

Project Development Phase
Project Development Delivery of Sprint 4

Date	08 November 2022
Team ID	PNT2022TMID48721
Project Name	Project - Signs with smart connectivity for Better road safety
Marks	8 Marks

Objective :

- >> Write a python code for print the random temperature, Road signs, Speed limit, Message
- >> Simulate and Generate the data
- >> Display the published data in IBM Watson IOT Platform

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

(RandomValues.py)

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

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

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

#define ORG "k0y7f8" //IBM ORGANITION ID
#define DEVICE_TYPE "ESP32_CONTROLLER" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "BME280_SENSOR" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "Md22fj*aovUH7gy60x" //Token
String data3;
float dist;

//----- 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
data to be send
char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type AND COMMAND IS TEST OF
FORMAT STRING
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
parameter like server id,portand wificredential

int LED = 4;
int trig = 5;
int echo = 18;
```

```

void setup()
{
  Serial.begin(115200);
  pinMode(trig,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(LED, OUTPUT);
  delay(10);
  wificonnect();
  mqttconnect();
}
void loop()// Recursive Function
{

  digitalWrite(trig,LOW);
  digitalWrite(trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(trig,LOW);
  float dur = pulseIn(echo,HIGH);
  float dist = (dur * 0.0343)/2;
  Serial.print ("Distancein cm");
  Serial.println(dist);


  PublishData(dist);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}

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

void PublishData(float dist) {
  mqttconnect();//function call for connecting to ibm
  /*
    creating the String in in form JSoN to update the data to ibm cloud
  */
  String object;
  if (dist <100)
  {
    digitalWrite(LED,HIGH);
    Serial.println("object is near");
    object = "Near";
  }
  else
  {
    digitalWrite(LED,LOW);
    Serial.println("no object found");
    object = "No";
  }

  String payload = "{\"distance\":";
  payload += dist;
  payload += "," " \"object\"\":\"";
  payload += object;
  payload += "\"}";

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

```

```

    if (client.publish(publishTopic, (char*) payload.c_str())) {
        Serial.println("Publish ok");// if it successfully upload data on the cloud then it will print publish ok
        in Serial monitor or else it will print publish failed
    } else {
        Serial.println("Publish failed");
    }
}

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

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

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

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

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

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

    // Serial.println("data: "+ data3);
    // if(data3=="Near")
    // {
    // Serial.println(data3);
    // digitalWrite(LED,HIGH);

    // }

    // else
    // {

```

```

// Serial.println(data3);
// digitalWrite(LED,LOW);

// }
data3="";

}

```

Python Simulation :

```

*spit.py - C:/Users/ELCOT/AppData/Local/Programs/Python/Python310/spit.py (3.10.2)*
File Edit Format Run Options Window Help

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

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

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

#define ORG "k0y7f8" //IBM ORGANITION ID
#define DEVICE_TYPE "ESP32_CONTROLLER" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "BME280_SENSOR" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "Md22fj*aovUH7gy6Ox" //Token
String data3;
float dist;

//----- 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 data to be send
char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING
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 parameter like server id,portand wificredential

int LED = 4;
int trig = 5;
int echo = 18;
void setup()
{
  Serial.begin(115200);
  pinMode(trig,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(LED, OUTPUT);
  delay(10);
}

```

Ln: 20 Col: 119



Type here to search



28°C



22:53
ENG Wednesday
16-11-2022



Import wiotp-sdk & ibmiotf :

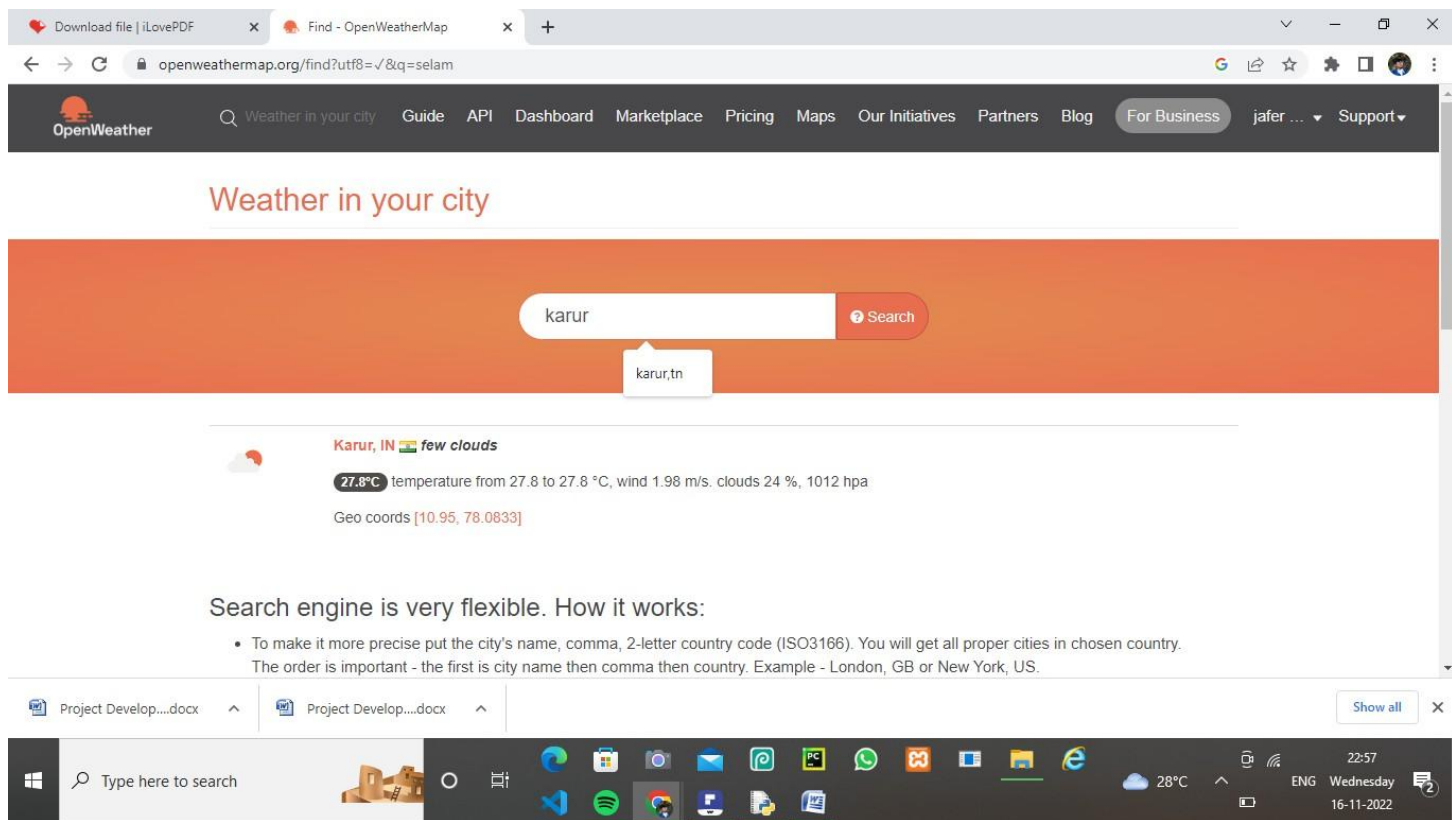
```
Command Prompt

