**SMART CONNECTED SIGNS FOR IMPROVED ROAD SAFETY**

**Introduction:**

1. **Overview:**

“Enjoying the journey to reach the destination is also important”, they say.

Since ages, we have come across the static sign boards while travelling, which are important for the driver to understand the rules to be followed on that particular road. This project mainly revolves around the concept of evolution of digitalizing sign boards. Based on the weather parameters around and any fatal situation like accidents, we can control the sign boards remotely.

It cautions us to maintain the speed limit.

Integration of the hardware like high end raspberry pi by configuring it with coding can achieve this for the perfect working model.



1. **Purpose:**

“Enjoying the journey is also important”, they say.

The main aim of this project is to digitalize the sign boards on the roads. Generally, when it is raining; the roads tend to be slippery. The driver should be careful at this moment. Likewise, we see many more situations when we need to change the instructions instantly, detecting the weather conditions. Internet of things makes this possible. This technology helps us integrating sensors and hardware and act according to our instruction.

In case of static boards, we always cannot afford to update them as circumstances change.

**Literature Survey**

1. **Existing Problem:**

**The problem statements given are:**

1. To replace the static signboards with smart connected sign boards (digital sign boards).
2. Guide (Schools), Warning and Service (Hospitals, Restaurant) signs are also displayed accordingly.
3. Based on the weather changes the speed may increase or decrease.
4. These smart connected sign gets the data from the weather API and updates automatically.
5. **Proposed Solution:**

**The solution proposed for the above problem statements are**

1. We used OLED representing the digital sign boards.
2. Signs used for the school (🡪), hospital (+), Restaurant (||^^||);

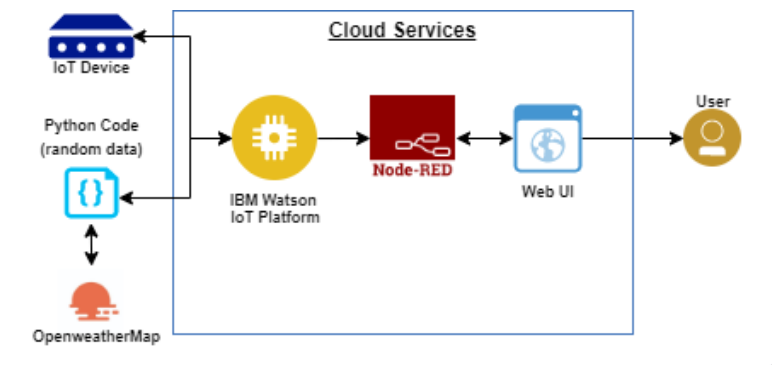
By configuring the buttons in the node red; we have displayed the signs of them.

1. The weather data is taken from the “Open Weather Map Organization Website” and accessed weather data of all the cities in India.
2. The “json” data received from the open weather map, the parameters we considered like temperature, humidity, weather description and visibility are extracted using a python code.
3. The controller who is responsible to control the sign boards remotely, can access the buttons of his requirement and update it instantly.

Theoretical Analysis:

1. **Block Diagram**:

The block diagram of our project is as follows:



1. **Hardware/Software Designing:**

**Hardware Designing:**

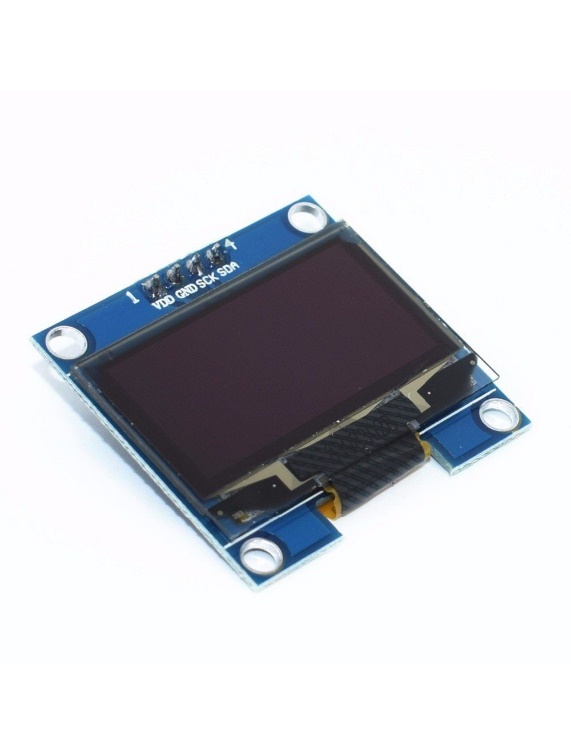
The hardware we used are:

1. ESP 32
2. OLED
3. **ESP32 :**



**ESP32** is a series of low-cost, low-power [system on a chip](https://en.wikipedia.org/wiki/System_on_a_chip) [microcontrollers](https://en.wikipedia.org/wiki/Microcontroller) with integrated [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) and dual-mode [Bluetooth](https://en.wikipedia.org/wiki/Bluetooth). The ESP32 series employs power amplifier, low-noise receive amplifier, filters, and power-management modules. It is a successor to the [ESP8266](https://en.wikipedia.org/wiki/ESP8266) microcontroller.

1. **OLED:**

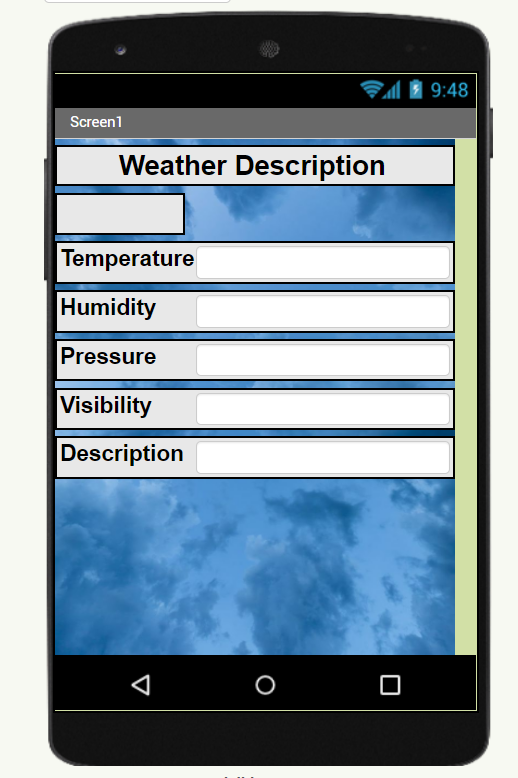


The image shows a 128x64 I2C based OLED module. These OLED do not require backlight since the display creates its own light. Hence it consumes less power. SCL and SDA are the serial clock and serial data pins for I2C interface. Power supply of 3.3V to 5V can be provided to its Vcc.

**Software designing:**

We have used the MIT App Inventor to design the app.

**MIT App Inventor** is an online platform designed for development of mobile applications. We can design mobile applications by using the components. Both front-end and back-end developments are possible. It has Bluetooth and Wi-Fi communication protocols. We have designed an application to display the weather parameters.



Experimentally Investigations:

OLED INTEGRATED WITH THE HELP OF ARDUINO IDE

THE WEB UI IS CREATED

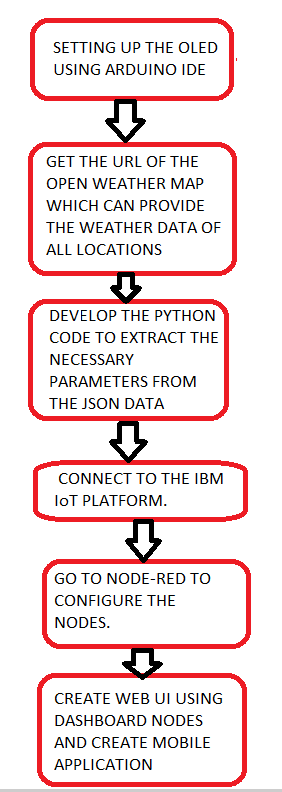
CONNECTED TO IBM CLOUD

EXTRACTED WEATHER DATA FROM OPEN WEATHER MAP

MOBILE APP RECEIVES NOTIFICATIONS BY WEATHER

OLED REPRESENTING SHOWS THE WARNINGS AND INSTRUCTIONS!

Flow chart:



Result:

We finally came up with the project where static boards can be replaced with the digital sign boards indicating the warnings like speed limits, diversions and any other updates to be given by controlling it remotely.

Mobile app was designed to track the weather data like visibility which plays a key role while driving.

**Note:**

**In this pandemic situation; we could not work it out with more equipped and most advanced hardware like raspberry pi and digital display.**

**Advantages:**

1. Replacing the static sign boards repeatedly would be a tedious process which we can overcome through this idea.
2. This will help the people understand the circumstances of the road more effectively.
3. This would be the one-time investment. That is when a digital board is placed, it can be controlled remotely.
4. Process becomes fast.
5. This can prevent accidents.

Disadvantages:

1. This can be costly.
2. Digital boards should be protected from the climatic conditions like heavy rains to avoid flaws.
3. Power consumption would be continuous.

Applications:

For bringing a new innovation in the field of Road management, this project would contribute its part. The digital sign boards are placed at the cross roads with the signals, indicating the people to stop or move. On the highways, these digital boards convey us the speed limit. When the road is under maintenance, it would even display the message about it, so that the passenger can understand and change his/her route.

Conclusion:

This project sets a revolution and ensures the happy journey of the passengers on the road or even on the tracks if we use the higher end technologies. This reduces so much of effort of the maintainers. This so called automation can be developed furthermore with the consistent amount of hardware.

Future Scope:

The more appropriate data we fetch with advanced API’s, this project can be developed in a better way, perhaps we having the consistent amount of knowledge. This could be a remarkable change in the road management sector and ensures road safety.

**Bibliography:**

<https://iotdunia.com/smart-roads/>

<https://www.mbengineering.in/road-safety-sign.html>

Appendix:

Source Code:

Arduino Code for integrating OLED to IBM IoT cloud:

#include <WiFi.h>

#include <PubSubClient.h>

#include <Wire.h>

#include <Adafruit\_GFX.h>

#include <Adafruit\_SSD1306.h>

#define SCREEN\_WIDTH 128 // OLED display width, in pixels

#define SCREEN\_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1);

String command;

String data="";

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

// CHANGE TO YOUR WIFI CREDENTIALS

const char\* ssid = "Network Error";//your wifi ssid

const char\* password = "Norw&y96##";//your password

// CHANGE TO YOUR DEVICE CREDENTIALS AS PER IN IBM BLUMIX

#define ORG "n8qnr4"

#define DEVICE\_TYPE "ESP32"

#define DEVICE\_ID "90596"

#define TOKEN "@07102001" // Authentication Token OF THE DEVICE

// PIN DECLARATIONS

#define led1 2

#define led2 0

//-------- Customise the above values --------

const char publishTopic[] = "iot-2/evt/Data/fmt/json";

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char topic[] = "iot-2/cmd/home/fmt/String";// cmd REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

WiFiClient wifiClient;

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

int publishInterval = 5000; // 30 seconds

long lastPublishMillis;

void publishData();

void setup() {

Serial.begin(115200);

Serial.println();

pinMode(led1,OUTPUT);

pinMode(led2,OUTPUT);

wifiConnect();

mqttConnect();

delay(2000);

Serial.println("oled test");

if(!display.begin(SSD1306\_SWITCHCAPVCC, 0x3C)) {

Serial.println("SSD1306 allocation failed");

for(;;);

