
END

Fig 5.1 Data flow graph

5.2 Solution and Technical Architecture:



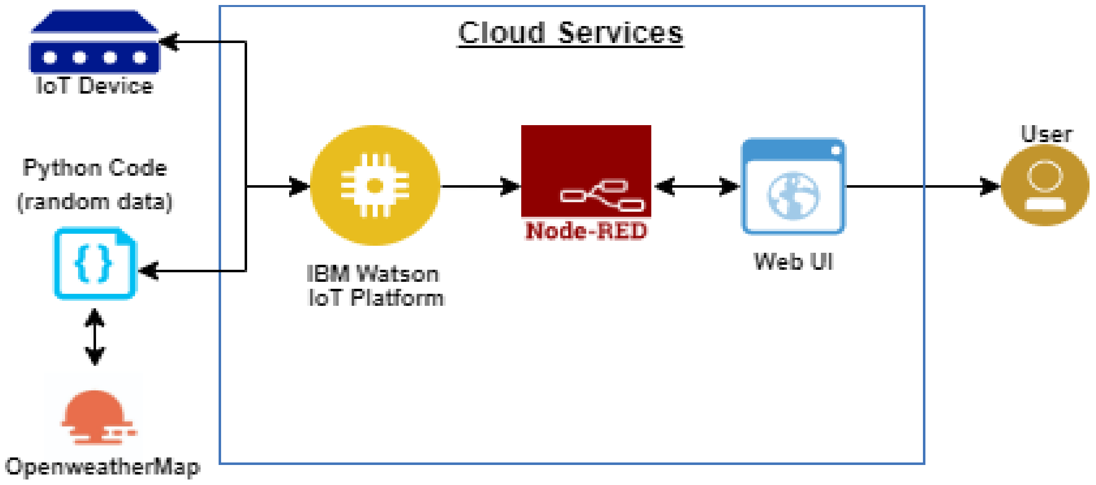


Fig 5.2 Technology architecture

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 and Technology:

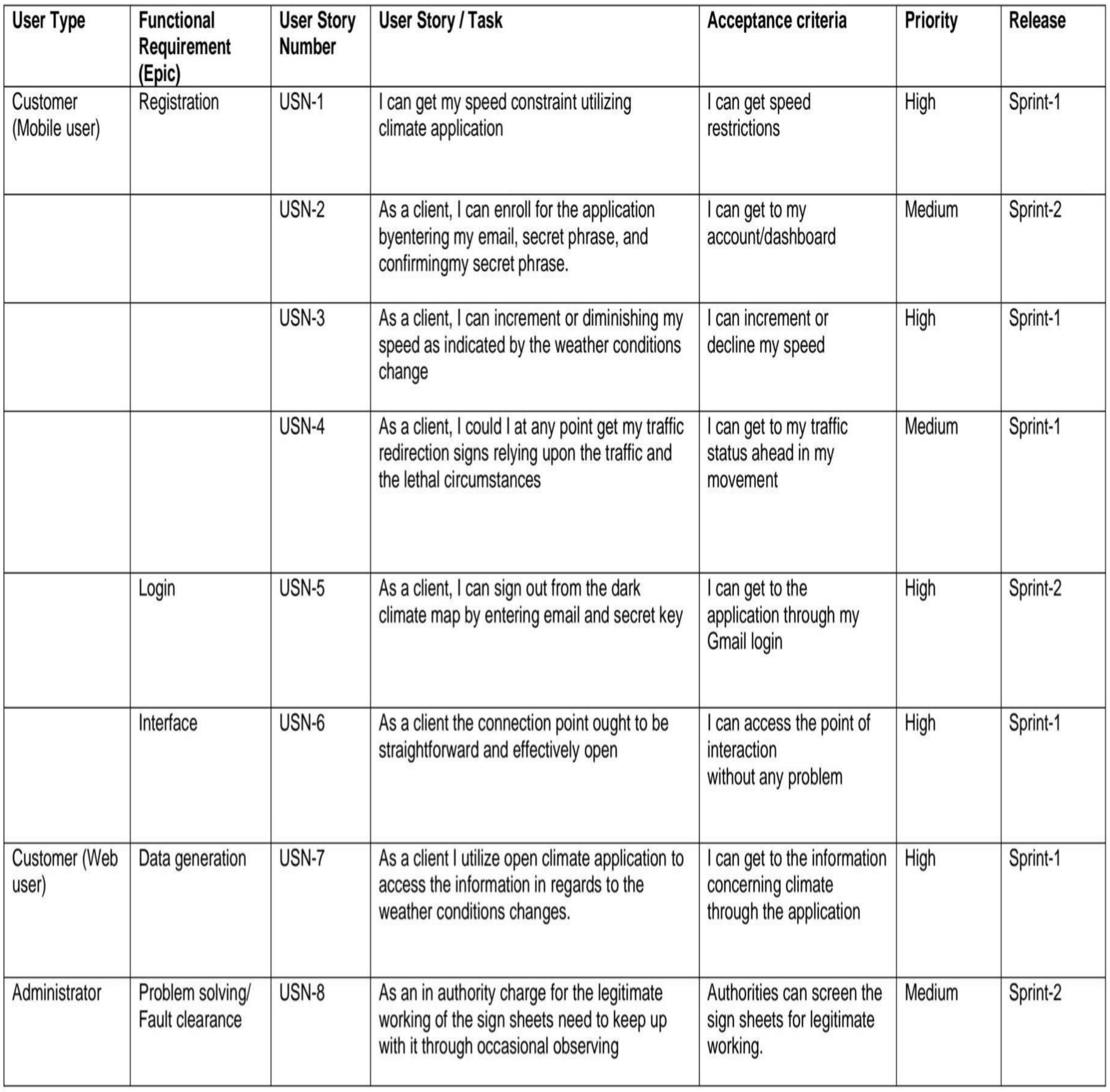
S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Jsetc.
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in 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 in to 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	I can get my speed constraint utilizing climate application	I can get speed restrictions	High	Sprint-1
		USN-2	As a client, I can enroll for the application byentering my email, secret phrase, and confirmingmy secret phrase.	I can get to my account/dashboard	Medium	Sprint-2
		USN-3	As a client, I can increment or diminishing my speed as indicated by the weather conditions change	I can increment or decline my speed	High	Sprint-1
		USN-4	As a client, I could I at any point get my traffic redirection signs relying upon the traffic and the lethal circumstances	I can get to my traffic status ahead in my movement	Medium	Sprint-1
	Login	USN-5	As a client, I can sign out from the dark climate map by entering email and secret key	I can get to the application through my Gmail login	High	Sprint-2
	Interface	USN-6	As a client the connection point ought to be straightforward and effectively open	I can access the point of interaction without any problem	High	Sprint-1
Customer (Web user)	Data generation	USN-7	As a client I utilize open climate application to access the information in regards to the weather conditions changes.	I can get to the information concerning climate through the application	High	Sprint-1
Administrator	Problem solving/ Fault clearance	USN-8	As an in authority charge for the legitimate working of the sign sheets need to keep up with it through occasional observing	Authorities can screen the sign sheets for legitimate working.	Medium	Sprint-2

6. Project Planning and Scheduling:

6.1 Sprint Planning and 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	Ajai Balaji P Dheepak Kumaar A S Vamsi Krishna D Gowshik Ram S
Sprint-1	Local Server/Software Run	Write a python program that outputs results given the inputs like weather and location.	1	MEDIUM	Ajai Balaji P Dheepak Kumaar A S Vamsi Krishna D Gowshik Ram 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	Ajai Balaji P Dheepak Kumaar A S Vamsi Krishna D Gowshik Ram S
Sprint-3	Hardware Initialization	Integrate the hardware to be able to access the cloud functions and provide inputs to the same.	2	HIGH	Ajai Balaji P Dheepak Kumaar A S Vamsi Krishna D Gowshik Ram S
Sprint-4	UI/UX Optimization & Debugging	Optimize all the shortcomings and provide better user experience.	2	LOW	Ajai Balaji P Dheepak Kumaar A S Vamsi Krishna D Gowshik Ram S

6.2 Sprint Delivery Schedule:

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	29 Oct 2022	20	15 Nov 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	15 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	15 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	15 Nov 2022	20	15 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:

- Node MCU:

New NodeMcu 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 scipt.

