

Internet Of Things

In partial fulfillment for the completion of

Signs with Smart Connectivity For Better Road Safety

PROJECT REPORT

Submitted by

PAVITHRA S (711319EC079)

NARESH KUMAR A (711319EC074)

ROHINI C S (711319EC090)

MALLSRI (711319EC058)

DEVADHARSHINI M

| | |
|---------|------------------|
| Date | 18 November 2022 |
| Team ID | PNT2022TMIDI4352 |

BACHELOR OF ENGINEERING

IN

OF

ELECTRONICS AND COMMUNICATION ENGINEERING

KPR INSTITUTE OF ENGINEERING AND TECHNOLOGY,

COIMBATORE.

**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:

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 References :

1. <https://www.hindawi.com/journals/jat/2022/5829607/>
2. https://en.wikipedia.org/wiki/Automotive_safety
3. <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 & PROPOSED SOLUTION :

3.1 Empathy Map Canvas :

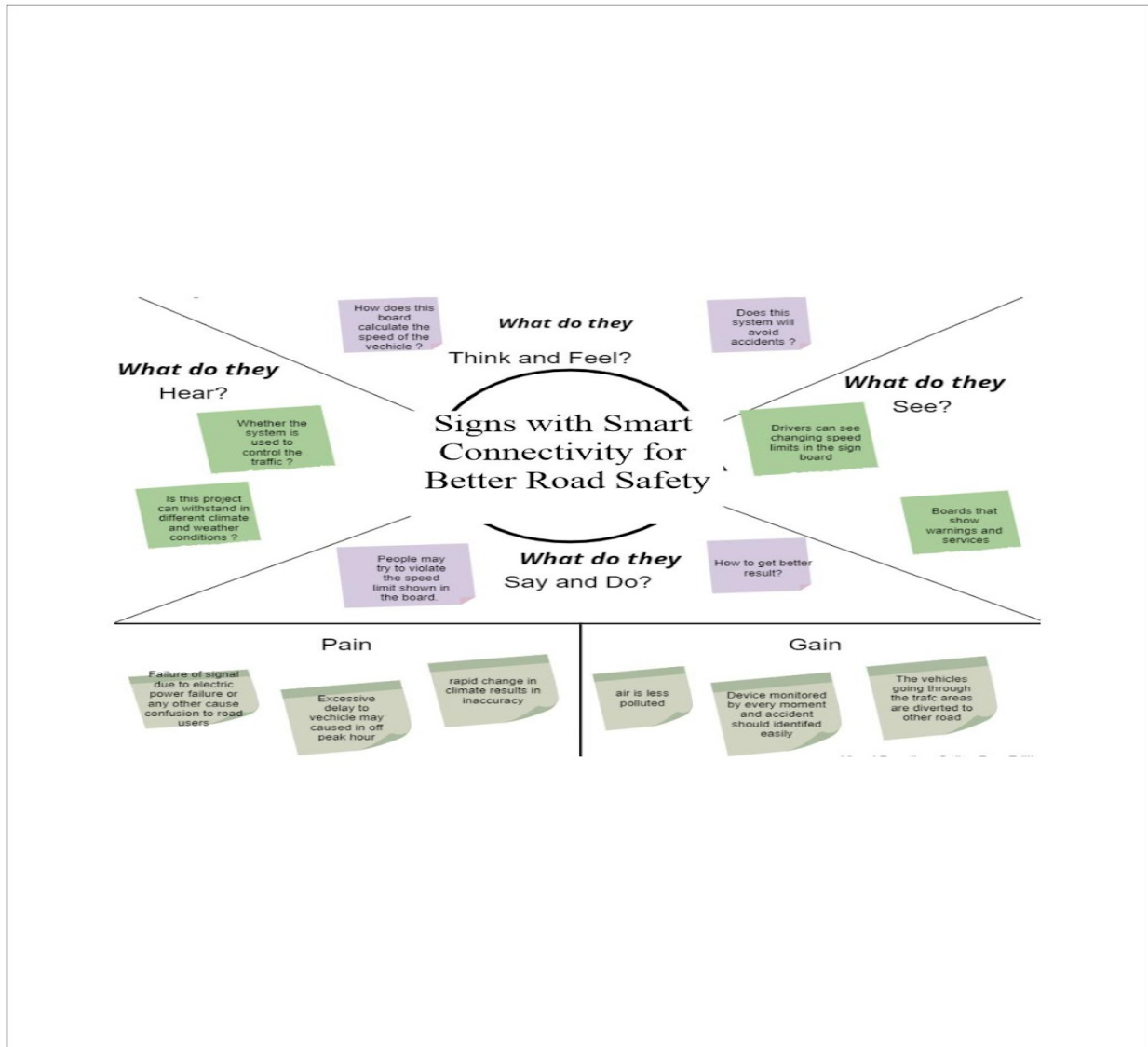


Fig 3.1 Empathy Map

3.2 Ideation & Brainstorming :

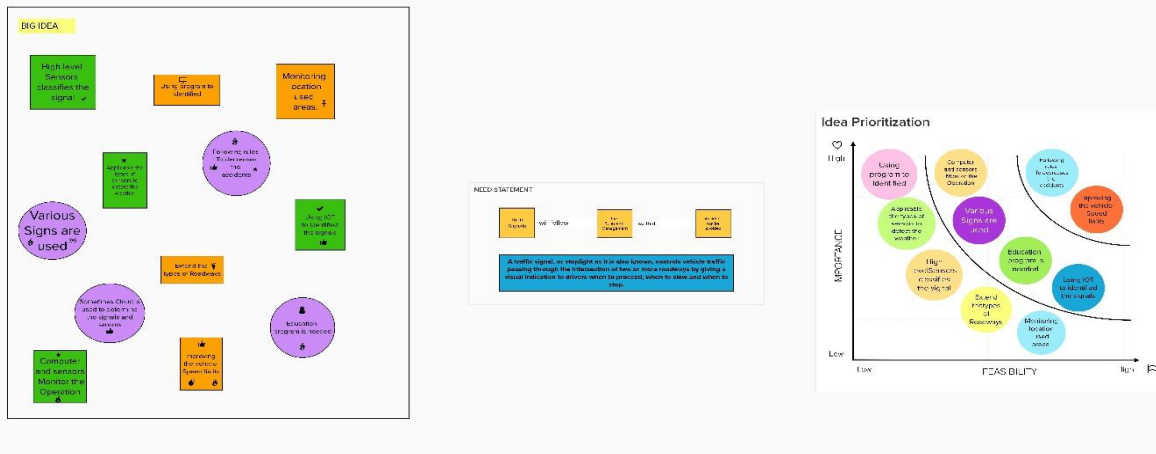


Fig 3.2 Ideation & Brainstorming

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 :