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

```
Command Prompt

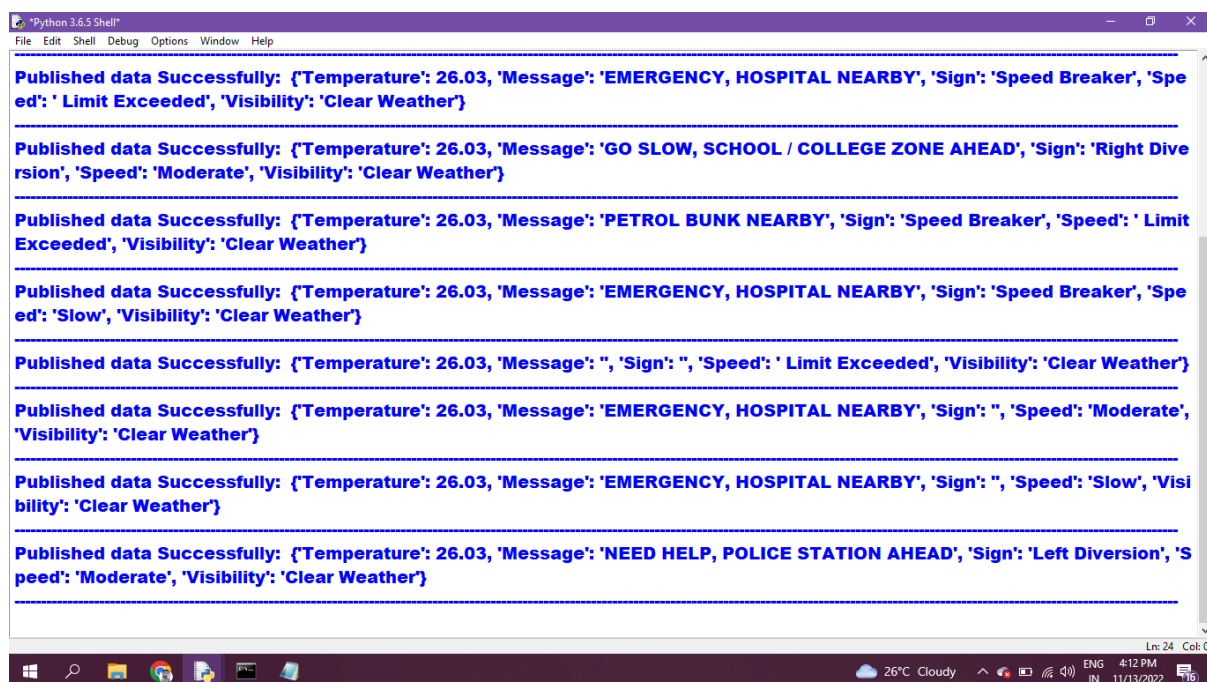
C:\Users\DHILEEP>pip install ibmiotf
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running 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., karur, IN) :



The screenshot shows the OpenWeatherMap website interface. The browser address bar displays the URL `openweathermap.org/find?utf8=✓&q=salam`. The website header includes the OpenWeather logo and navigation links: Weather in your city, Guide, API, Dashboard, Marketplace, Pricing, Maps, Our Initiatives, Partners, Blog, For Business, jafer ..., and Support. The main content area features a search bar with the text "karur" and a "Search" button. Below the search bar, a dropdown menu shows "karur,tn". The weather data for Karur, IN is displayed, showing a temperature of 27.8°C, a message "few clouds", and geo-coordinates [10.95, 78.0833]. A search engine flexibility note explains that users can specify city names and country codes (ISO3166) for more precise results, with examples like "London, GB" or "New York, US". The Windows taskbar at the bottom shows the search bar, taskbar icons, and system tray information including the date and time (22:57, Wednesday, 16-11-2022).

Python IDLE Output :



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help

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

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

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

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

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

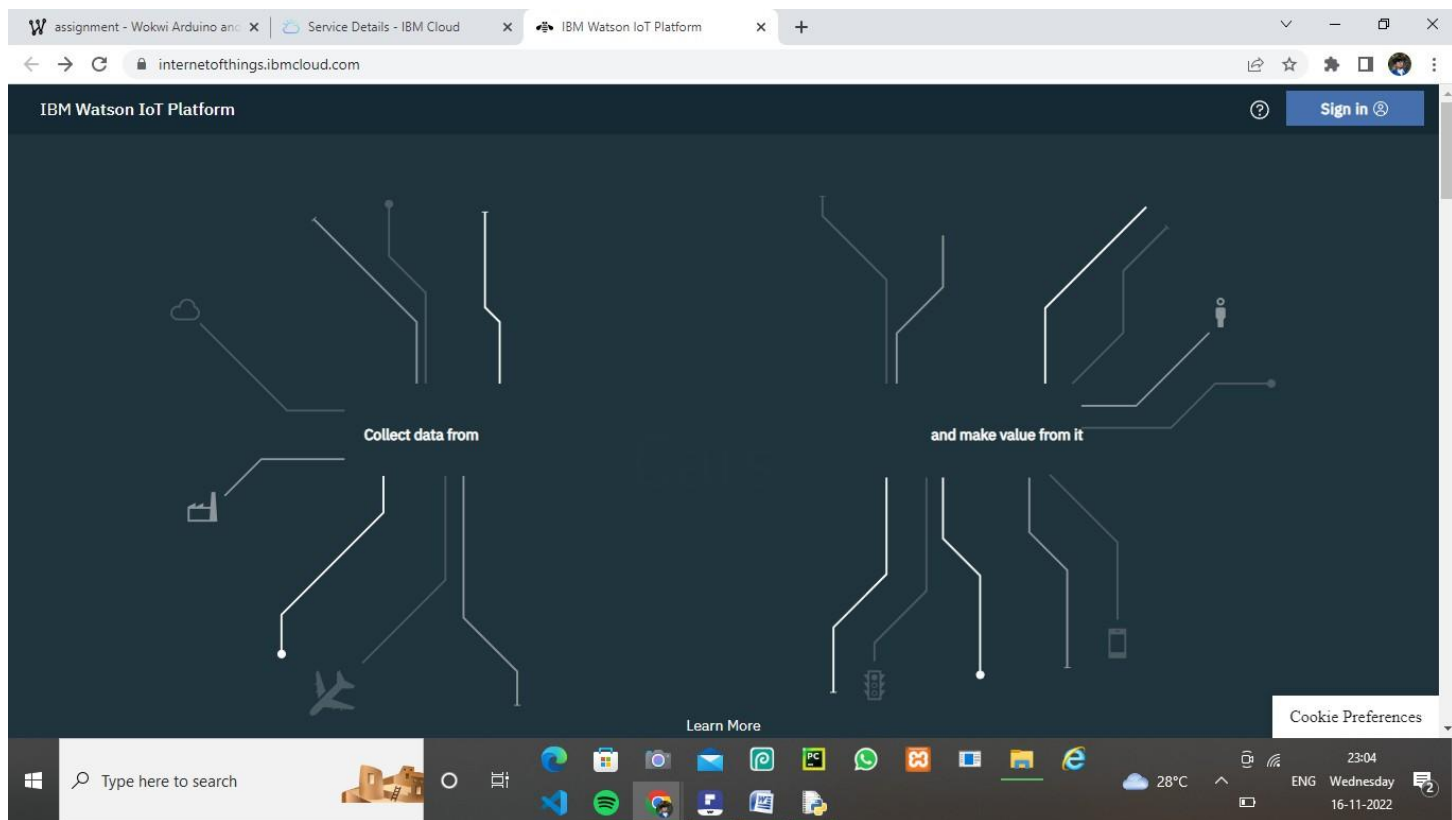
Published data Successfully: {'Temperature': 26.03, 'Message': 'EMERGENCY, HOSPITAL NEARBY', 'Sign': '', 'Speed': 'Moderate', 'Visibility': 'Clear Weather'}

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

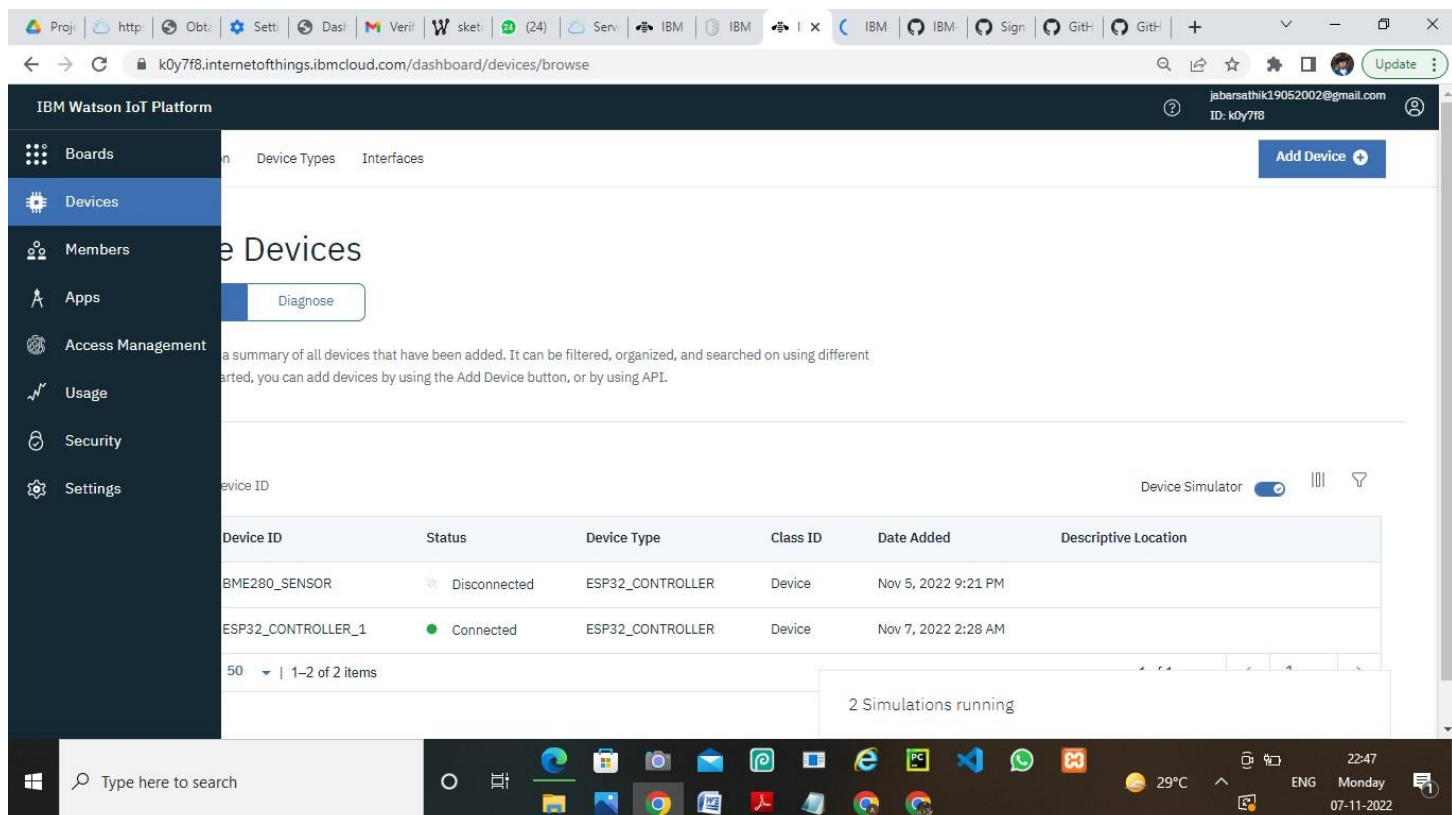
Published data Successfully: {'Temperature': 26.03, 'Message': 'NEED HELP, POLICE STATION AHEAD', 'Sign': 'Left Diversion', 'Speed': 'Moderate', 'Visibility': 'Clear Weather'}

Ln: 24 Col: 0
```

IBM Watson IOT Platform :



IBM Watson IOT Platform - Device Creation :



IBM Watson IOT Platform - Display the published data :

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present with the text 'Search by Device ID'. The main content area displays a table of devices. The selected device is 'BME280_SENSOR', which is 'Disconnected' and of type 'ESP32_CONTROLLER'. Below the table, the 'Recent Events' tab is active, showing a list of events. The events table has columns for 'Event', 'Value', 'Format', and 'Last Received'. The events listed are 'event_1' with values like '{\"randomNumber\":84}', '{\"randomNumber\":96}', '{\"randomNumber\":77}', '{\"randomNumber\":50}', and '{\"randomNumber\":34}', all in 'json' format, received 'a few seconds ago'. A status message at the bottom right indicates '2 Simulations running'.

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
BME280_SENSOR	Disconnected	ESP32_CONTROLLER	Device	Nov 5, 2022 9:21 PM	

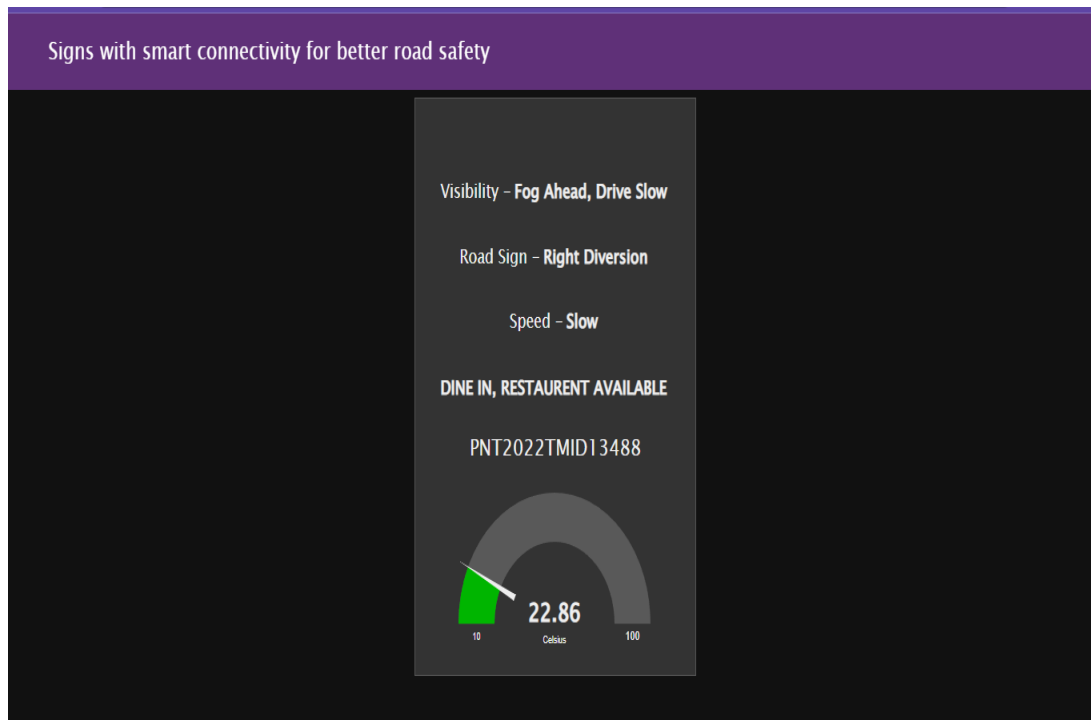
Event	Value	Format	Last Received
event_1	{\"randomNumber\":84}	json	a few seconds ago
event_1	{\"randomNumber\":96}	json	a few seconds ago
event_1	{\"randomNumber\":77}	json	a few seconds ago
event_1	{\"randomNumber\":50}	json	a few seconds ago
event_1	{\"randomNumber\":34}	json	a few seconds ago

Signs with smart connectivity for better road safety - Node-Red :