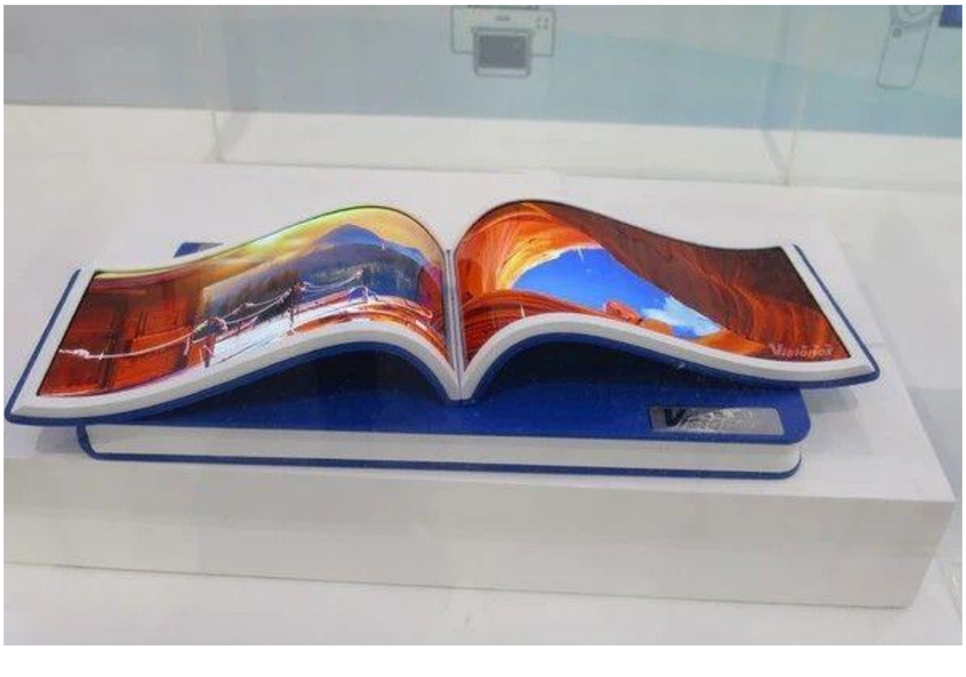
- Push Button:

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.

❖ OLED screen:

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.



8. Coding and Solution:

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 "c3wgxl"
#define DEVICE_TYPE "ajai"
#define DEVICE_ID "12345"
#define TOKEN "123456789"
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 graphicsdisplay.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 "c3wgxl"
#define DEVICE_TYPE "ajai"
#define DEVICE_ID "12345"
#define TOKEN "123456789"
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=="2"){
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 graphicsdisplay.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]) : "";
}
```

8.3 Feature 3:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h> const
char* ssid = "SB-IOT1";
const char* password = "sb@iot11";
String command1,command2;
#define ORG "c3wgxl"
#define DEVICE_TYPE "ajai"
#define DEVICE_ID "12345"
#define TOKEN "123456789"
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);
```

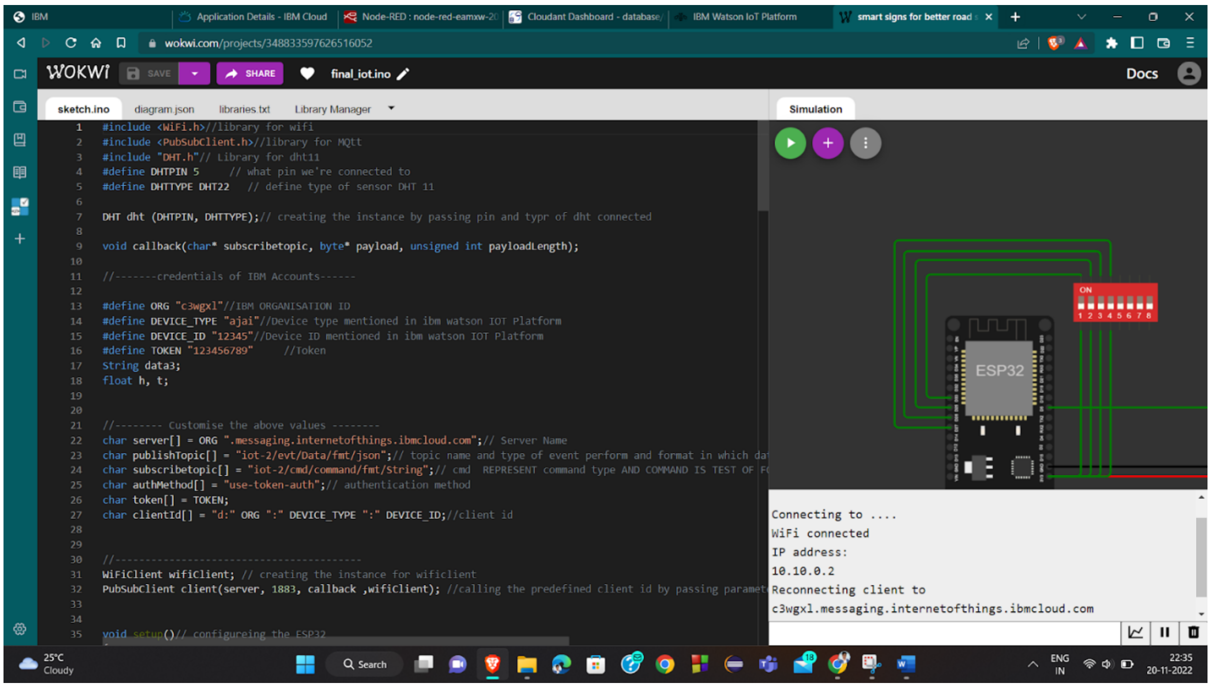
```
WiFiClient
wifiClient;
PubSubClient client(server, 1883, callback, wifiClient); void
setup() {
display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR);
Serial.begin(115200);
Serial.println();
pinMode(D1,OUTPUT);
wifiConnect();
mqttConnect();
}
void loop() {
if (!client.loop()) {
mqttConnect();
}
delay(100);
}
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 graphicsdisplay.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:

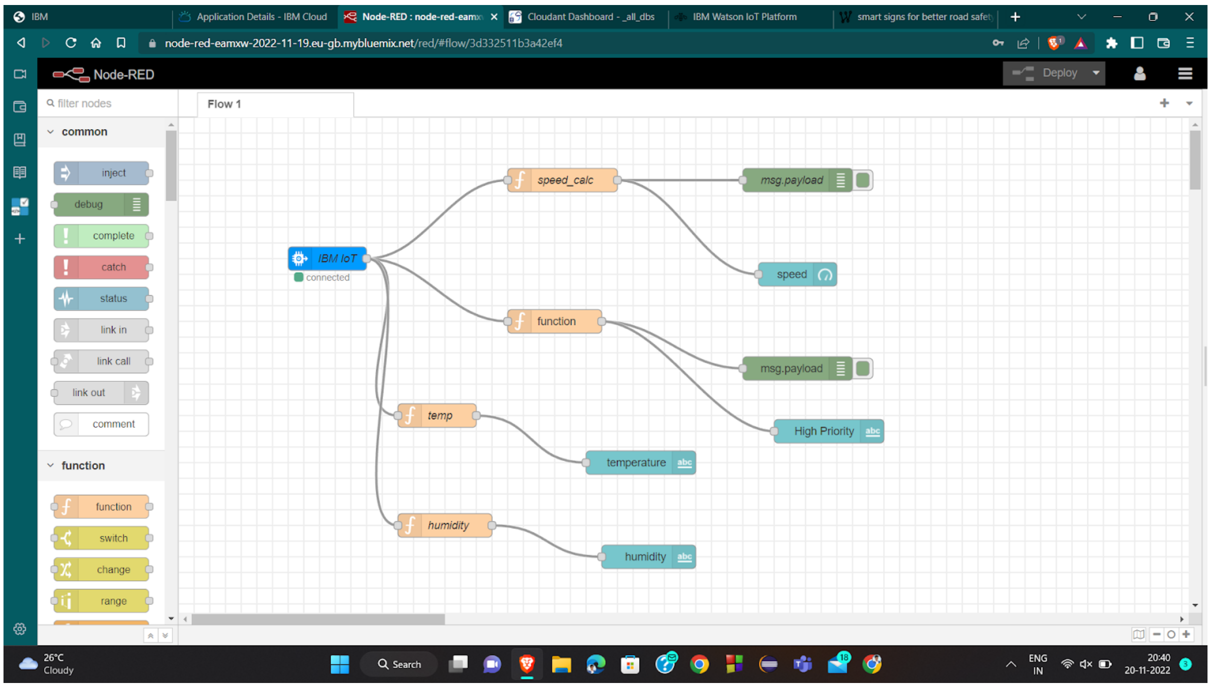
Wokwi Simulation:

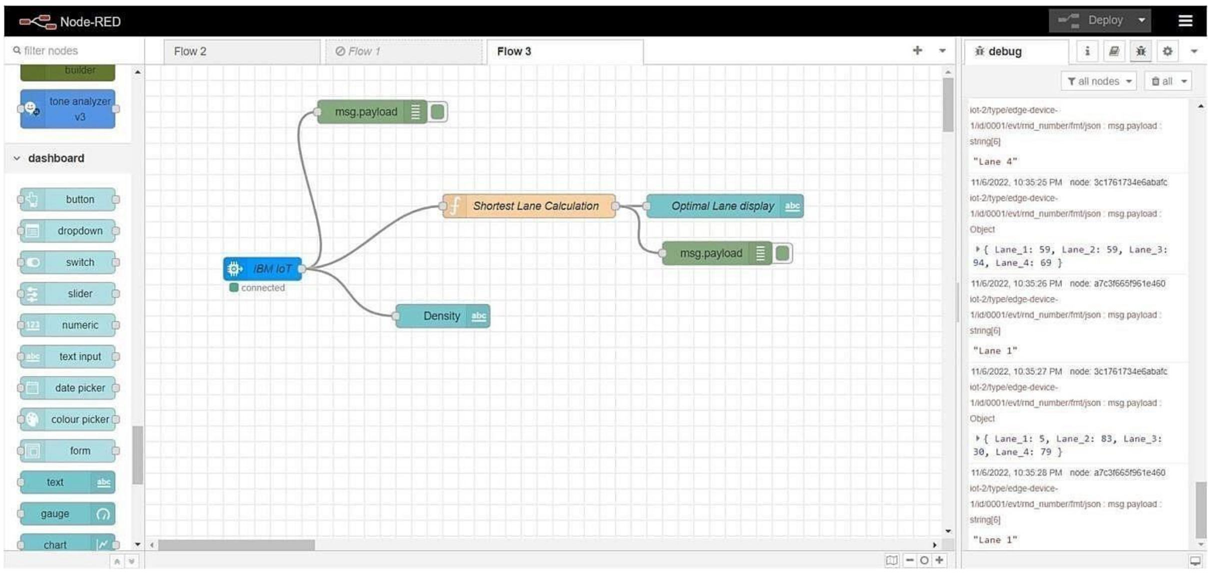