Problem-Solution fit canvas 2.0 Signs with Smart Connectivity for Better Road Safety TEAM ID – PNT2022TMD14352

| | | | | |
|---------------------------|---|---|--|---------------------------|
| Define CS, fit into CC | 1. CUSTOMER SEGMENT(S) CS Who is your customer? <input type="checkbox"/> highway division <input type="checkbox"/> Passenger | 6. CUSTOMER CONSTRAINTS CC What constraints prevent your customers from taking action or limit their choices of solutions? The impact of the network on the tests was a significant and unexpected element. Given the quantity of sensors, this IoT-based system was successful in simulating a large-scale smart agricultural 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? Along roadways, static signs with clear directions are up as potential fixes which gives clear solution. | Explore AS, differentiate |
| | 2. JOBS-TO-BE-DONE / PROBLEMS J&P Which jobs-to-be-done (or problems) do you address for your customers? There may be having of 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? 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 any people could lead to problems. | 7. BEHAVIOUR BE What does your customer do to address the problem and get the job done? As a teacher, the IOT cloud updates the smartboard 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, fit into CS | 3. TRIGGERS TR What triggers customers to act? i.e. seeing their neighbour installing 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. | 10. YOUR SOLUTION SL 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 3.1 ONLINE What kind of actions do customers take online? The departments can receive direct emails or messages from customers. (Officers on nearby patrol). 3.2 What kind of actions do customers take offline? Following directions is one of the major tasks for the traveler, but they can utilize the smartboard signs to check the state of the road from wherever they are standing. | Focus on AS, fit into CC |
| | 4. EMOTIONS: BEFORE / AFTER EM How do customers feel when they face a problem or a job and afterwards? Clients will feel better after selecting an operation mode with the use of smartboard connectivity, and they will then follow the instructions on the smartboard. | EM | | |

Identify strong TR & EM Extract online & offline CH of BE

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 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 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.

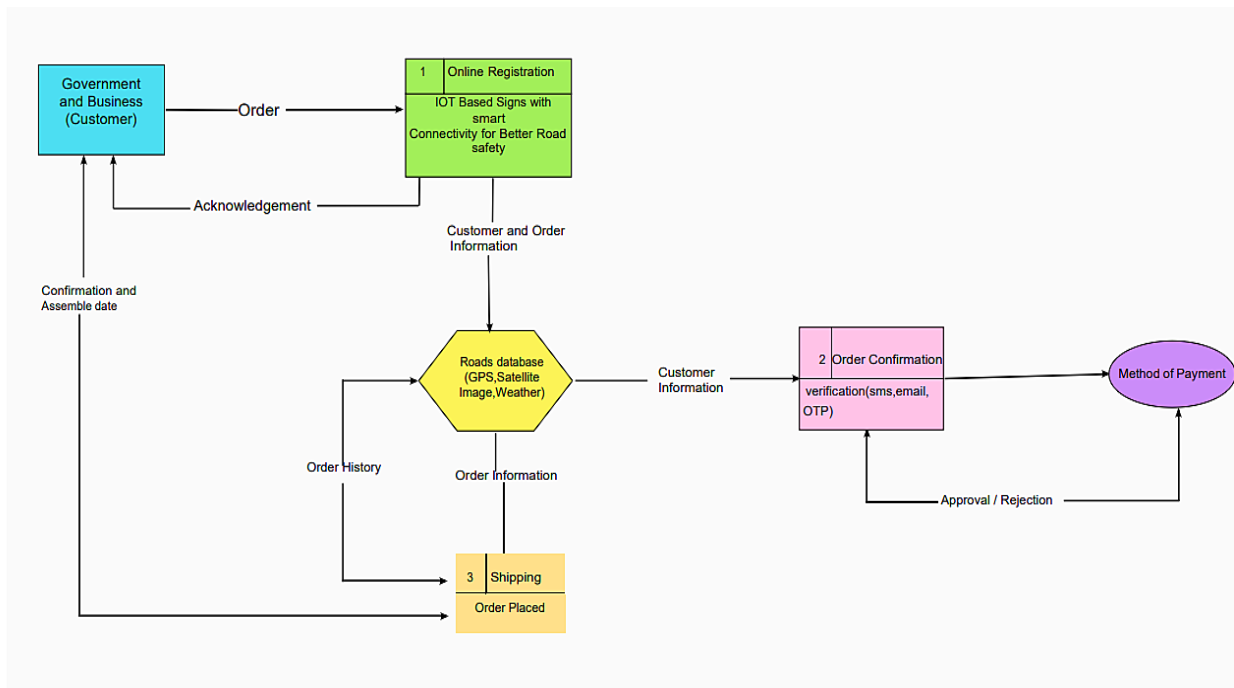


Fig 5.1 Data Flow Diagram - Signs with smart connectivity for better road safety

5.2 Solution & Technical Architecture :

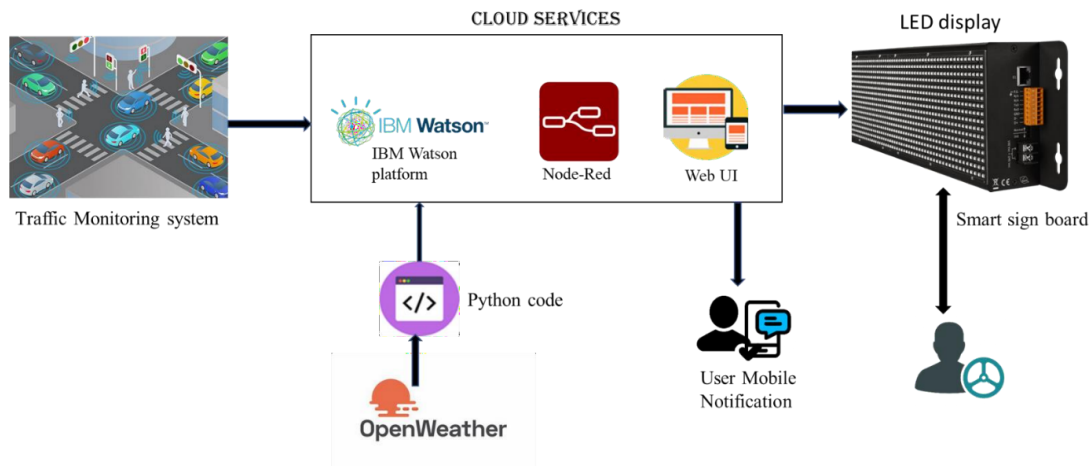


Fig 5.2 Solution & Technical Architecture - Signs with smart connectivity for better road safety

