

**Project Development Phase**  
**Sprint III**

Team ID	PNT2022TMID30708
Project Name	Signs with Smart Connectivity for better road safety

**Sprint Targets:**

Sprint	Functional Requirement (Epic)	User Story Number	User Story /Task	Story Points	Priority	Team Members
Sprint-3	Login	USN-5	I need a website account because I'm the administrator.	7	Low	Janani.S Preethi.S Mohanapriya.P Elavarasi.G
Sprint-3	Dashboard	USN-6SSS	I should be able to watch over and add sign nodes as an administrator.	13	Medium	Janani.S Preethi.S Mohanapriya.P Elavarasi.G

## WOWKI Simulation:

The screenshot displays the WOKWI web-based IoT simulation platform. The interface is divided into two main sections: a code editor on the left and a simulation window on the right.

**Code Editor (Left):** The code is written in C++ for an ESP32 microcontroller. It includes libraries for WiFi, MQTT, and DHT11. The code defines a DHT22 sensor on pin 5 and sets up an MQTT client to connect to the IBM Watson IoT Platform. The MQTT client is configured with the organization 'psh4py', device type 'alert-device', device ID '4571', and token '12345678'. The code sets up a callback function to handle incoming MQTT messages and publishes temperature and humidity data to the topic 'iot-2/evt/Data/fmt/json'.

```
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* subscribetopic, byte* payload, unsigned int payloadLength);
10
11 //-----credentials of IBM Accounts-----
12
13 #define ORG "psh4py" // IBM ORGANIZATION 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 subscribetopic[] = "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
31 WiFiClient wificlient; // creating the instance for wificlient
32 PubSubClient client(server, 1883, callback, wificlient); // calling the predefined client
33
34
```

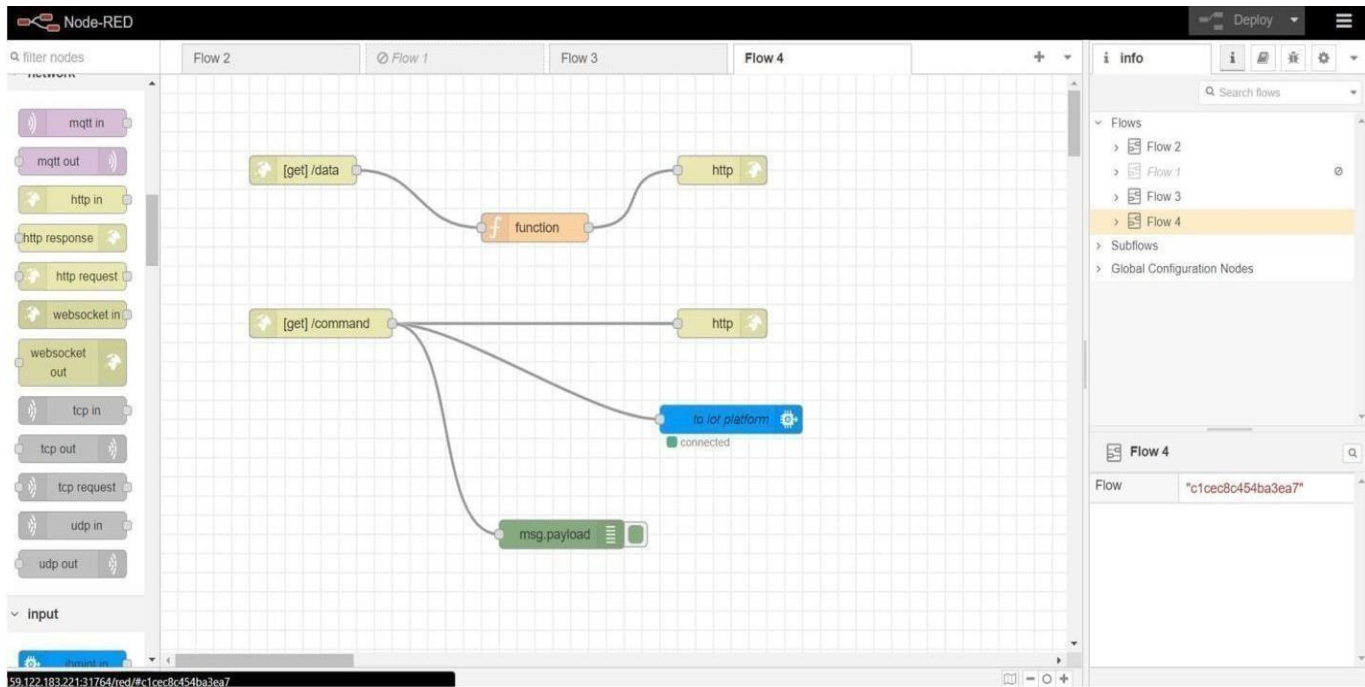
**Simulation Window (Right):** The simulation window shows a 3D model of an ESP32 microcontroller connected to a DHT22 digital temperature and humidity sensor. The sensor is connected to the ESP32 via I2C or 1-Wire. The simulation is running, and the output console shows the following data:

```
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
.....
```

The bottom status bar of the WOKWI interface shows the current temperature as 23°C Cloudy, the time as 08:23, and the date as 13-11-2022.

## IoT Device – IoT Platform





Edit function node

Delete

Cancel

Done

⚙️ Properties

⚙️

📄

🖨️

📁 Name

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")

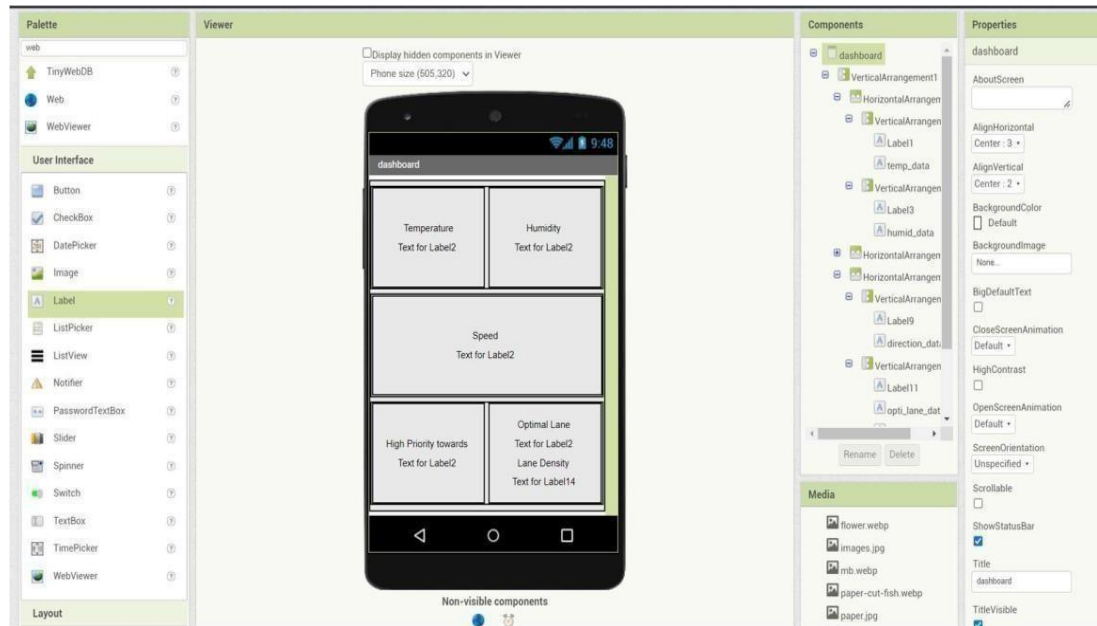
15

16 ^ };

17

18 return msg;

MIT App Inventor UI design:



## MIT App Inventor Backend design

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

```
when Web1.GetText
  uri responseCode responseType responseContent
do
  set temp_data.Text to look up in pairs key "temp"
  pairs call Web1.JsonTextDecode
  jsonText get responseContent
  notFound

  set humid_data.Text to look up in pairs key "humid"
  pairs call Web1.JsonTextDecode
  jsonText get responseContent
  notFound

  set speed_data.Text to look up in pairs key "speed"
  pairs call Web1.JsonTextDecode
  jsonText get responseContent
  notFound

  set direction_data.Text to look up in pairs key "res"
  pairs call Web1.JsonTextDecode
  jsonText get responseContent
  notFound

  set opt_lane_data.Text to look up in pairs key "optimal_lane"
  pairs call Web1.JsonTextDecode
  jsonText get responseContent
  notFound

  set lane_data.Text to join
  look up in pairs key "1"
  pairs call Web1.JsonTextDecode
  jsonText get responseContent
  notFound

  look up in pairs key "2"
  pairs call Web1.JsonTextDecode
  jsonText get responseContent
  notFound

  look up in pairs key "3"
  pairs call Web1.JsonTextDecode
  jsonText get responseContent
  notFound

  look up in pairs key "4"
  pairs call Web1.JsonTextDecode
  jsonText get responseContent
  notFound
```

## Sprint 3 delivery:

(OUTPUT) Display from MIT App:

