

## SPRINT 3

### INITIALIZING HARDWARE

|              |  |
|--------------|--|
| Team ID      | PNT2022TMID52810   |
| Project Name | Project - Signs with smart connectivity for Better road safety |

Integrate hardware that can connect to cloud services and modify output based on output processing.

#### **PYTHON SCRIPT:**

```
import wiotp.sdk.device
import time
import random
import requests,json
import datetime
myConfig = {
    "identity": {
        "orgId": "tmwrsv",
        "typeId": "Sprint",
        "deviceId": "sprint12"
    },
    "auth": {
        "token": "KxMwjzjw)BijreluFk"
    }
}
CITY = "Coimbatore"
API_KEY = "9111b726e6aa664188c5a2924f15f78e"
URL=
"https://api.openweathermap.org/data/2.5/weather?q=Coimbatore,%20IN&appid=9111b726e6aa664188c5a2924f15f78e"
response = requests.get(URL)
if response.status_code == 200:
    data = response.json()
    main = data['main']
    temp = round(main['temp'] - 273,2)
    humy = main['humidity']
    pres = main['pressure']
    rept = data['weather']
    report = rept[0]['description']
    time = datetime.datetime.now()
    morning = time.replace(hour=11, minute=59, second=0, microsecond=0)
    if time <= morning:
        me = '8.30 AM - 9.30 AM'
    else:
        me = '3.45 PM - 5.00 PM'
```

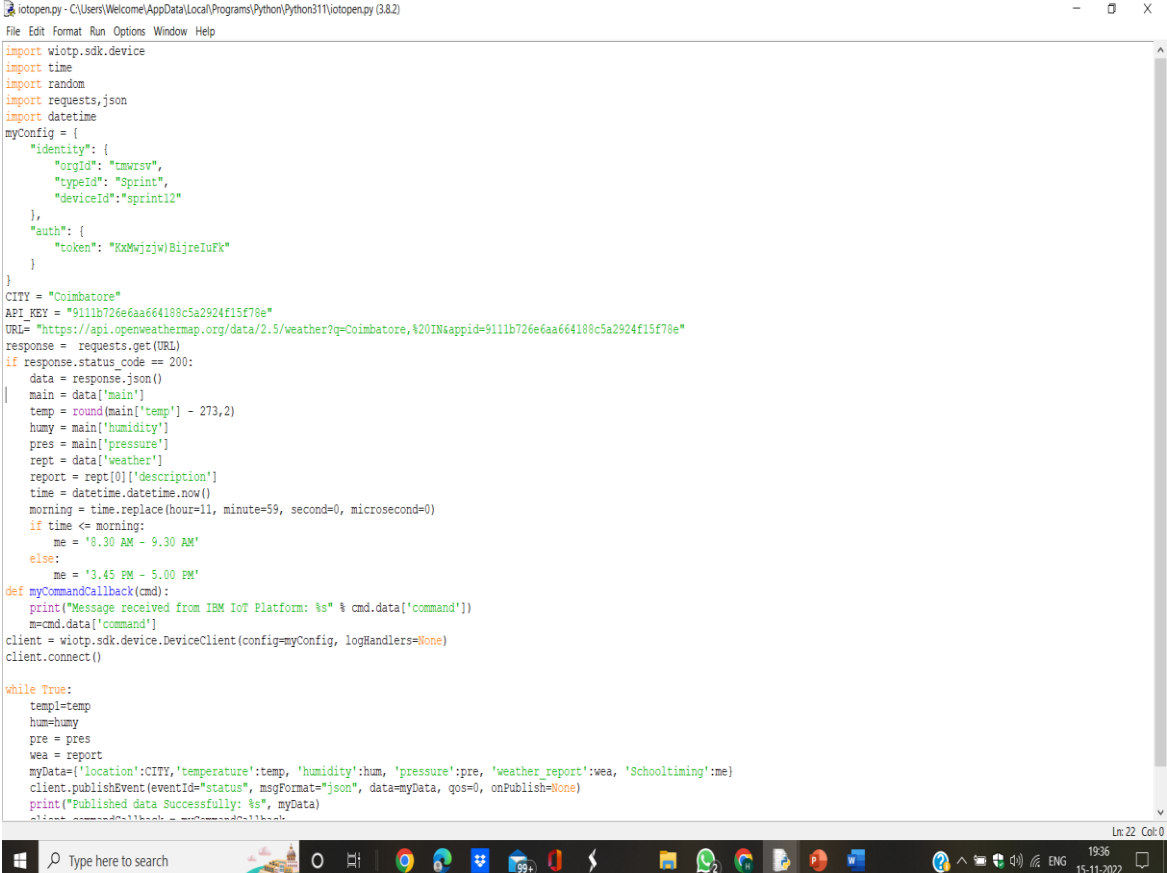
```

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:
    temp1=temp
    hum=humy
    pre = pres
    wea = report
    myData={'location':CITY,'temperature':temp, 'humidity':hum, 'pressure':pre,
'weather_report':wea, 'Schooltiming':me}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0,
onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback
client.disconnect()