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 these application 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 | Pavithra Rohini Naresh Kumar Mallisri |
| Sprint-1 | Local Server/Software Run | Write a Python program that outputs results given the inputs like weather and location. | 1 | MEDIUM | Pavithra Rohini Naresh Kumar Mallisri |
| 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 | Pavithra Rohini Naresh Kumar Mallisri |
| Sprint-3 | Hardware initialization | Integrate the hardware to be able to access the cloud functions and provide inputs to the same. | 2 | HIGH | Pavithra Rohini Naresh Kumar Mallisri |
| Sprint-4 | UI/UX Optimization & Debugging | Optimize all the shortcomings and provide better user experience. | 2 | LOW | Pavithra Rohini Naresh Kumar Mallisri |

6.2 Sprint Delivery Schedule:

| Sprint | Total Sto ry Poi nt | Duration | Sprint Start Date | Sprint End Date (Planne d) | Story Points Completed (as on Planned End Date) | Sprint Release Date (Actual) |
|---------------|--|-----------------|----------------------------------|---|--|---|
| Sprint-1 | 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 |
| Sprint-3 | 20 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 20 | 07 Nov 2022 |
| Sprint-4 | 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 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 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=="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 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]) : "";
}

```

8.3 Feature 3 :

```
#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/348178332935782994>

The screenshot displays the Wokwi web-based IDE. On the left, the 'sketch.ino' file contains the following code:

```

1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #include "DHT.h" // Library for dht11
4 #define DHTPIN 5 // what pin we're connected to
5 #define DHTTYPE DHT22 // define type of sensor DHT 11
6
7 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of dht connect
8
9 void callback(char* topic, byte* payload, unsigned int payloadLength);
10
11 //-----credentials of IBM Accounts-----
12
13 #define ORG "psh4py" //IBM ORGANITION ID
14 #define DEVICE_TYPE "alert-device" //Device type mentioned in ibm watson IOT Platform
15 #define DEVICE_ID "4571" //Device ID mentioned in ibm watson IOT Platform
16 #define TOKEN "12345678" //Token
17 String data3;
18 float h, t;
19
20 //----- Customise the above values -----
21
22 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
23 char publishTopic[] = "iot-2/evt/data/fmt/json"; // topic name and type of event perform a
24 char subscribtopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND
25 char authMethod[] = "use-token-auth"; // authentication method
26 char token[] = TOKEN;
27 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
28
29 //-----
30 WiFiClient wifiClient; // creating the instance for wifiClient
31 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client
32
33
34

```

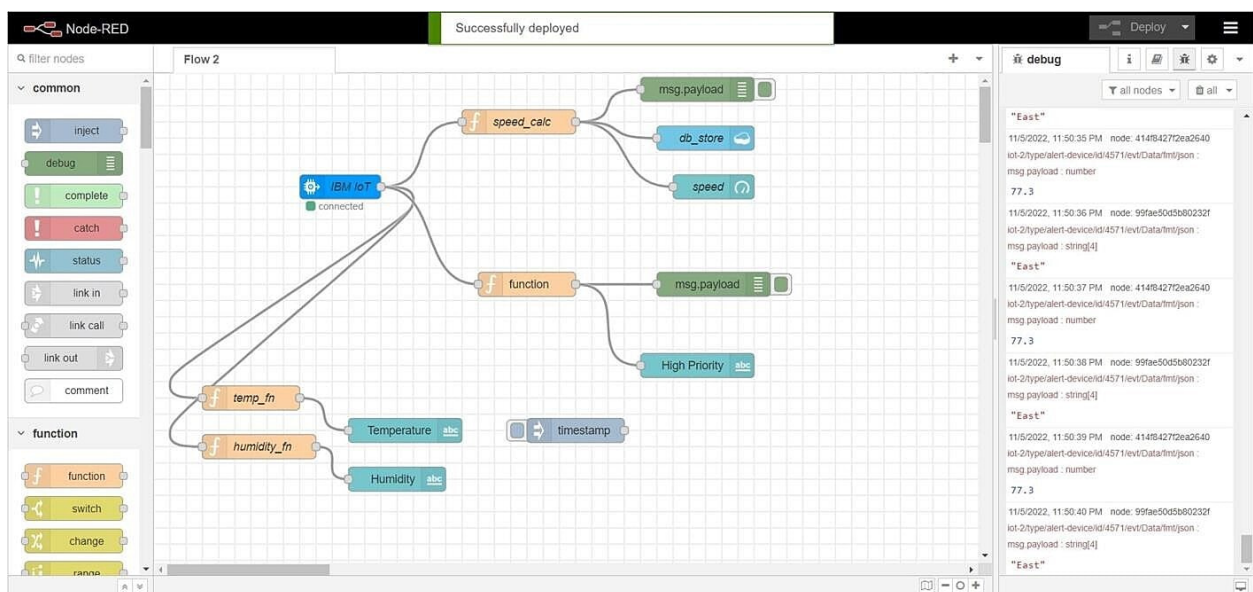
On the right, the 'Simulation' window shows a visual representation of the Arduino board and its connections. Below the board, the console output displays the following data:

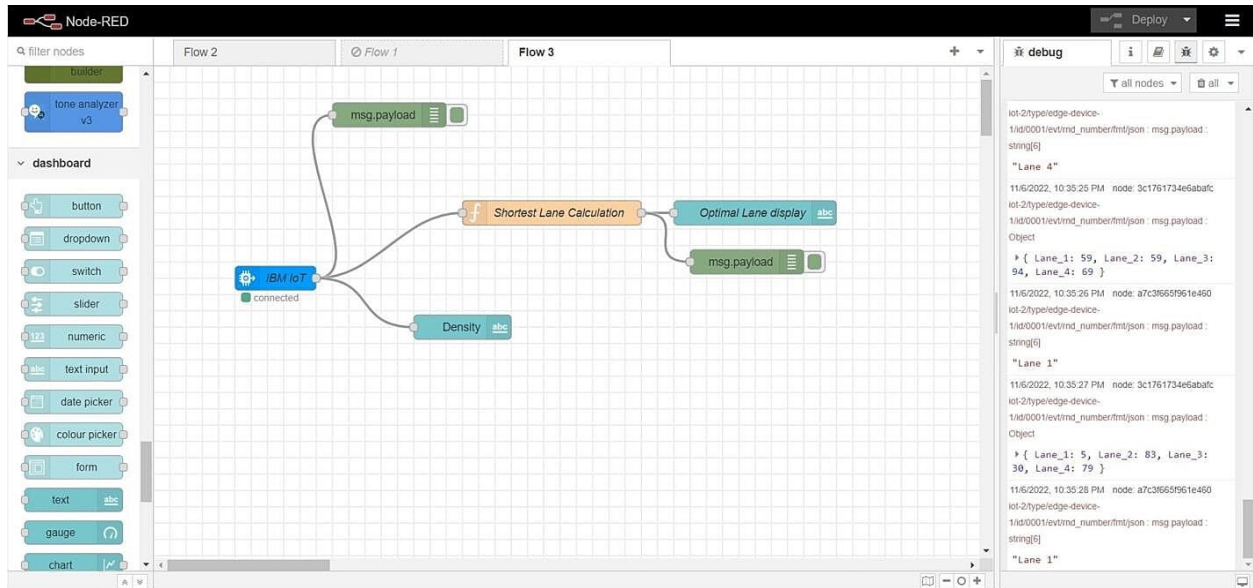
```

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

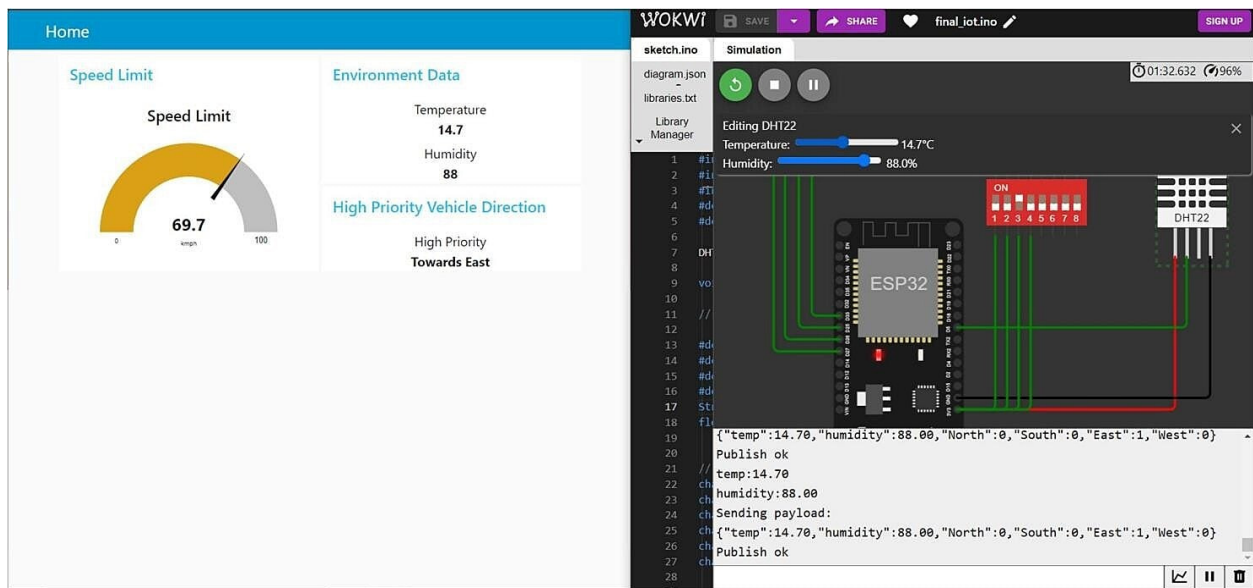
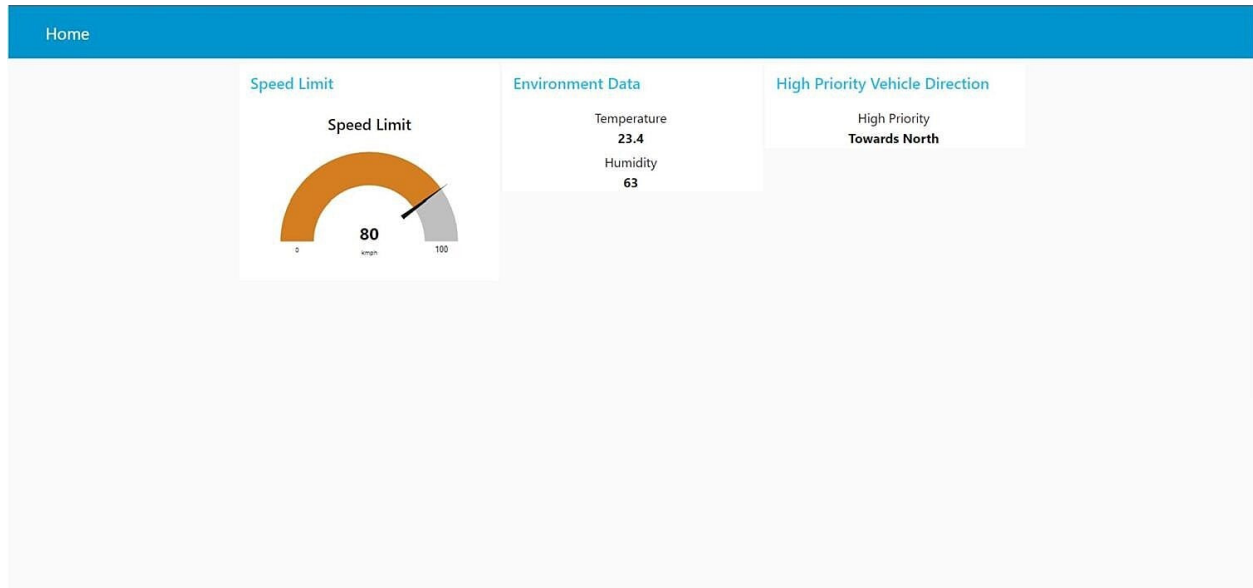
Name: Shortest Lane Calculation

Setup On Start **On Message** On Stop

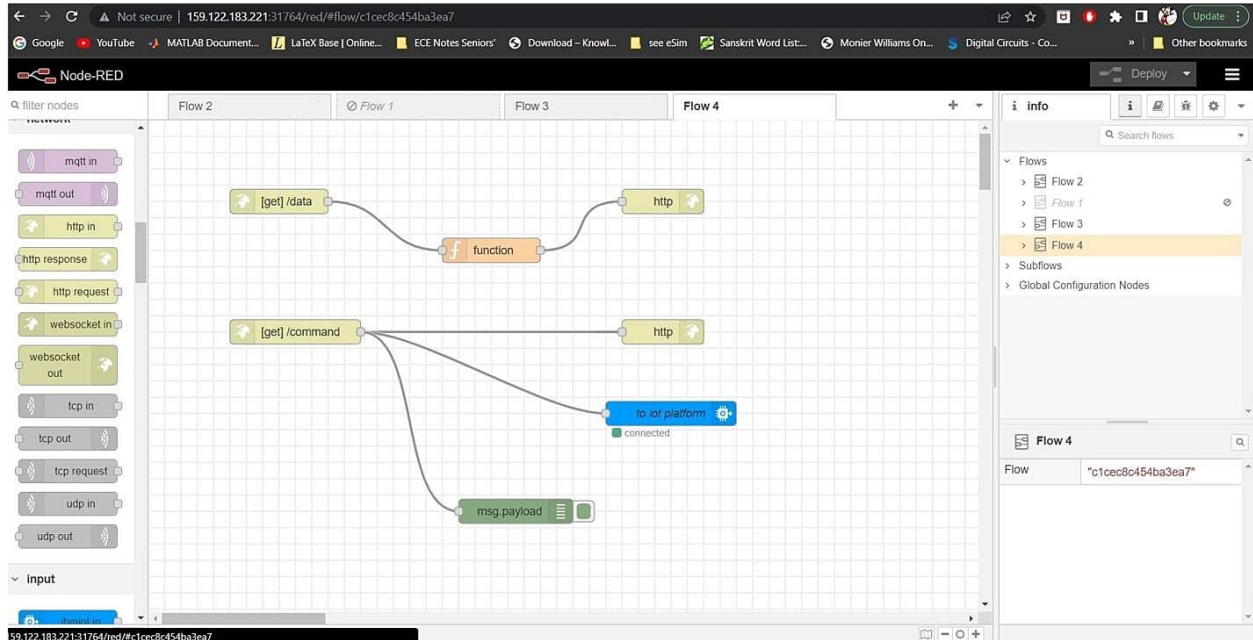
```

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 :



Node Red - Connect with MIT APP Inventor :



Edit function node

Buttons: Delete, Cancel, Done

Properties

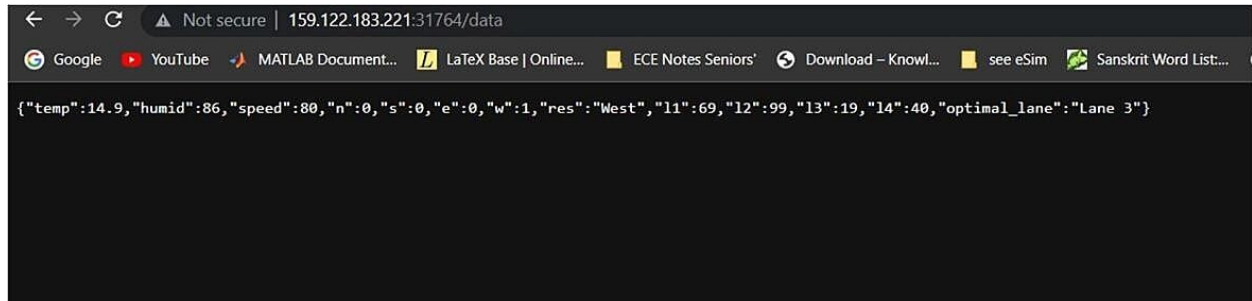
Name:

Setup | On Start | **On Message** | On Stop

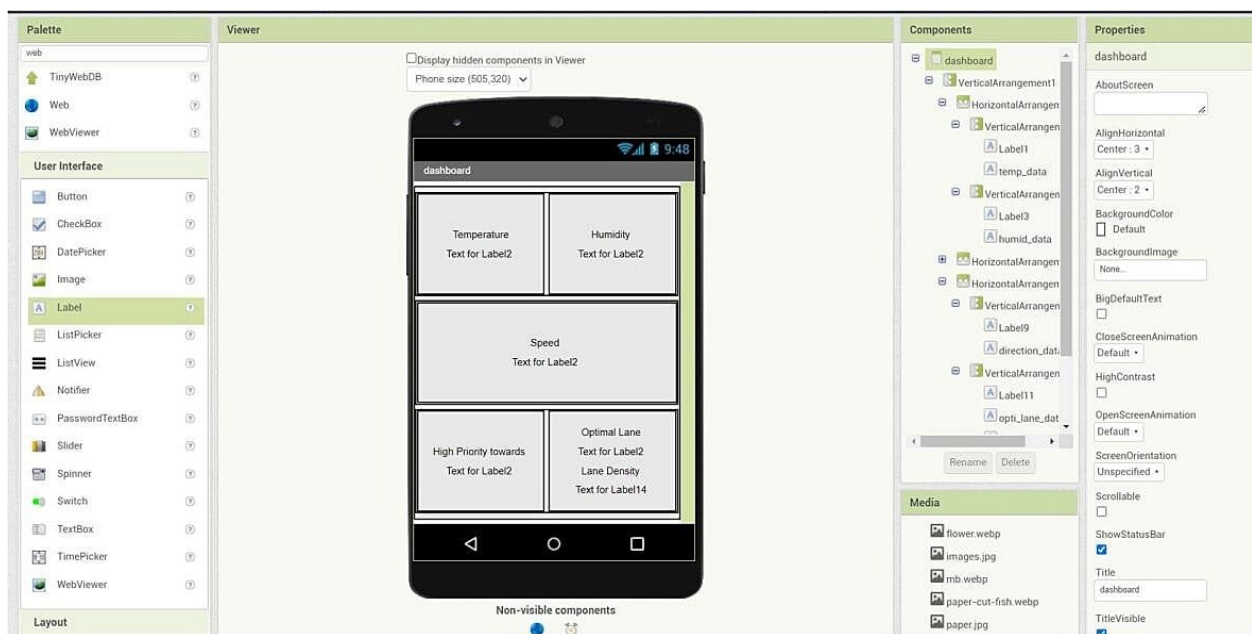
```

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;
  