```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 5     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11

DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht connected

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

//-------credentials of IBM Accounts------

#define ORG "c3wgxl"//IBM ORGANISATION ID
#define DEVICE_TYPE "ajai"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "12345"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "123456789"     //Token
String data3;
float h, t;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format in which da
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT command type AND COMMAND IS TEST OF F
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id


//-----------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by passing paramet


void setup()// configureing the ESP32
```

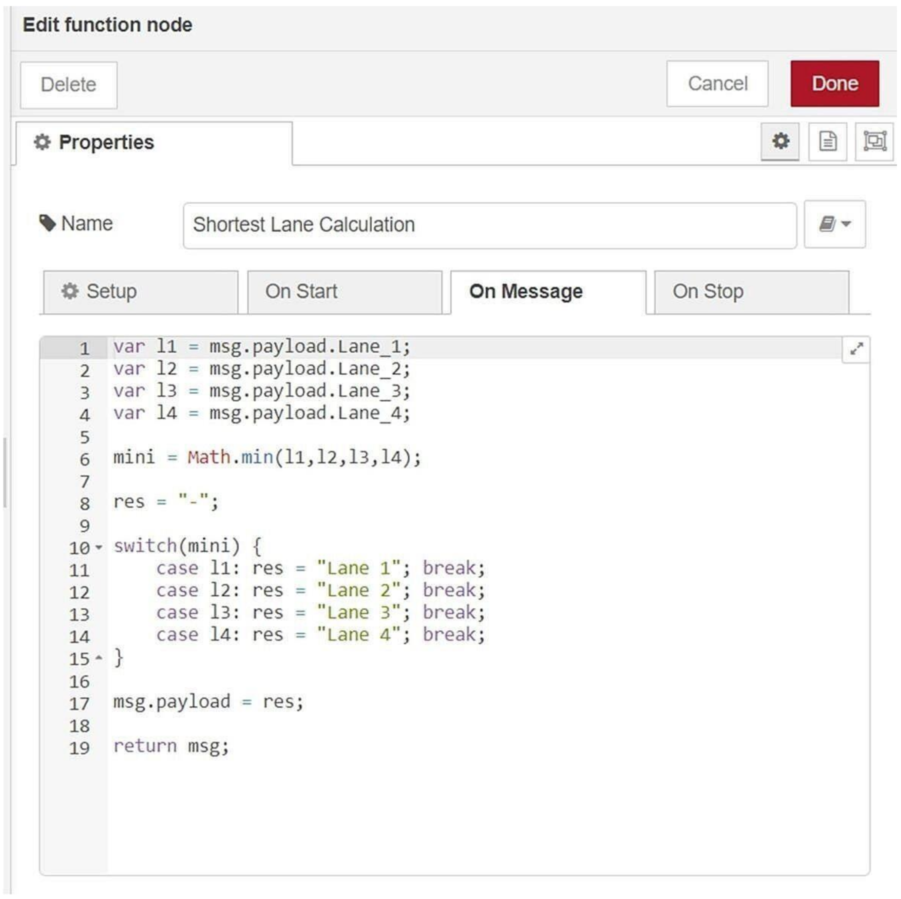
```
Connecting to ....
WiFi connected
IP address:
10.10.0.2
Reconnecting client to
c3wgxl.messaging.internetofthings.ibmcloud.com
```

Node Red:





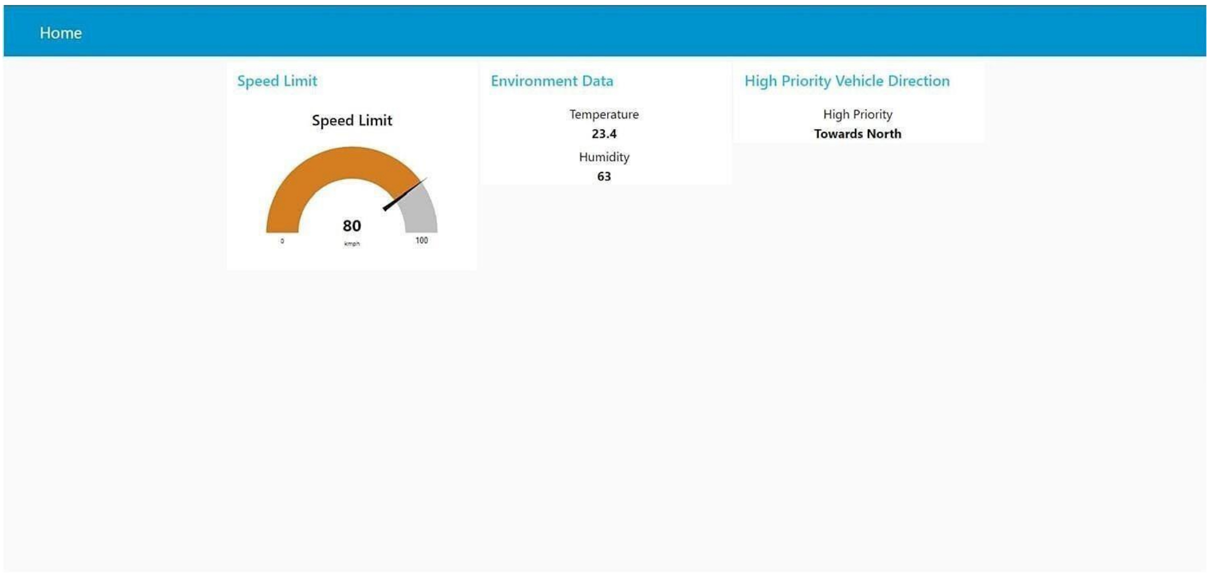
```
iot-2/type/edge-device-1/id/0001/evt/rnd_number/fmt/json : msg.payload : string[6]
"Lane 4"
11/6/2022, 10:35:25 PM  node: 3c1761734e6abafc
iot-2/type/edge-device-1/id/0001/evt/rnd_number/fmt/json : msg.payload : Object
▸{ Lane_1: 59, Lane_2: 59, Lane_3: 94, Lane_4: 69 }
11/6/2022, 10:35:26 PM  node: a7c3f665f961e460
iot-2/type/edge-device-1/id/0001/evt/rnd_number/fmt/json : msg.payload : string[6]
"Lane 1"
11/6/2022, 10:35:27 PM  node: 3c1761734e6abafc
iot-2/type/edge-device-1/id/0001/evt/rnd_number/fmt/json : msg.payload : Object
▸{ Lane_1: 5, Lane_2: 83, Lane_3: 30, Lane_4: 79 }
11/6/2022, 10:35:28 PM  node: a7c3f665f961e460
iot-2/type/edge-device-1/id/0001/evt/rnd_number/fmt/json : msg.payload : string[6]
"Lane 1"
```



Name: Shortest Lane Calculation

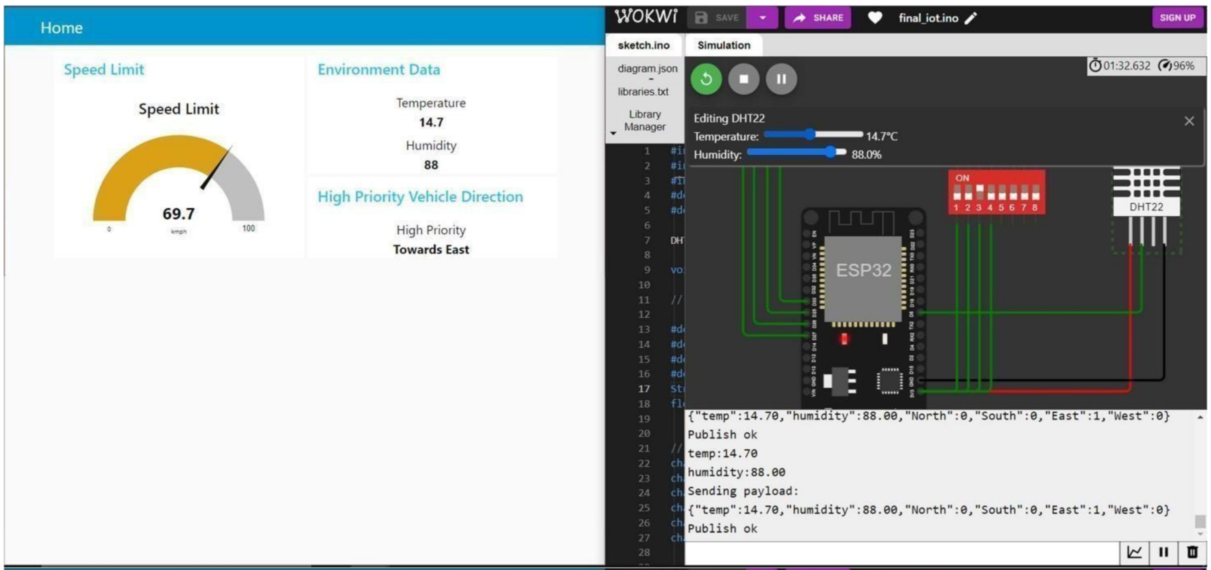
```
var l1 = msg.payload.Lane_1;
var l2 = msg.payload.Lane_2;
var l3 = msg.payload.Lane_3;
var l4 = msg.payload.Lane_4;