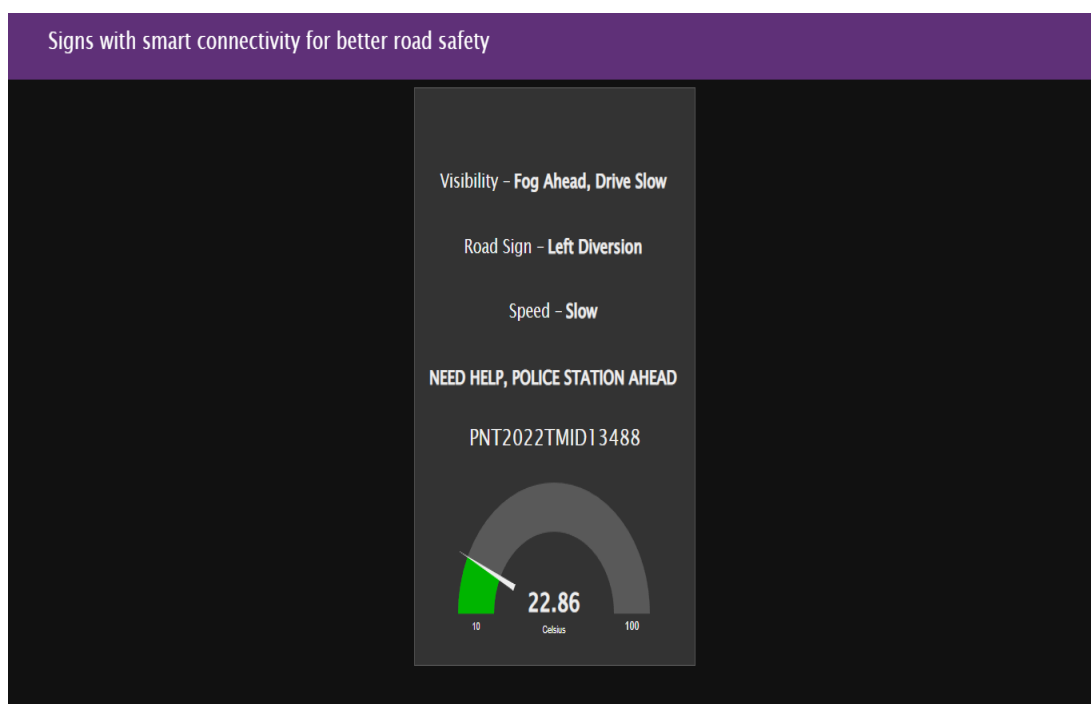
The screenshot shows a Node-RED workflow for processing data from an IBM IoT device. The workflow starts with an 'IBM IoT' node (labeled 'connected') which triggers a 'function' node. This function node outputs to a 'msg.payload' node, which then branches into several 'function' nodes: 'Temperature', 'Message', 'Speed', 'Sign', and 'Visibility'. Each of these function nodes is connected to a corresponding 'abc' node (e.g., 'Temperature abc', 'Message abc', 'Speed abc', 'Sign abc', 'Visibility abc'). These 'abc' nodes are then connected to 'audio out' nodes, which are labeled 'playing'. The right sidebar shows the 'debug' console with log messages. The messages are JSON objects containing temperature, message, speed, sign, and visibility data. For example, one message is: {\"Temperature\": 22.86, \"Message\": \"\", \"Sign\": \"Speed Breaker\", \"Speed\": \"Slow\", \"Visibility\": \"Fog Ahead, Drive Slow\"}.

```
graph LR
    IoT[IBM IoT] --> Function1[function]
    Function1 --> Payload[msg.payload]
    Payload --> Temp[function]
    Payload --> Message[function]
    Payload --> Speed[function]
    Payload --> Sign[function]
    Payload --> Visibility[function]
    Temp --> TempABC[Temperature abc]
    Message --> MessageABC[Message abc]
    Speed --> SpeedABC[Speed abc]
    Sign --> SignABC[Sign abc]
    Visibility --> VisibilityABC[Visibility abc]
    TempABC --> TempAudio[audio out]
    MessageABC --> MessageAudio[audio out]
    SpeedABC --> SpeedAudio[audio out]
    SignABC --> SignAudio[audio out]
    VisibilityABC --> VisibilityAudio[audio out]
```