```

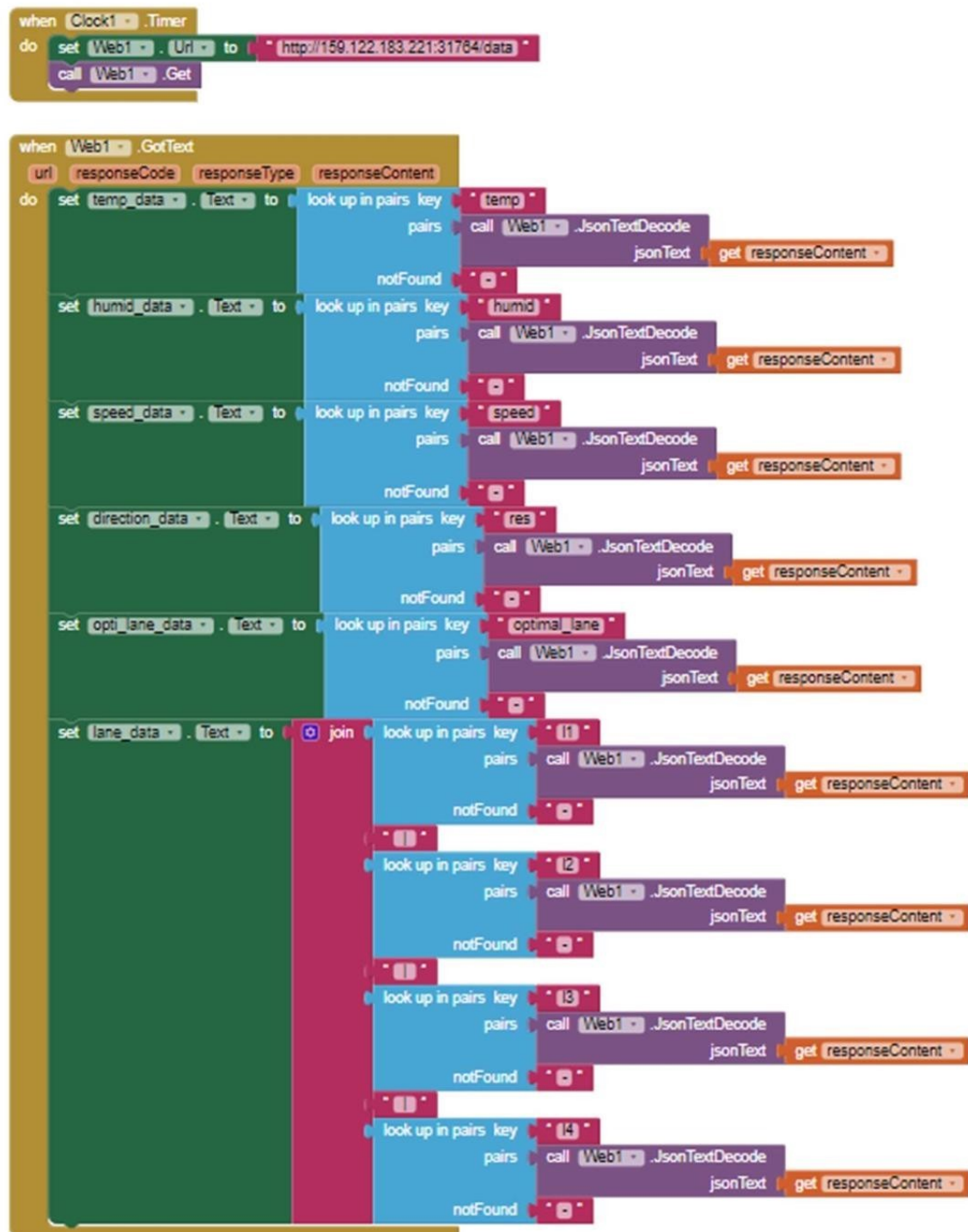
Output from Node red :



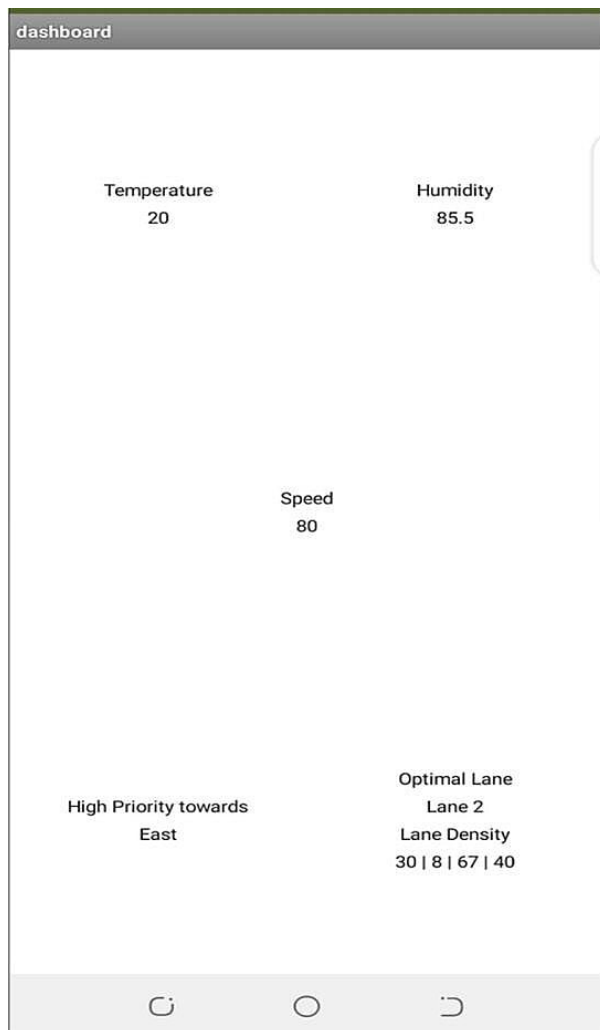
MIT App Inventor UI design:



MIT App Inventor Backend design:

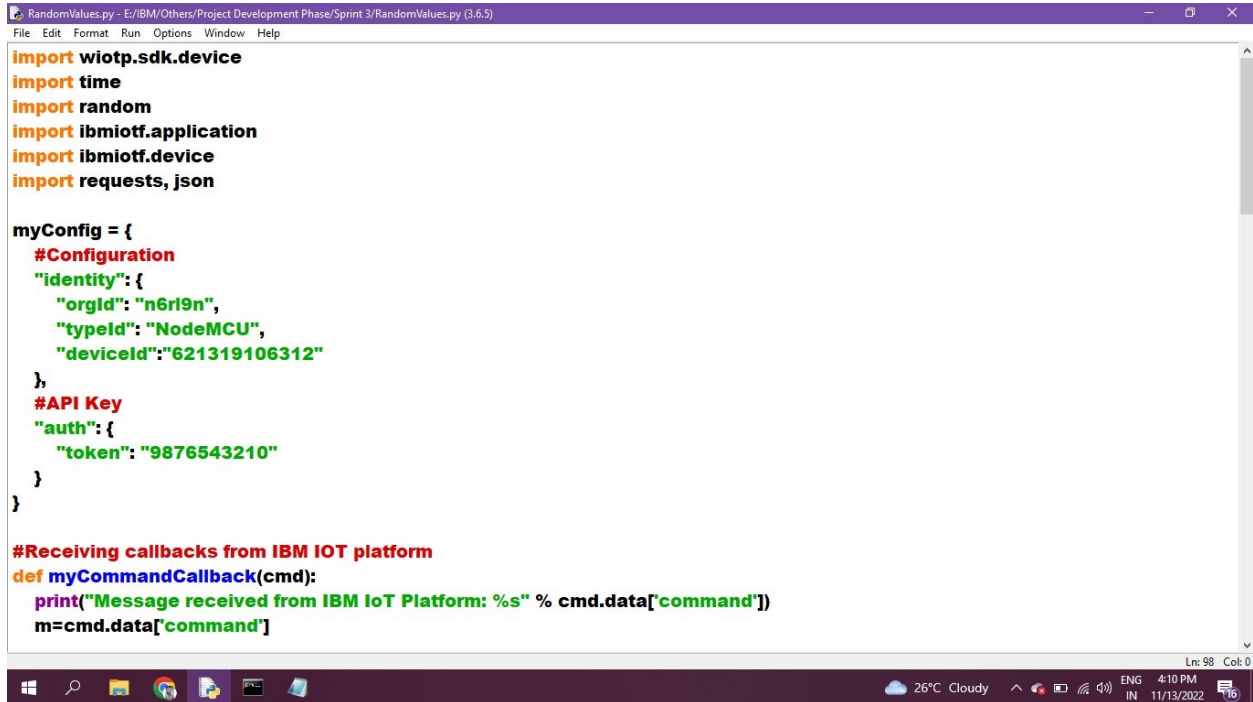


(OUTPUT) Display from MIT App:



9.2 User Acceptance Testing :

Python Simulation :



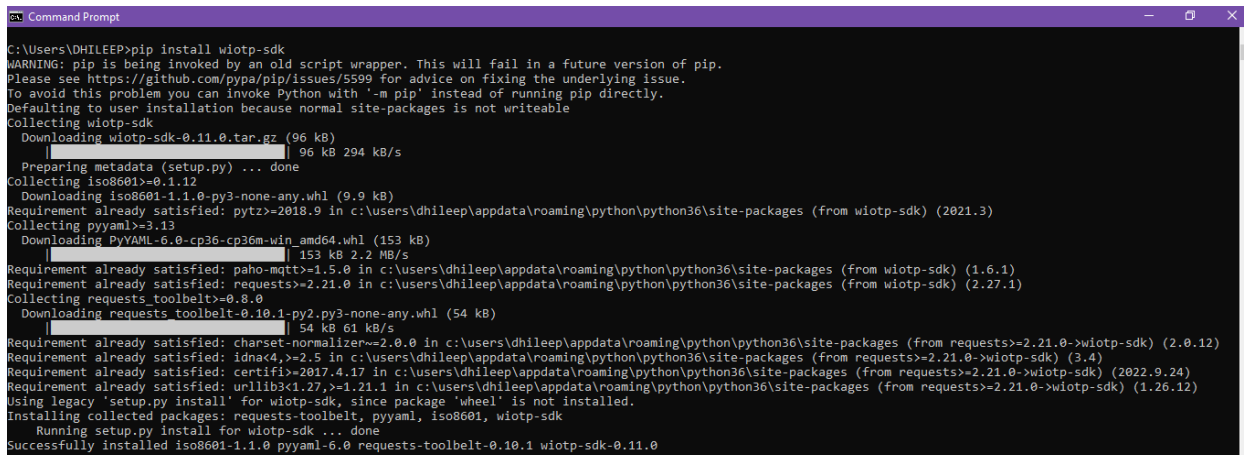
```

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"]
  