mini = Math.min(l1,l2,l3,l4);

res = "-";

switch(mini) {
    case l1: res = "Lane 1"; break;
    case l2: res = "Lane 2"; break;
    case l3: res = "Lane 3"; break;
    case l4: res = "Lane 4"; break;
}

msg.payload = res;

return msg;
```

Node red Web UI:



Temperature
23.4
Humidity
63

High Priority
Towards North

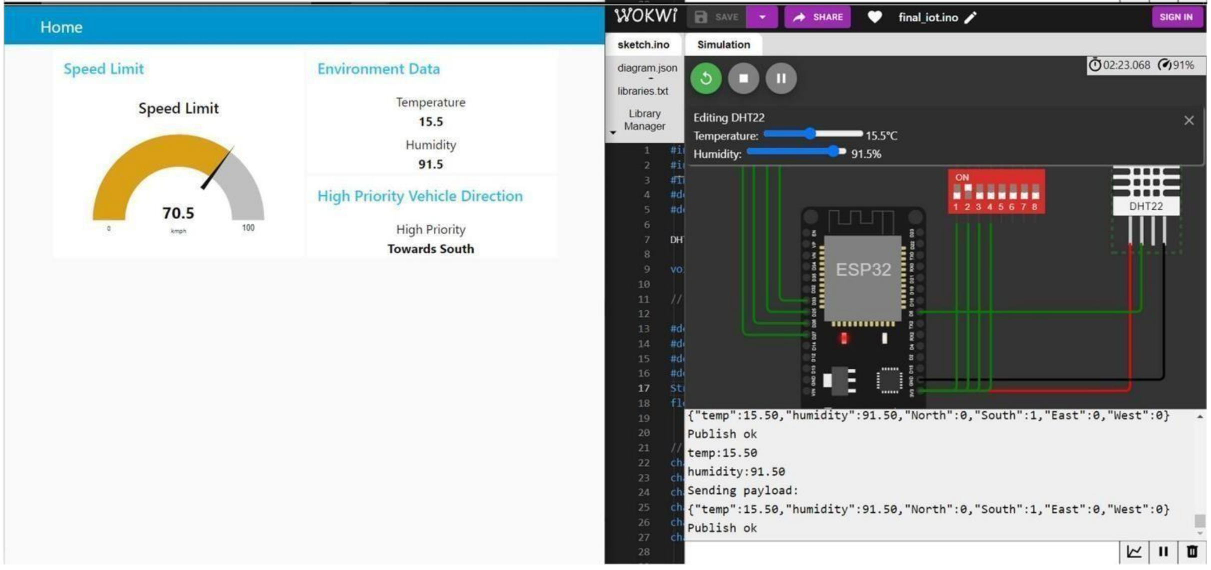


Temperature
14.7
Humidity
88

High Priority
Towards East

Editing DHT22
Temperature: 14.7°C
Humidity: 88.0%

```
{"temp":14.70,"humidity":88.00,"North":0,"South":0,"East":1,"West":0}
Publish ok
temp:14.70
humidity:88.00
Sending payload:
{"temp":14.70,"humidity":88.00,"North":0,"South":0,"East":1,"West":0}
Publish ok
```



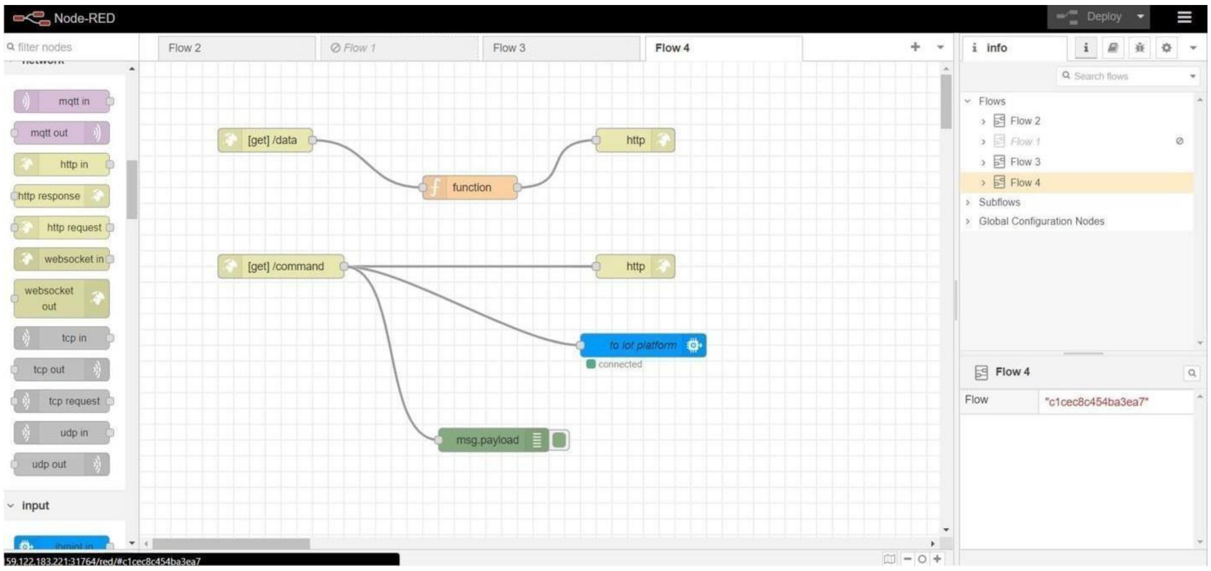
Temperature
15.5
Humidity
91.5

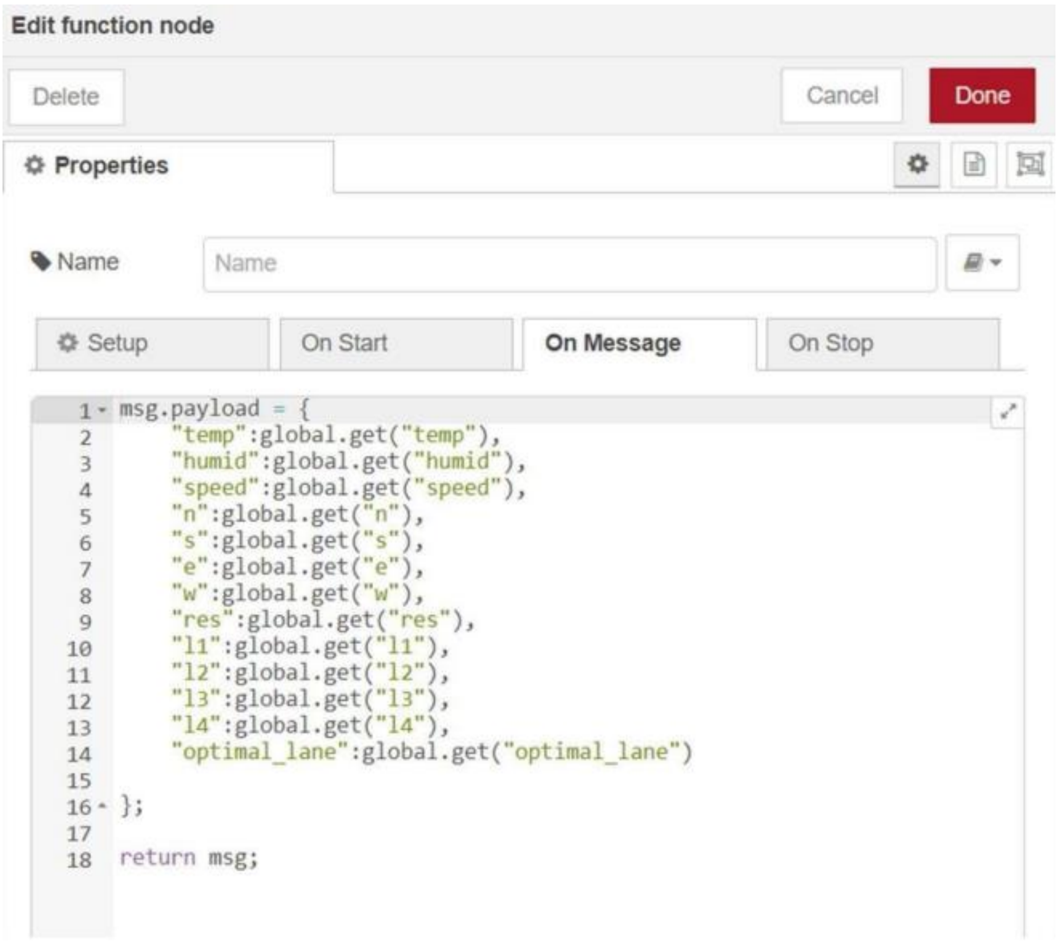
High Priority
Towards South

Editing DHT22
Temperature: 15.5°C
Humidity: 91.5%

```
{"temp":15.50,"humidity":91.50,"North":0,"South":1,"East":0,"West":0}
Publish ok
temp:15.50
humidity:91.50
Sending payload:
{"temp":15.50,"humidity":91.50,"North":0,"South":1,"East":0,"West":0}
Publish ok
```

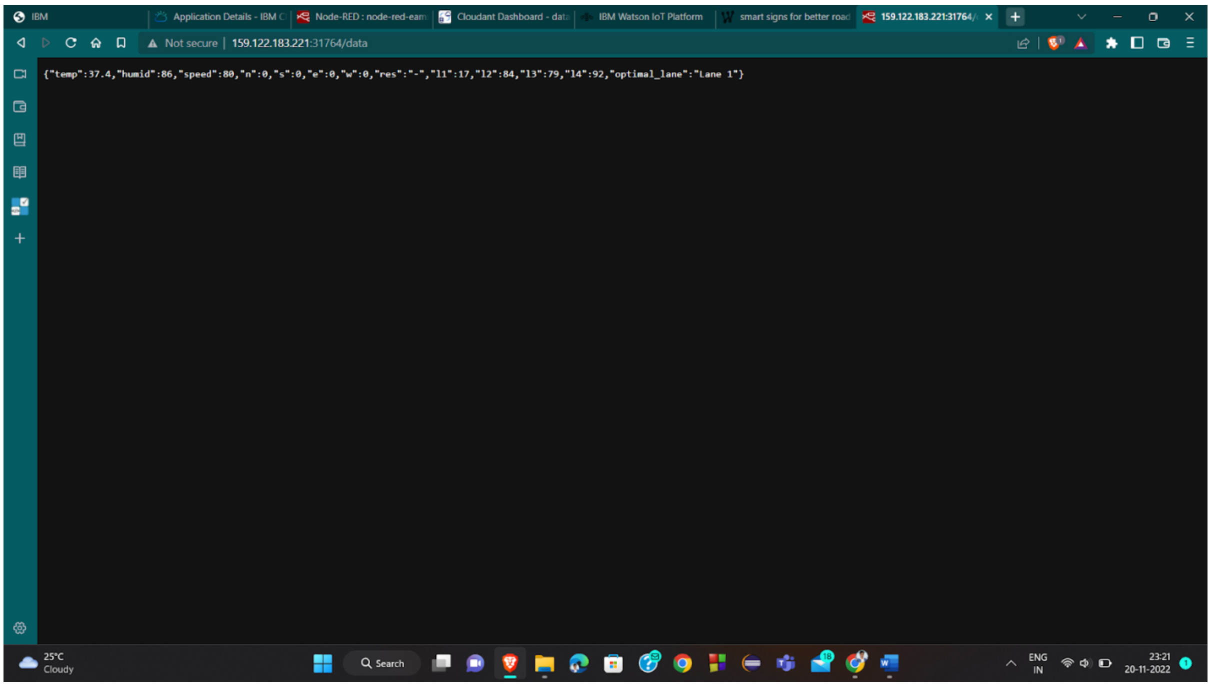
Node red – connect with MIT App Invertor:



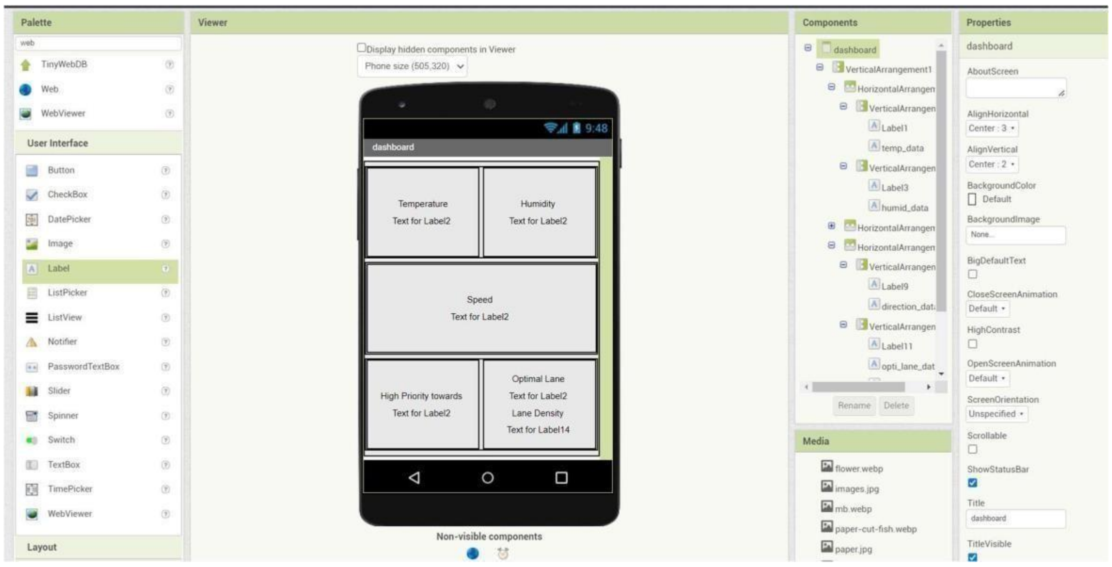


```
msg.payload = {
      "temp":global.get("temp"),
      "humid":global.get("humid"),
      "speed":global.get("speed"),
      "n":global.get("n"),
      "s":global.get("s"),
      "e":global.get("e"),
      "w":global.get("w"),
      "res":global.get("res"),
      "l1":global.get("l1"),
      "l2":global.get("l2"),
      "l3":global.get("l3"),
      "l4":global.get("l4"),
      "optimal_lane":global.get("optimal_lane")

};

return msg;
```

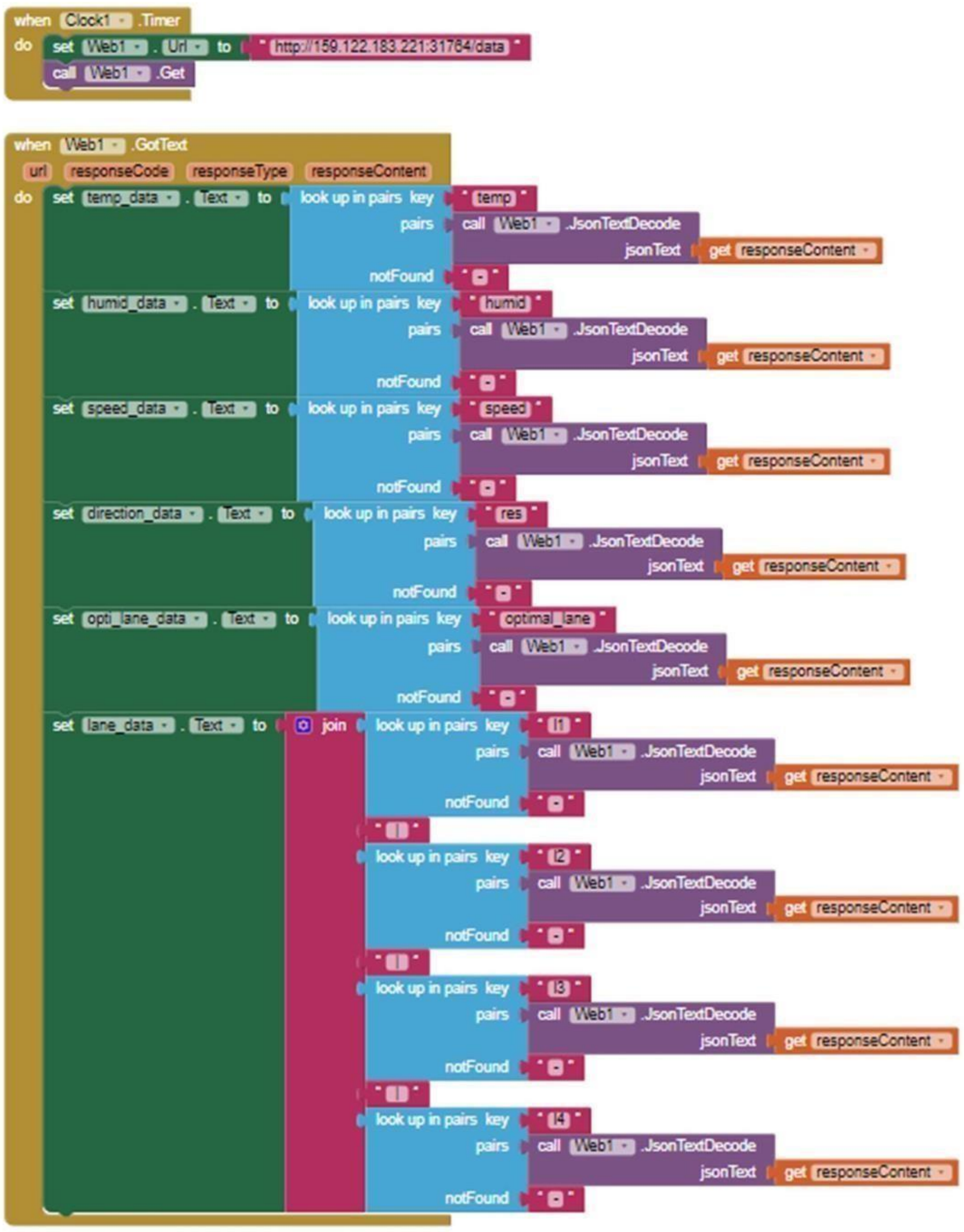
Output From Node red:


```
{"temp":37.4,"humid":86,"speed":80,"n":0,"s":0,"e":0,"w":0,"res":"-","l1":17,"l2":84,"l3":79,"l4":92,"optimal_lane":"Lane 1"}
```

MIT App Invertor UI Design:

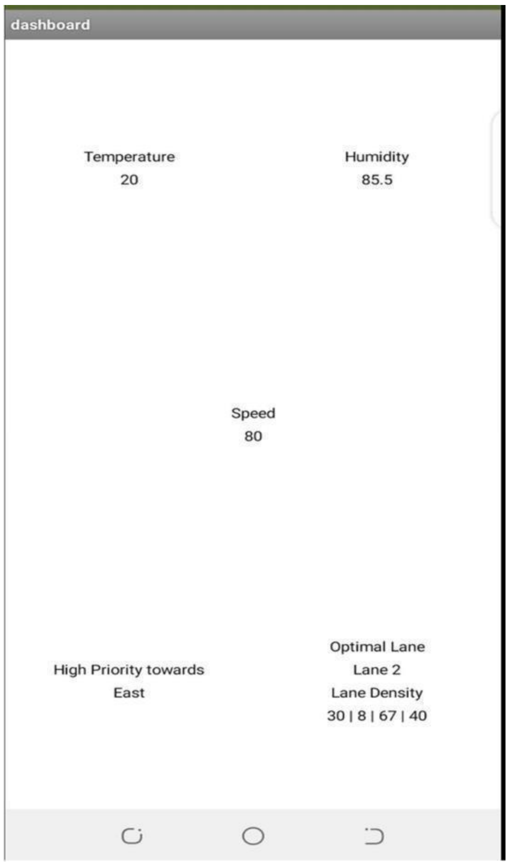


MIT App Inventor Backend design:



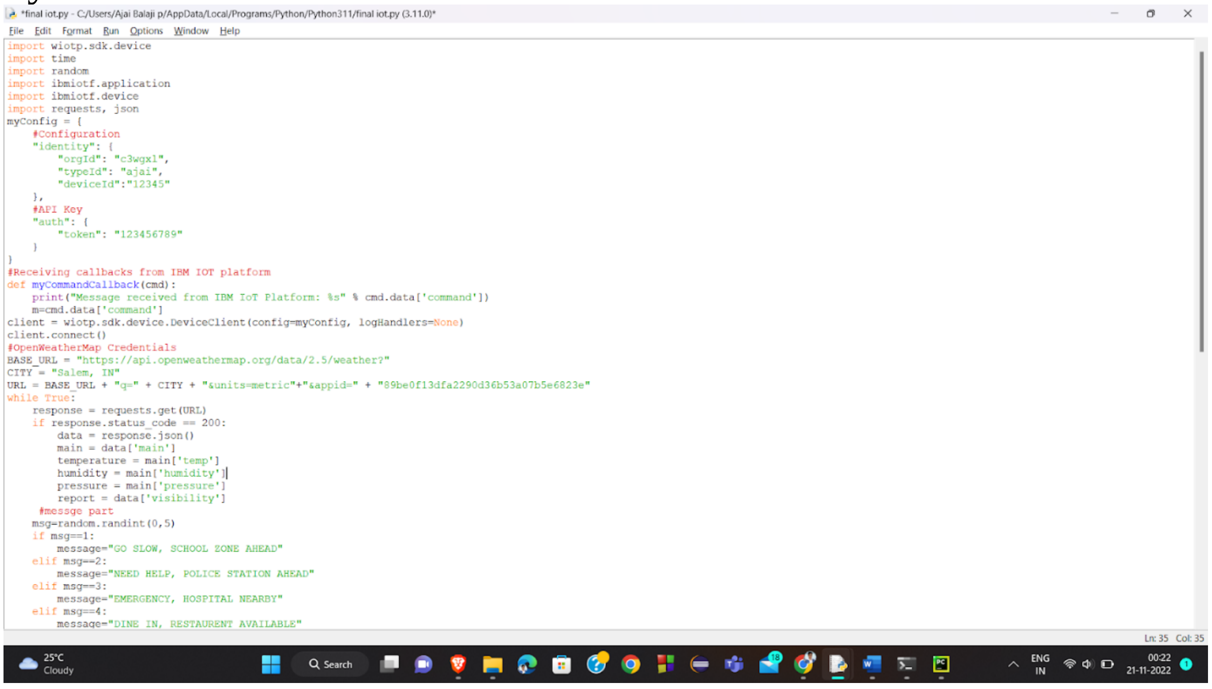
```
when Clock1 .Timer
do  set Web1 . Url to " http://159.122.183.221:31764/data "
    call Web1 .Get