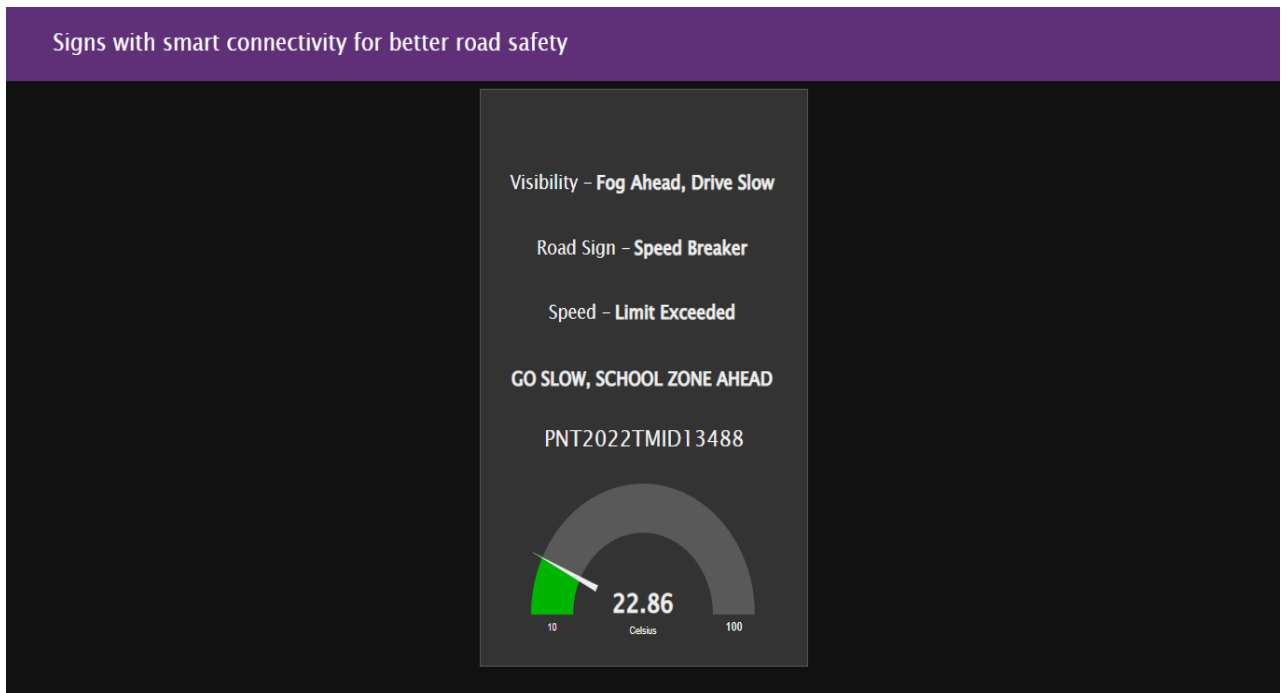

Test Case – 1



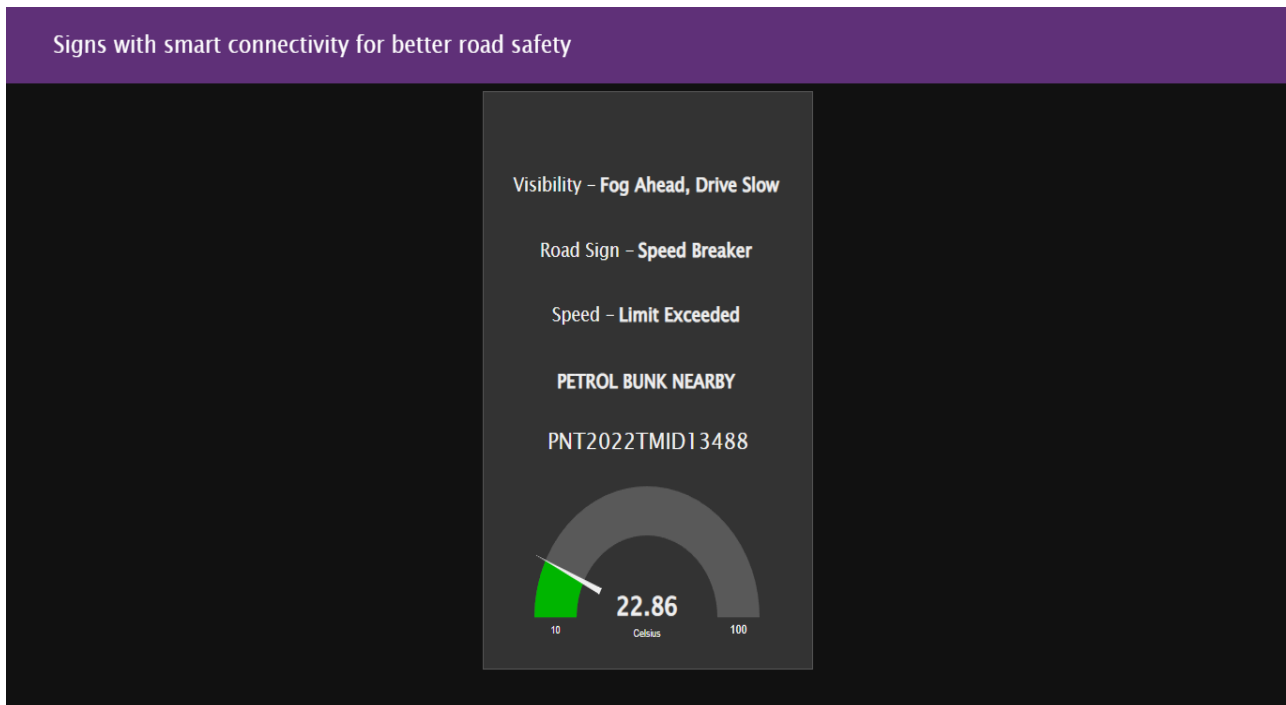
Test Case – 2



Test Case – 3



Test Case – 4



Test Case – 5