}

}

void loop() {

if (millis() - lastPublishMillis > publishInterval)

{

publishData();

lastPublishMillis = millis();

}

if (!client.loop()) {

mqttConnect();

}

}

void wifiConnect() {

Serial.print("Connecting to "); Serial.print(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.print("WiFi 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++) {

command+= (char)payload[i];

}

Serial.print("data: "+ command);

control\_func();

command= "";

}

void control\_func()

{

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

display.setCursor(0, 10);

// Display static text

display.println(command);

delay(1500);

display.display();

}

void publishData()

{

}

**Python Code for extracting json weather data from open weather map:**

import wiotp.sdk.device

import time

import random

import requests, json

myConfig = {

"identity": {

"orgId": "n8qnr4",

"typeId": "ESP32",

"deviceId":"90596"

},

"auth": {

"token": "@07102001"

}

}

api\_key = "902856e07c32938df0024a47e76f9ae4"

base\_url = "http://api.openweathermap.org/data/2.5/weather?"

city\_name = input("Enter city name : ")

complete\_url = base\_url + "appid=" + api\_key + "&q=" + city\_name

response = requests.get(complete\_url)

x = response.json()

def myCommandCallback(cmd):

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

m=cmd.data['command']

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

client.connect()

while True:

if x["cod"] != "404":

# store the value of "main"

# key in variable y

y = x["main"]

# store the value corresponding

# to the "temp" key of y

current\_temperature = y["temp"]

# store the value corresponding

# to the "pressure" key of y

current\_pressure = y["pressure"]

# store the value corresponding

# to the "humidity" key of y

current\_humidity = y["humidity"]

# store the value of "weather"

# key in variable z

z = x["weather"]

# store the value corresponding

# to the "description" key at

# the 0th index of z

weather\_description = z[0]["description"]

#storing value corresponding to visibility

visibility=x["visibility"]

myData={'temperature':current\_temperature, 'humidity':current\_humidity, 'description':weather\_description, 'visibility':visibility }

client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)

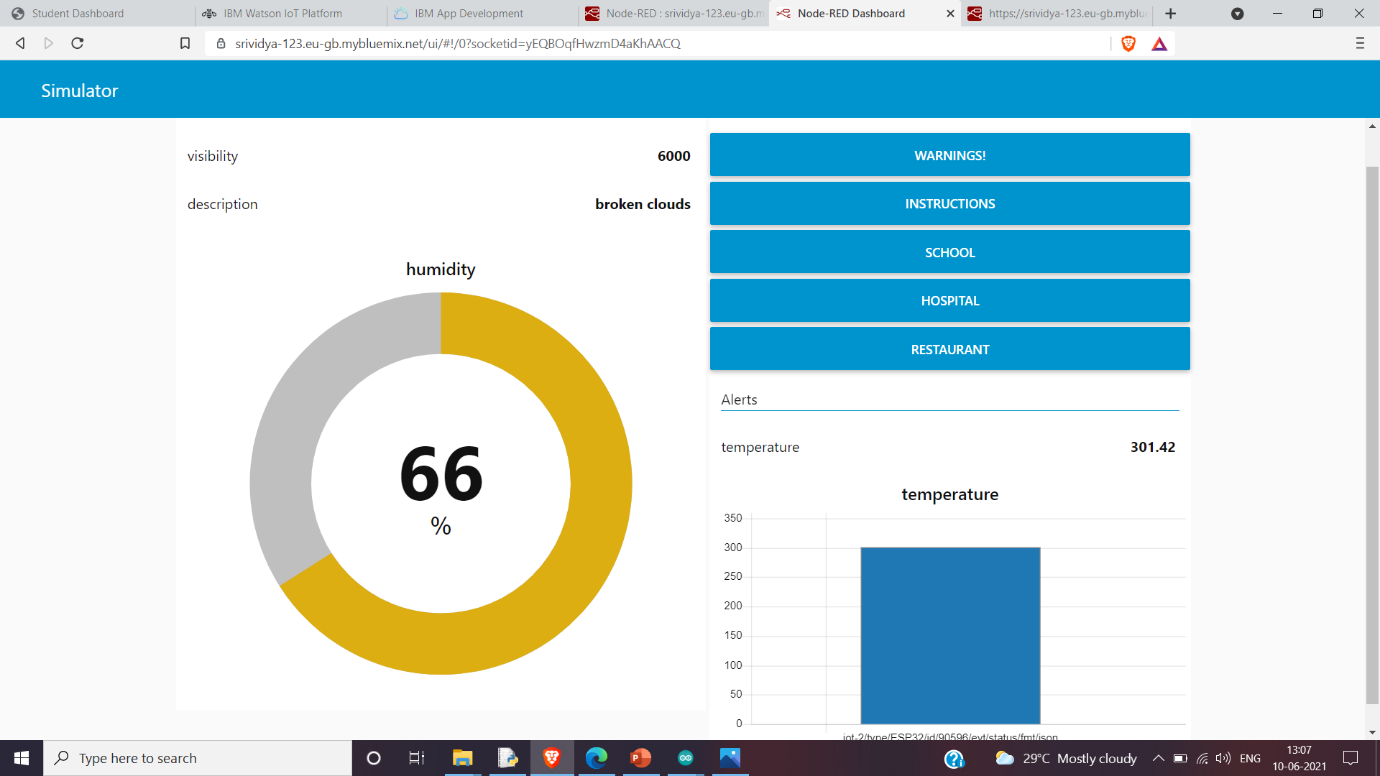
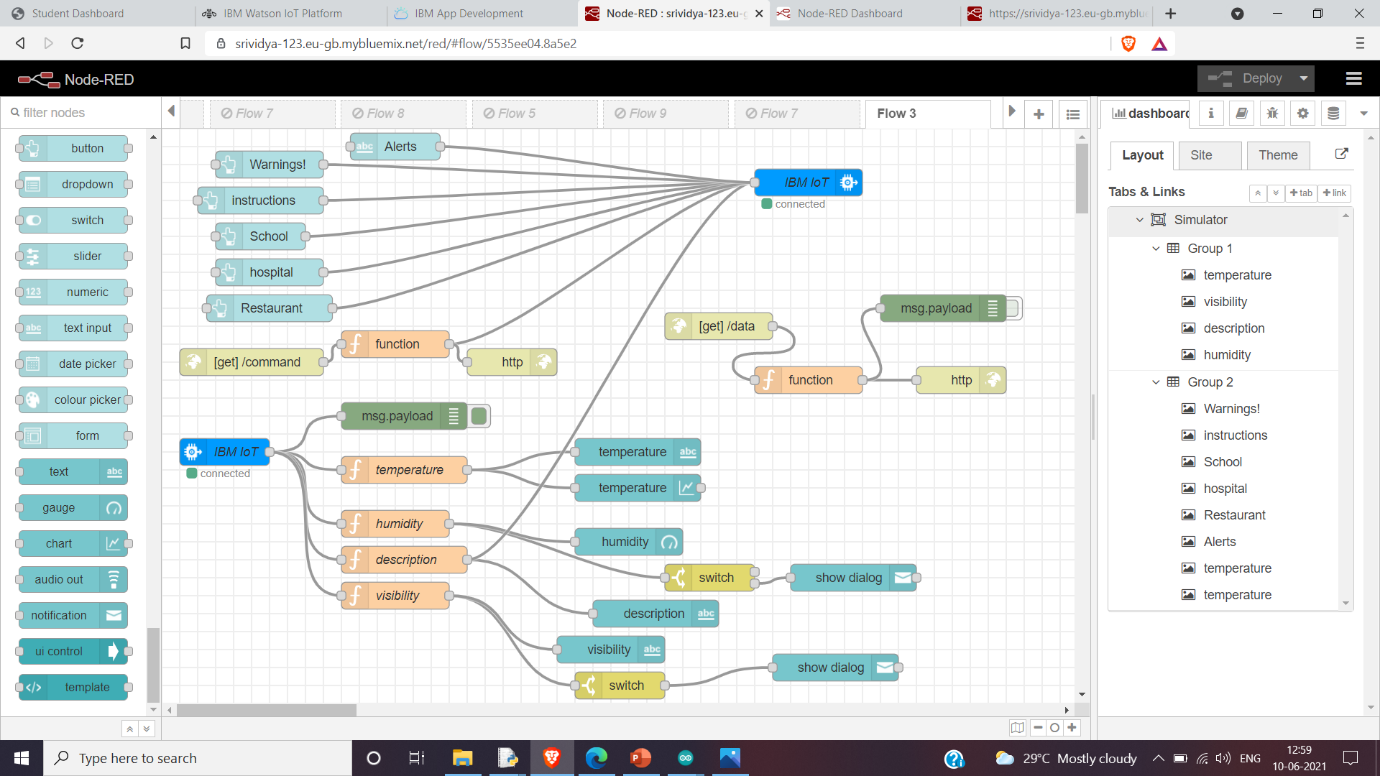
print("Published data Successfully: %s", myData)

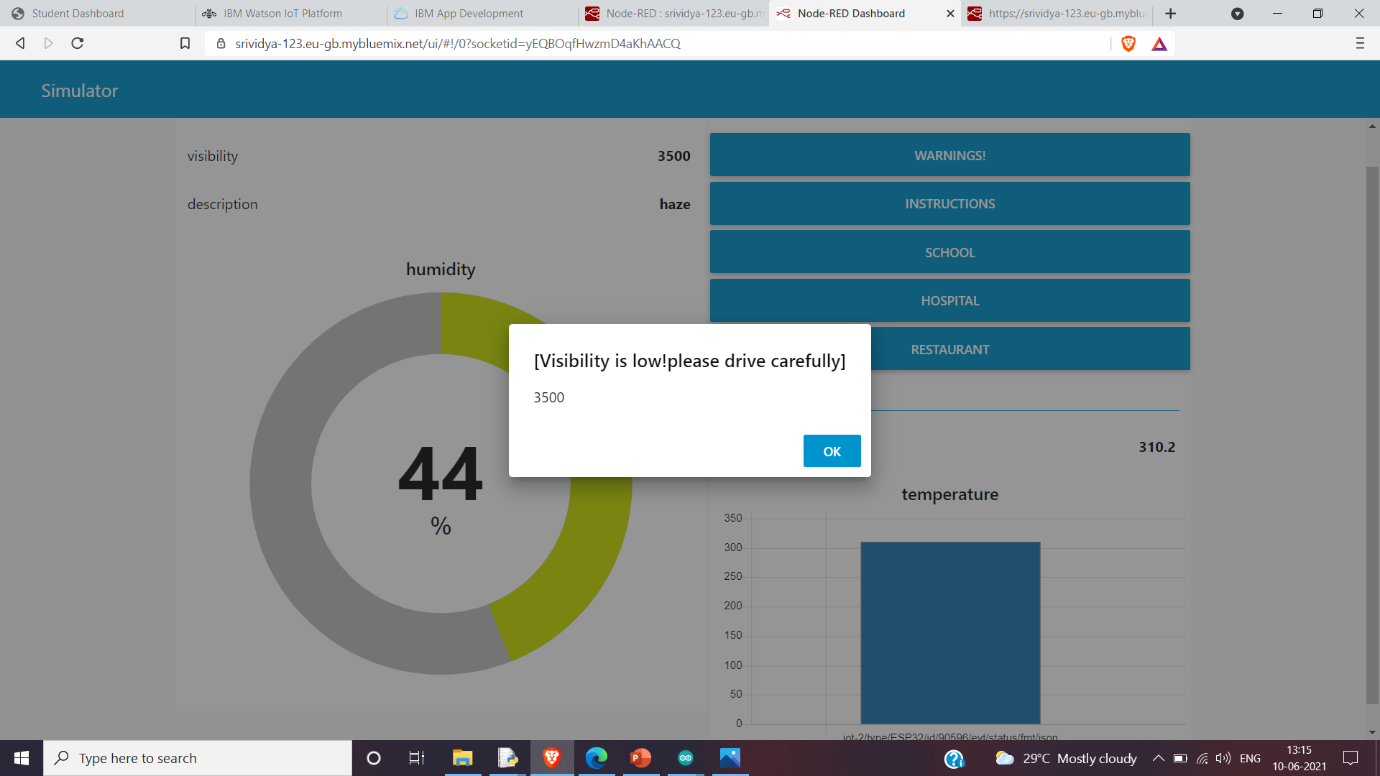
client.commandCallback = myCommandCallback

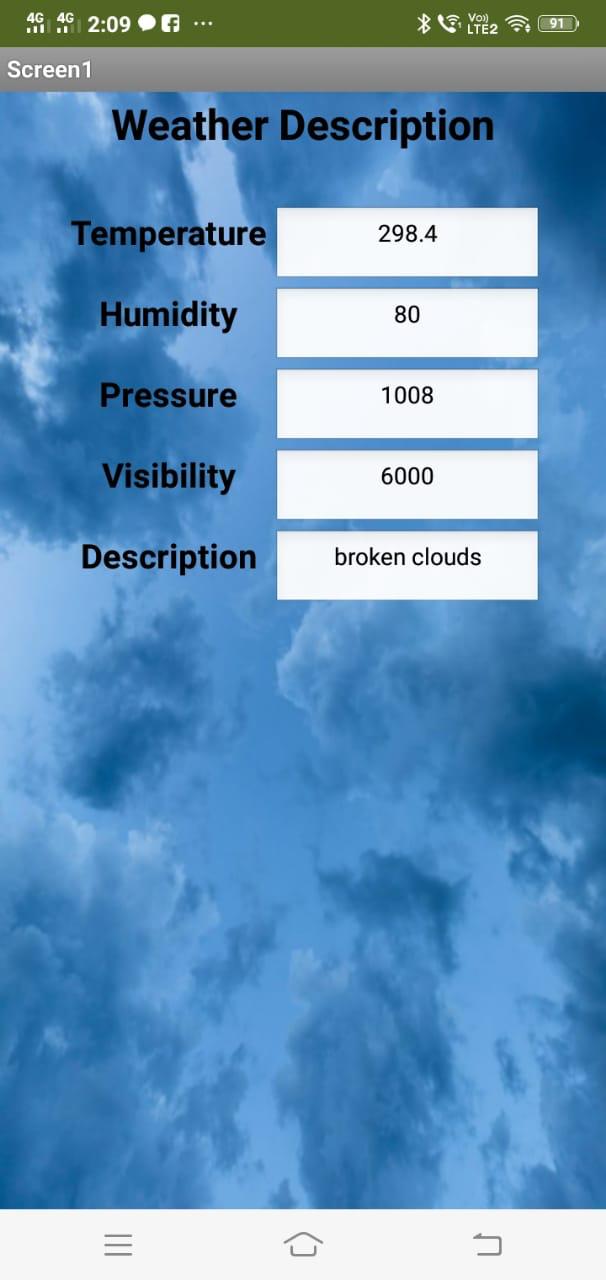
time.sleep(2)

client.disconnect()

UI outputs:







Project done by:

Sai Eshwar (19R11A0450)

B. Rachana(19R11A0453)

C. Hemalatha(19R11A0456)

C. Shushanth Kumar Reddy(19R11A0457)

G. Srividya(19R11A0463)