when Web1 .GotText
 url  responseCode  responseType  responseContent
do  set temp_data . Text to  look up in pairs  key " temp "
                                               pairs  call Web1 .JsonTextDecode
                                                                      jsonText  get responseContent
                                            notFound " "
    set humid_data . Text to  look up in pairs  key " humid "
                                                pairs  call Web1 .JsonTextDecode
                                                                       jsonText  get responseContent
                                             notFound " "
    set speed_data . Text to  look up in pairs  key " speed "
                                                pairs  call Web1 .JsonTextDecode
                                                                       jsonText  get responseContent
                                             notFound " "
    set direction_data . Text to  look up in pairs  key " res "
                                                    pairs  call Web1 .JsonTextDecode
                                                                           jsonText  get responseContent
                                                 notFound " "
    set opti_lane_data . Text to  look up in pairs  key " optimal_lane "
                                                    pairs  call Web1 .JsonTextDecode
                                                                           jsonText  get responseContent
                                                 notFound " "
    set lane_data . Text to  join  look up in pairs  key " l1 "
                                                     pairs  call Web1 .JsonTextDecode
                                                                            jsonText  get responseContent
                                                  notFound " "
                                   " | "
                                   look up in pairs  key " l2 "
                                                     pairs  call Web1 .JsonTextDecode
                                                                            jsonText  get responseContent
                                                  notFound " "
                                   " | "
                                   look up in pairs  key " l3 "
                                                     pairs  call Web1 .JsonTextDecode
                                                                            jsonText  get responseContent
                                                  notFound " "
                                   " | "
                                   look up in pairs  key " l4 "
                                                     pairs  call Web1 .JsonTextDecode
                                                                            jsonText  get responseContent
                                                  notFound " "
```

Output from MIT App:



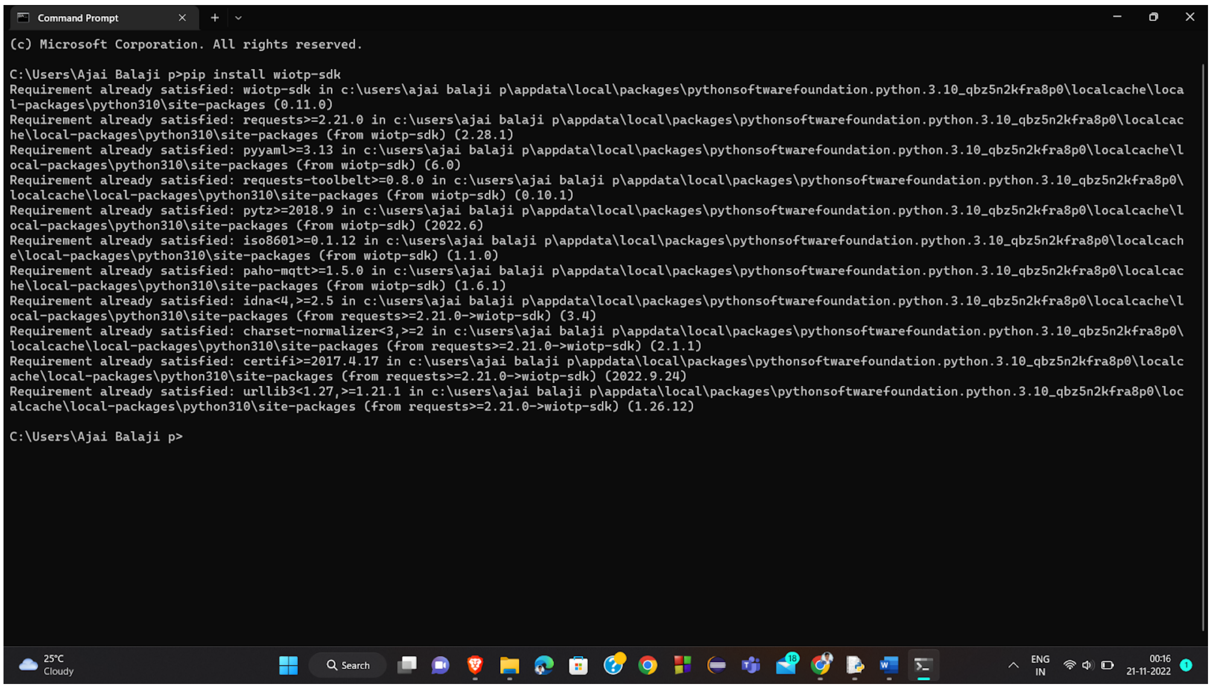
9.2 User Acceptance Testing:

Python Simulation:

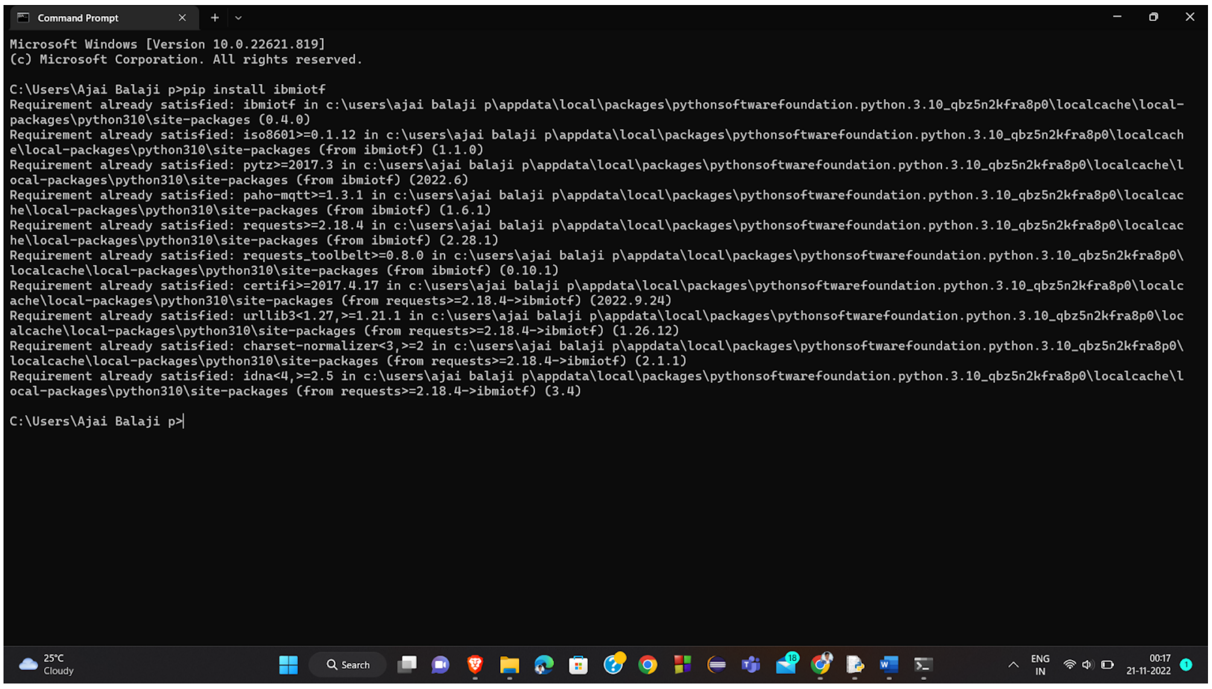

final iot.py - C:/Users/Ajai Balaji p/AppData/Local/Programs/Python/Python311/final iot.py (3.11.0)
File Edit Format Run Options Window Help

```
import wiotp.sdk.device
import time
import random
import ibmiotf.application
import ibmiotf.device
import requests, json
myConfig = {
    #Configuration
    "identity": {
        "orgId": "c3wgxl",
        "typeId": "ajai",
        "deviceId":"12345"
    },
    #API Key
    "auth": {
        "token": "123456789"
    }
}
#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=" + "89be0f13dfa2290d36b53a07b5e6823e"
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"
```

Import wiotp-sdk and ibmiotf:

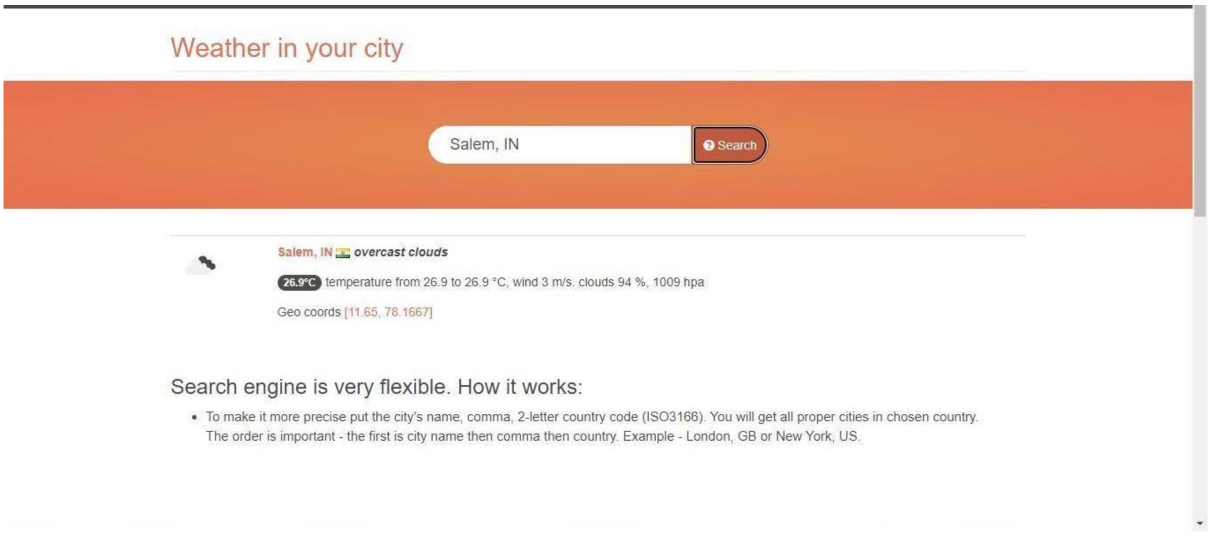


```
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ajai Balaji p>pip install wiotp-sdk
Requirement already satisfied: wiotp-sdk in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\loca
l-packages\python310\site-packages (0.11.0)
Requirement already satisfied: requests>=2.21.0 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcac
he\local-packages\python310\site-packages (from wiotp-sdk) (2.28.1)
Requirement already satisfied: pyyaml>=3.13 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\l
ocal-packages\python310\site-packages (from wiotp-sdk) (6.0)
Requirement already satisfied: requests-toolbelt>=0.8.0 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\
localcache\local-packages\python310\site-packages (from wiotp-sdk) (0.10.1)
Requirement already satisfied: pytz>=2018.9 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\l
ocal-packages\python310\site-packages (from wiotp-sdk) (2022.6)
Requirement already satisfied: iso8601>=0.1.12 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcach
e\local-packages\python310\site-packages (from wiotp-sdk) (1.1.0)
Requirement already satisfied: paho-mqtt>=1.5.0 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcac
he\local-packages\python310\site-packages (from wiotp-sdk) (1.6.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\l
ocal-packages\python310\site-packages (from requests>=2.21.0->wiotp-sdk) (3.4)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\
localcache\local-packages\python310\site-packages (from requests>=2.21.0->wiotp-sdk) (2.1.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localc
ache\local-packages\python310\site-packages (from requests>=2.21.0->wiotp-sdk) (2022.9.24)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\loc
alcache\local-packages\python310\site-packages (from requests>=2.21.0->wiotp-sdk) (1.26.12)

C:\Users\Ajai Balaji p>
```



```
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ajai Balaji p>pip install ibmiotf
Requirement already satisfied: ibmiotf in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\local-
packages\python310\site-packages (0.4.0)
Requirement already satisfied: iso8601>=0.1.12 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcach
e\local-packages\python310\site-packages (from ibmiotf) (1.1.0)
Requirement already satisfied: pytz>=2017.3 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\l
ocal-packages\python310\site-packages (from ibmiotf) (2022.6)
Requirement already satisfied: paho-mqtt>=1.3.1 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcac
he\local-packages\python310\site-packages (from ibmiotf) (1.6.1)
Requirement already satisfied: requests>=2.18.4 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcac
he\local-packages\python310\site-packages (from ibmiotf) (2.28.1)
Requirement already satisfied: requests_toolbelt>=0.8.0 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\
localcache\local-packages\python310\site-packages (from ibmiotf) (0.10.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localc
ache\local-packages\python310\site-packages (from requests>=2.18.4->ibmiotf) (2022.9.24)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\loc
alcache\local-packages\python310\site-packages (from requests>=2.18.4->ibmiotf) (1.26.12)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\
localcache\local-packages\python310\site-packages (from requests>=2.18.4->ibmiotf) (2.1.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ajai balaji p\appdata\local\packages\pythonsoftwarefoundation.python.3.10_qbz5n2kfra8p0\localcache\l
ocal-packages\python310\site-packages (from requests>=2.18.4->ibmiotf) (3.4)

C:\Users\Ajai Balaji p>
```

OpenWeatherMap:



Weather in your city

Salem, IN

Salem, IN overcast clouds

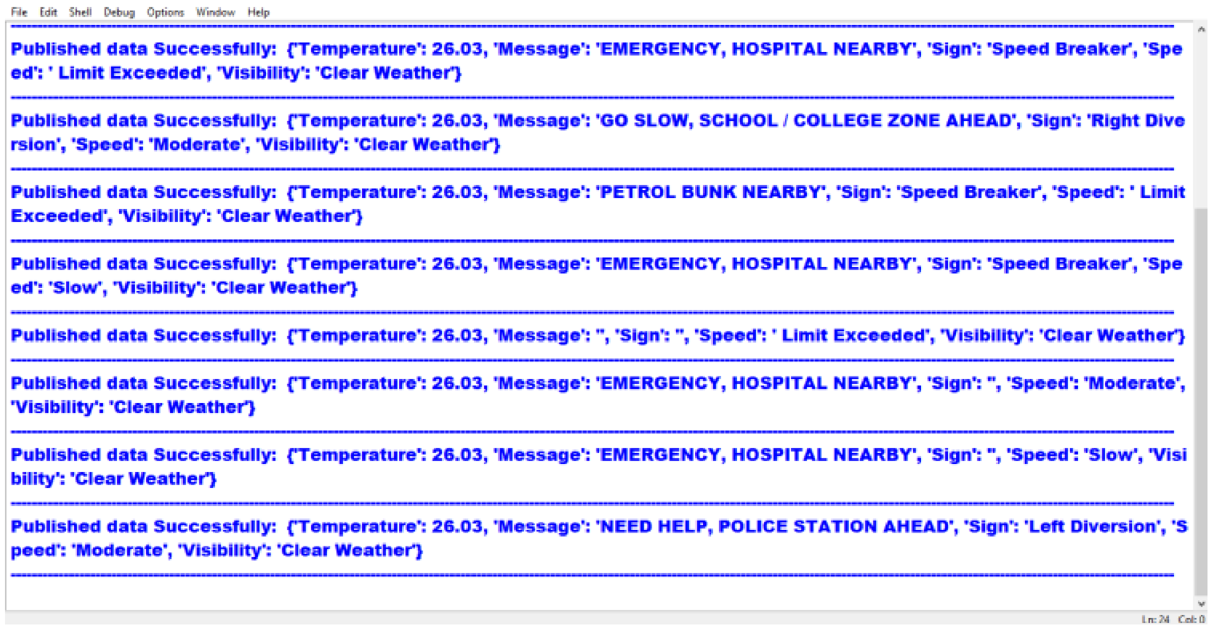
26.9°C temperature from 26.9 to 26.9 °C, wind 3 m/s. clouds 94 %, 1009 hpa

Geo coords [11.65, 78.1667]

Search engine is very flexible. How it works:

- To make it more precise put the city's name, comma, 2-letter country code (ISO3166). You will get all proper cities in chosen country. The order is important - the first is city name then comma then country. Example - London, GB or New York, US.

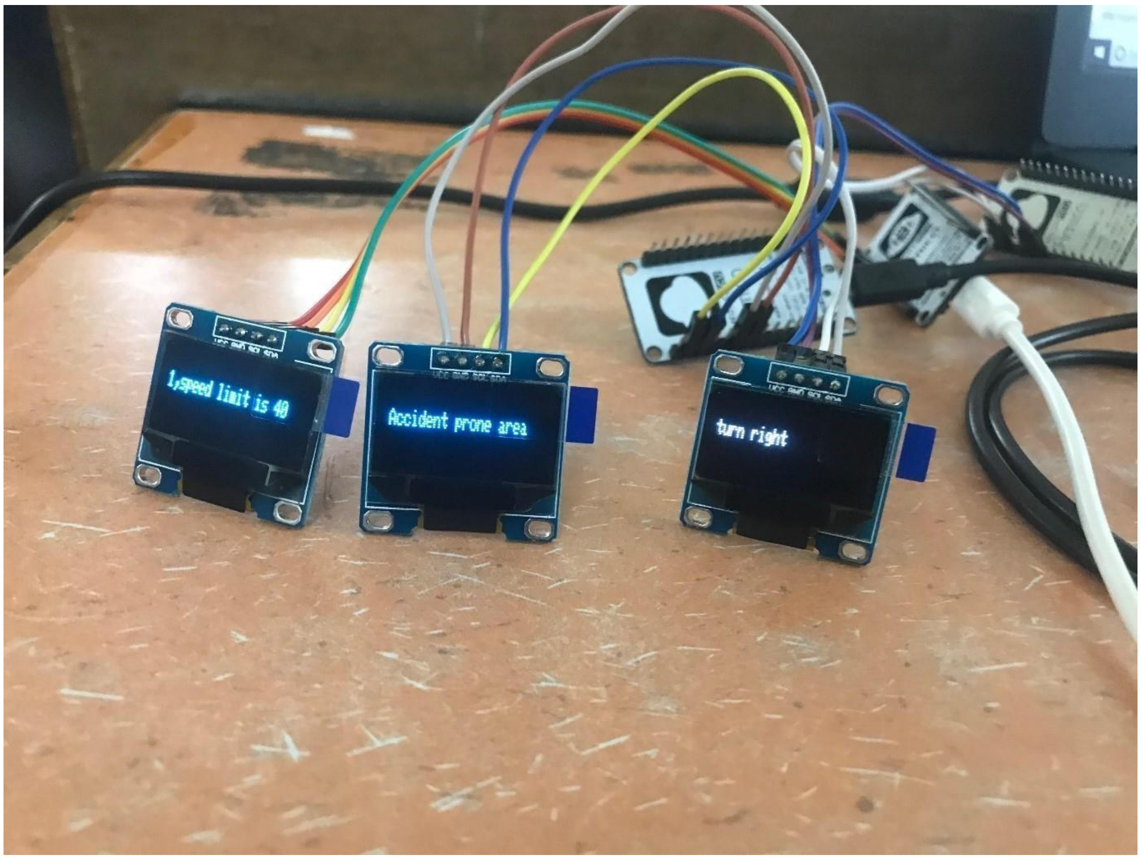
Python IDLE output:



```
------------------------------------------------------------------------------------------------
Published data Successfully:  {'Temperature': 26.03, 'Message': 'EMERGENCY, HOSPITAL NEARBY', 'Sign': 'Speed Breaker', 'Spe
ed': ' Limit Exceeded', 'Visibility': 'Clear Weather'}
------------------------------------------------------------------------------------------------
Published data Successfully:  {'Temperature': 26.03, 'Message': 'GO SLOW, SCHOOL / COLLEGE ZONE AHEAD', 'Sign': 'Right Dive
rsion', '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', 'Spe
ed': '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', 'Visi
bility': 'Clear Weather'}
------------------------------------------------------------------------------------------------
Published data Successfully:  {'Temperature': 26.03, 'Message': 'NEED HELP, POLICE STATION AHEAD', 'Sign': 'Left Diversion', 'S
peed': 'Moderate', 'Visibility': 'Clear Weather'}
------------------------------------------------------------------------------------------------
```

10. Results:

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



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 and 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, long- lasting, 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 utilization, weather conditions, and accident records.

13. Future Scope:

- 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.

- 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.

- Glow in 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.

- 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.

- 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.

- 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.

- 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:

```
import wiotp.sdk.device
import time
import random
import ibmiotf.application
import ibmiotf.device
import requests, json
myConfig = {
    #Configuration
    "identity": {
        "orgId": "c3wgxl",
        "typeId": "ajai",
        "deviceId":"12345"
    },
    #API Key
    "auth": {
        "token": "123456789"
    }
}
#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)
print("________________________________________")
client.commandCallback = myCommandCallback
time.sleep(5)
client.disconnect()
```

Github Link:

https://github.com/IBM-EPBL/IBM-Project-33637-1660224780