```

## **PYTHON IDE:**



The screenshot shows a Python IDE window titled "iotopen.py - C:\Users\Welcome\AppData\Local\Programs\Python\Python311\iotopen.py (3.8.2)". The code in the editor is as follows:

```

import wiotp.sdk.device
import time
import random
import requests,json
import datetime
myConfig = {
    "identity": {
        "orgId": "tmwrsrv",
        "typeId": "Sprint",
        "deviceId": "sprint12"
    },
    "auth": {
        "token": "KxMwjzjwBijreIuFk"
    }
}
CITY = "Coimbatore"
API_KEY = "9111b726e6aa664188c5a2924f15f78e"
URL= "https://api.openweathermap.org/data/2.5/weather?q=Coimbatore,%20IN&appid=9111b726e6aa664188c5a2924f15f78e"
response = requests.get(URL)
if response.status_code == 200:
    data = response.json()
    main = data['main']
    temp = round(main['temp'] - 273,2)
    humy = main['humidity']
    pres = main['pressure']
    rept = data['weather']
    report = rept[0]['description']
    time = datetime.datetime.now()
    morning = time.replace(hour=11, minute=59, second=0, microsecond=0)
    if time <= morning:
        me = '8.30 AM - 9.30 AM'
    else:
        me = '3.45 PM - 5.00 PM'
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:
    temp1=temp
    hum=humy
    pre = pres
    wea = report
    myData={'location':CITY,'temperature':temp, 'humidity':hum, 'pressure':pre, 'weather_report':wea, 'Schooltiming':me}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Published data Successfully: %s", myData)
    client.commandCallback = myCommandCallback

```

The IDE interface includes a menu bar (File, Edit, Format, Run, Options, Window, Help) and a status bar at the bottom showing "Ln 22 Col 0". The Windows taskbar is visible at the very bottom of the image.

### PYTHON IDE OUTPUT:

[illegible]

### IBM WATSON IOT PLATFORM OUTPUT:

IBM Watson IoT Platform

2004201ec@cit.edu.in  
ID: tmwrsv

Browse Action Device Types Interfaces

Add Device

All Devices Diagnose

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Search by Device ID

Device Simulator

|   | Device ID   | Status       | Device Type | Class ID | Date Added            | Descriptive Location |
|---|-------------|--------------|-------------|----------|-----------------------|----------------------|
| > | 1110        | Disconnected | iot_new     | Device   | Nov 12, 2022 4:15 PM  |                      |
| > | sprint12    | Connected    | Sprint      | Device   | Nov 14, 2022 11:06 PM |                      |
| > | weather_tdy | Disconnected | weather     | Device   | Nov 13, 2022 7:09 PM  |                      |

Items per page 50 | 1-3 of 3 items

1 of 1 page

1 Simulation running

tmwrsv.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

2004201ec@cit.edu.in  
ID: tmwrsv

Browse Action Device Types Interfaces

Add Device +

Identity Device Information Recent Events State Logs

The recent events listed show the live stream of data that is coming and going from this device.

| Event  | Value  | Format | Last Received     |
|--------|--|--------|-------------------|
| status | {"location":"Coimbatore","temperature":26.03,"h... | json   | a few seconds ago |
| status | {"location":"Coimbatore","temperature":26.03,"h... | json   | a few seconds ago |
| status | {"location":"Coimbatore","temperature":26.03,"h... | json   | a few seconds ago |
| status | {"location":"Coimbatore","temperature":26.03,"h... | json   | a few seconds ago |
| status | {"location":"Coimbatore","temperature":26.03,"h... | json   | a few seconds ago |

> ☐ weather\_tdy ☐ Disconnected weather Device Nov 15 11:02 PM 1 Simulation running

Items per page 50 1-3 of 3 items

tmwrsv.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

2004201ec@cit.edu.in  
ID: tmwrsv

Browse Action Device Types Interfaces

Add Device +

Identity Device Information

The recent events listed show the live

| Event  | Value             |
|--------|-------------------|
| status | {"location":"Coin |
| status | {"location":"Coin |
| status | {"location":"Coin |
| status | {"location":"Coin |
| status | {"location":"Coin |

### Event Payload

Event Name status

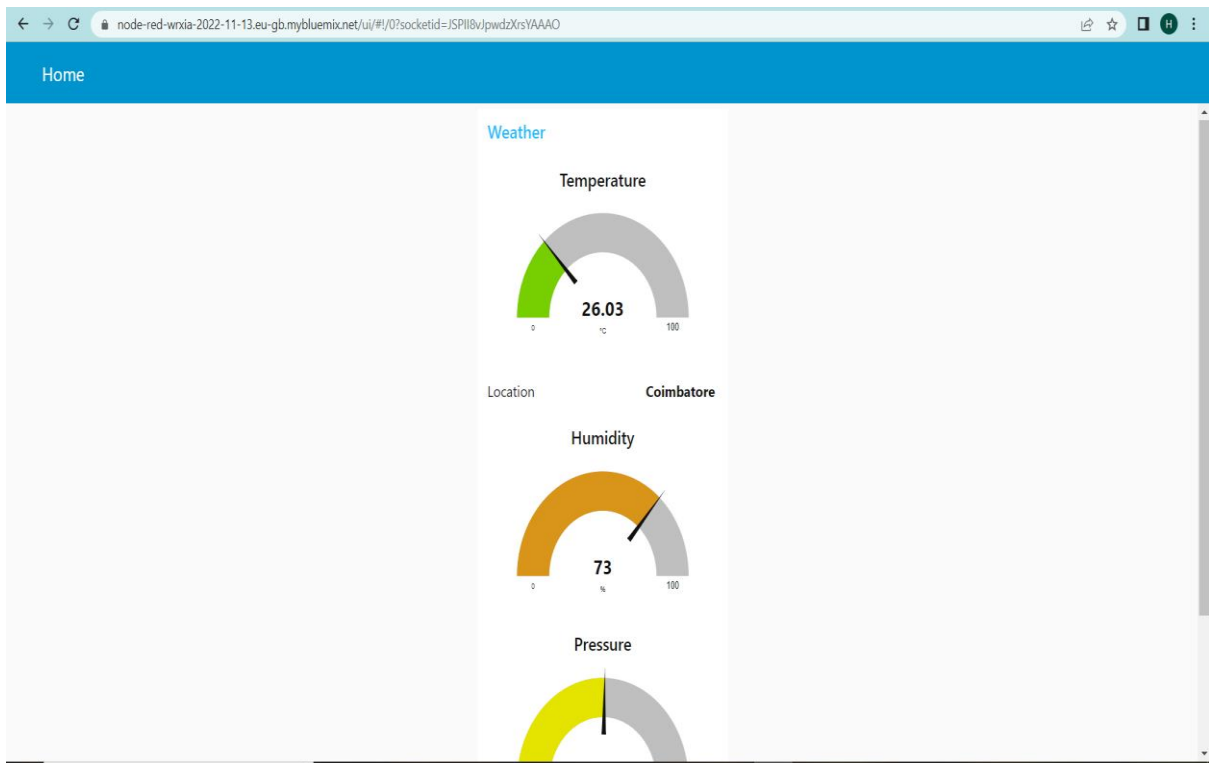
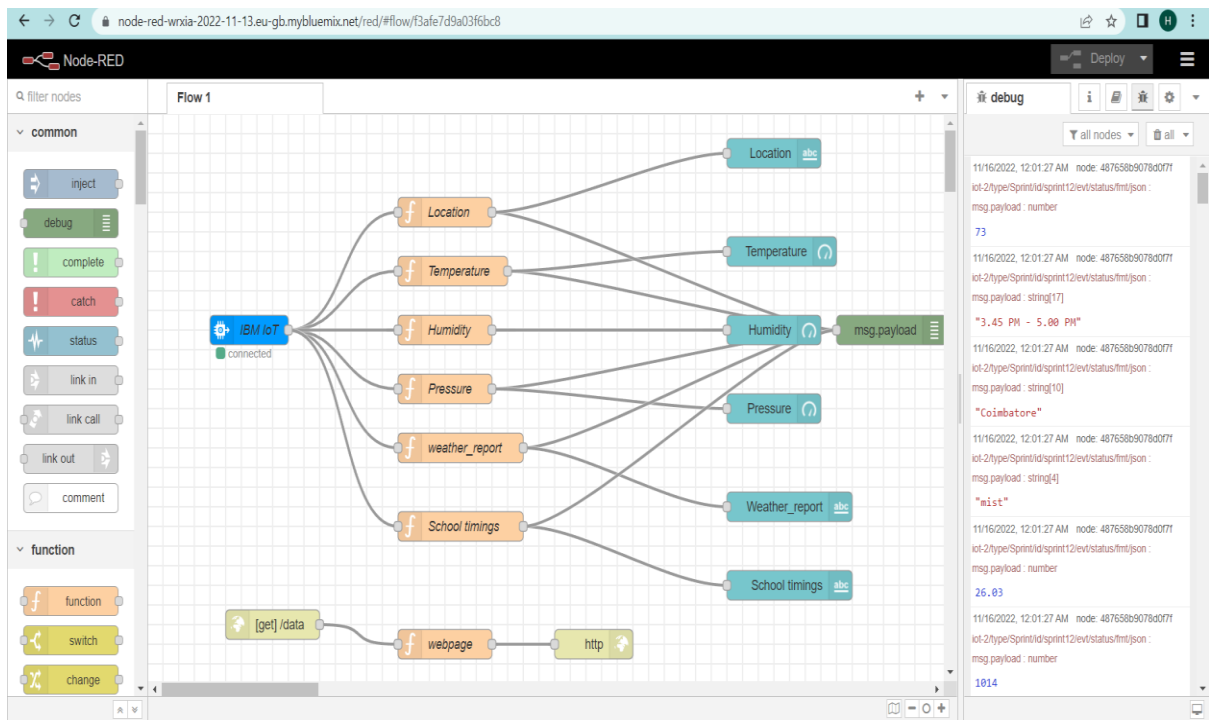
Time Received Nov 15, 2022 11:02 PM