```

Import wiotp-sdk & ibmiotf :



```

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):

Weather in your city

Salem, IN

Salem, IN overcast clouds

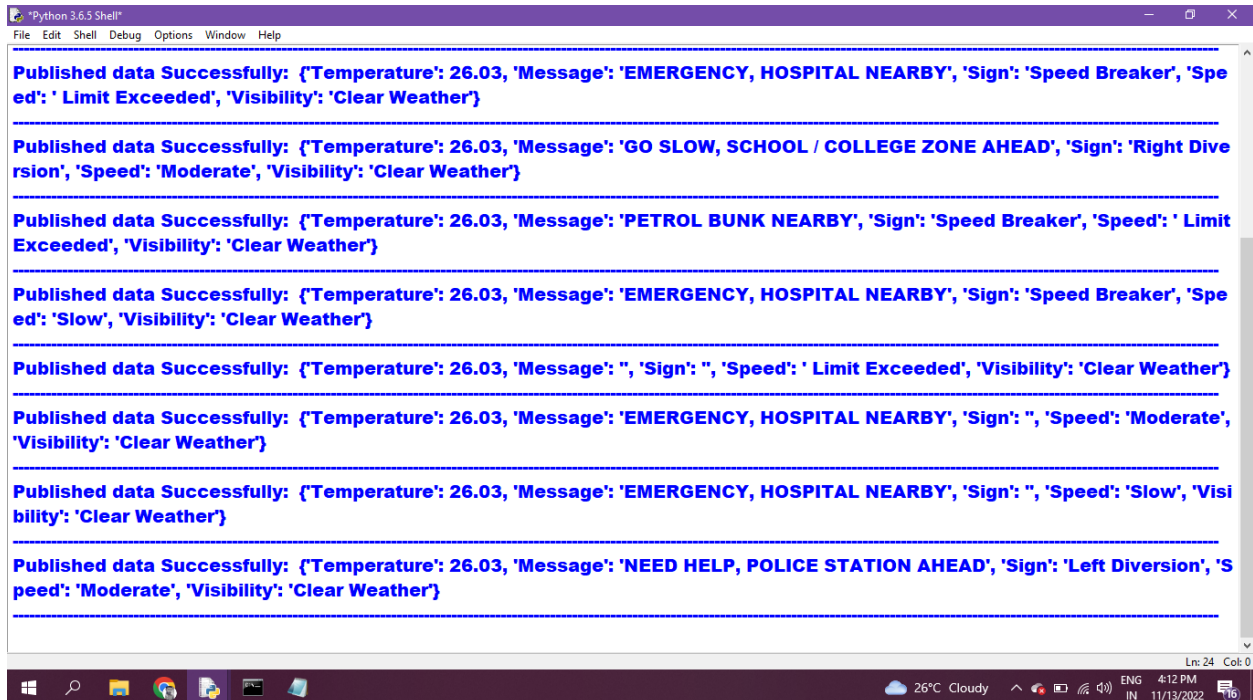
26.9°C temperature from 26.9 to 26.9 °C, wind 3 m/s, clouds 94 %, 1009 hpa

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 :



The screenshot shows a Python 3.6.5 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help) and a toolbar. The main text area displays seven lines of JSON output, each preceded by the text "Published data Successfully:". The JSON objects contain fields for Temperature, Message, Sign, Speed, and Visibility. The messages include "EMERGENCY, HOSPITAL NEARBY", "GO SLOW, SCHOOL / COLLEGE ZONE AHEAD", "PETROL BUNK NEARBY", and "NEED HELP, POLICE STATION AHEAD". The signs include "Speed Breaker", "Right Diversion", "Left Diversion", and empty strings. The speeds include "Limit Exceeded", "Moderate", and "Slow". The visibility for all entries is "Clear Weather". The window's status bar at the bottom shows "Ln: 24 Col: 0" and a system tray with weather information (26°C, Cloudy), date (11/13/2022), and time (4:12 PM).

```
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'}
```


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

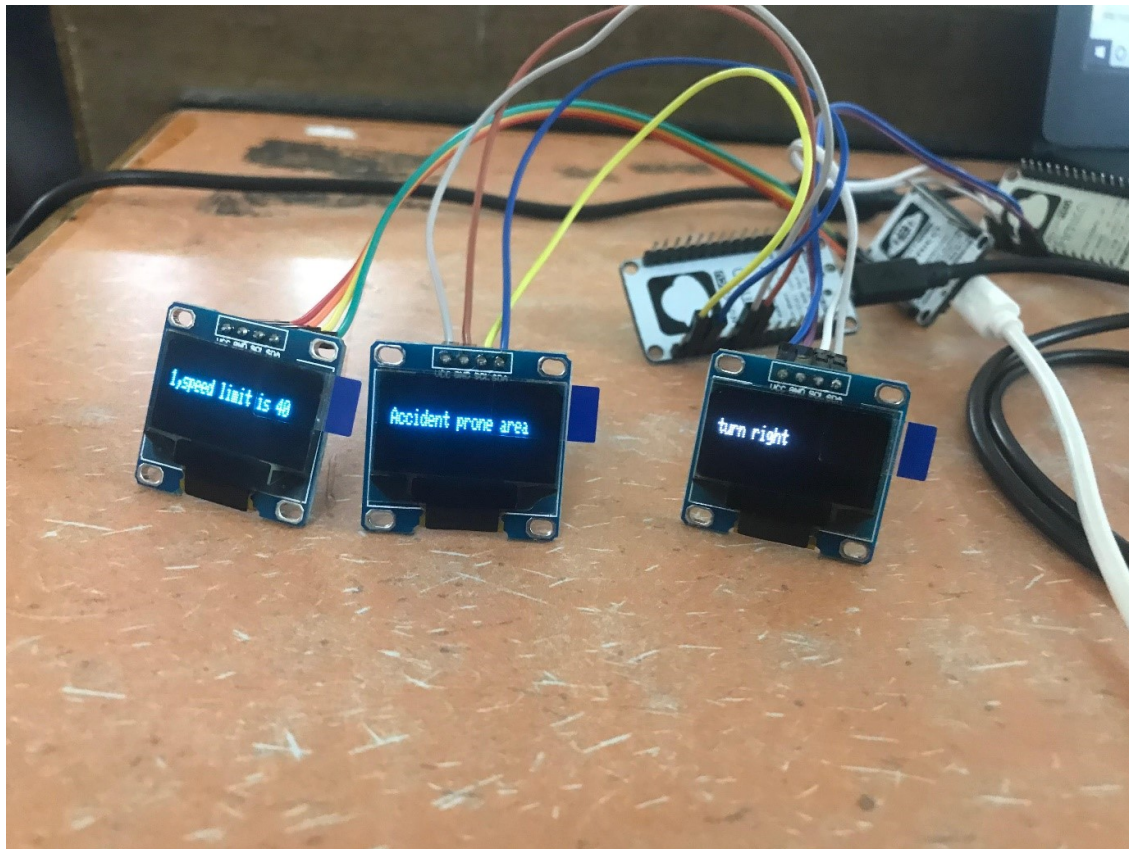


Fig 10 - This figure shows three OLEDs that we are displaying.

- (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, 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 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-10124-1659097522>