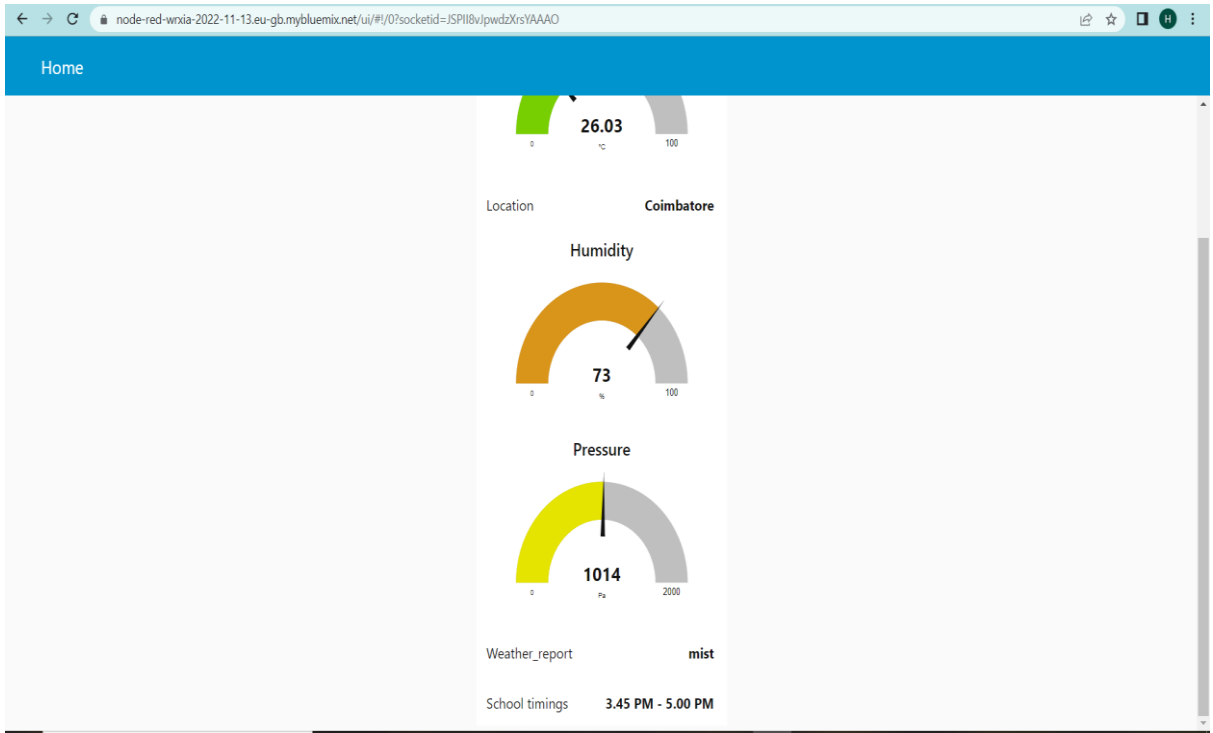
```
1 {  
2   "location": "Coimbatore",  
3   "temperature": 26.83,  
4   "humidity": 73,  
5   "pressure": 1014,  
6   "weather_report": "mist",  
7   "SchoolTiming": "3.45 PM - 5.00 PM"  
8 }
```

> ☐ weather\_tdy ☐ Disconnected weather Device Nov 15 11:02 PM 1 Simulation running

Items per page 50 1-3 of 3 items

## WEB APPLICATION USING NODE-RED SERVICE:





node-red-wrxia-2022-11-13.eu-gb.mybluemix.net/data

```
{ "location": "Coimbatore", "temperature": 26.03, "humidity": 73, "pressure": 1014, "weather_report": "mist", "Schooltiming": "3.45 PM - 5.00 PM" }
```

## **HARDWARE USED:**

### **1. ESP32:**

ESP32 is a microcontroller with built-in Bluetooth and Wi-Fi. It is used to get data from the sensors and send control signals to the LCD display. Also used to send the data to the cloud.

### **2. ULTRASONIC SENSOR:**

Ultrasonic sensor is used to detect the presence of an object over a distance. So here it is used to find the traffic density by detecting the presence of a number of vehicles.

### **3. DHT11:**

The **DHT11** is a commonly used **Temperature and humidity sensor** that comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data.

### **4. I2C:**

The I2C bus is a very popular and powerful bus used for communication between a master (or multiple masters) and a single or multiple slave device. Here it is used to simplify 16 pins of LCD display into 4 pins.

### **5. LCD DISPLAY:**

A 16x2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in 5x7 pixel matrix. The 16 x 2 intelligent alphanumeric dot matrix display is capable of displaying 224 different characters and symbols. This LCD has two registers, namely, Command and Data.



## Output:

