

**Project Development Phase**  
**Sprint-3: MIT App Design and Testing**

Date	16/11/2022
Team ID	PNT2022TMID14966
Project Name	Signs with Smart Connectivity for BetterRoad Safety
Maximum Marks	8 Marks

## Wokwi Simulation:

The screenshot shows the Wokwi simulation environment. On the left is the code editor with the file `sketch.ino` containing C++ code for an ESP32. The code includes includes for WiFi, PubSubClient, and DHT libraries, defines pins for DHT22 and relay modules, sets up credentials for IBM Watson IoT Platform, and configures the WiFi client. The main area shows a schematic diagram of the hardware setup: an ESP32 microcontroller connected to a DHT22 temperature/humidity sensor and an 8-channel relay module. The relay module is controlled by digital pins 2 through 9. The simulation window also displays a live serial monitor output showing the published JSON payload: `{"temp":37.40,"humidity":86.00,"North":0,"South":0,"East":0,"West":0}`.

```
sketch.ino
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 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
21 //----- Customise the above values -----
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[] = ":" 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
35 void setup() // configuring the ESP32
```

Simulation  
00:49.005 93%  
ON  
1 2 3 4 5 6 7 8  
ESP32  
DHT22  
{"temp":37.40,"humidity":86.00,"North":0,"South":0,"East":0,"West":0}  
Publish ok  
temp:37.40  
humidity:86.00  
Sending payload:  
{"temp":37.40,"humidity":86.00,"North":0,"South":0,"East":0,"West":0}

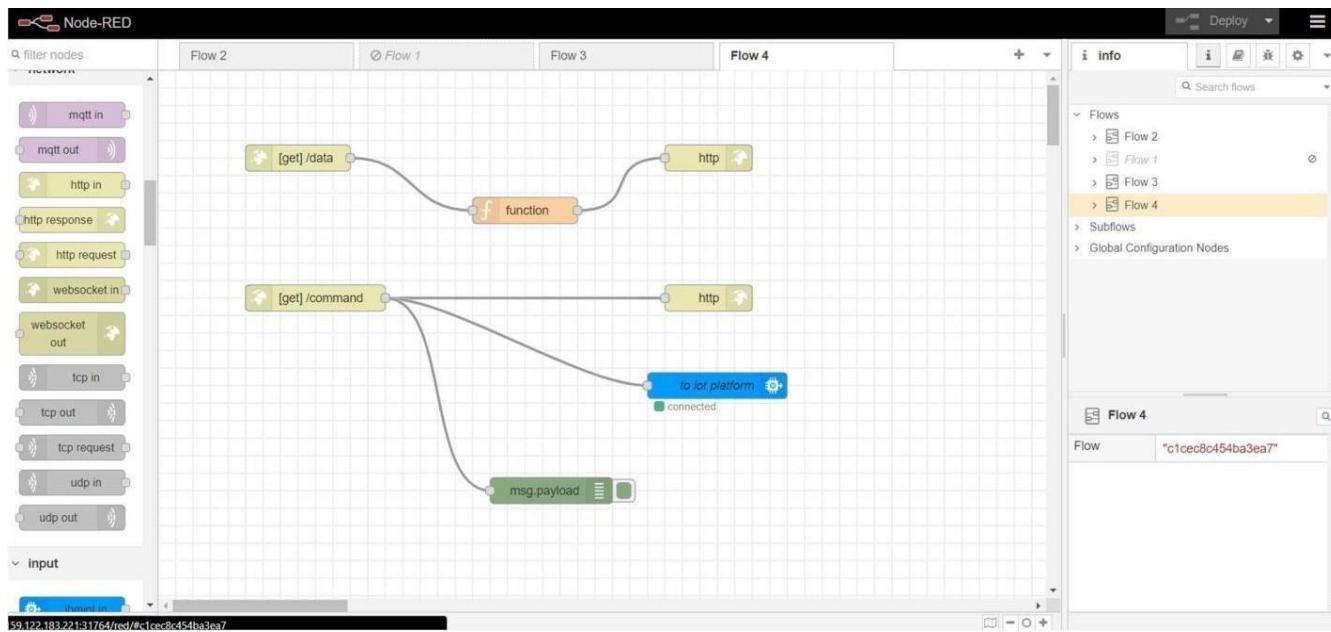
## IoT Device – IoT Platform

The screenshot shows the IoT Platform interface with a device detail view for 'edge-device-1'. The top navigation bar includes 'Browse', 'Action', 'Device Types', 'Interfaces', and an 'Add Device' button. The left sidebar features icons for device management, monitoring, and connectivity. The main content area displays the device's identity ('0001'), status ('Disconnected'), device type ('edge-device-1'), class ID ('Device'), and date added ('Nov 5, 2022 8:56 PM'). Below this, a 'Recent Events' section is shown, with the following data:

Event	Value	Format	Last Received
rnd_number	{"Lane_1":5,"Lane_2":83,"Lane_3":30,"Lane_4":...}	json	a few seconds ago
rnd_number	{"Lane_1":59,"Lane_2":59,"Lane_3":94,"Lane_4":...}	json	a few seconds ago
rnd_number	{"Lane_1":93,"Lane_2":88,"Lane_3":49,"Lane_4":...}	json	a few seconds ago
rnd_number	{"Lane_1":2,"Lane_2":61,"Lane_3":21,"Lane_4":...}	json	a few seconds ago
rnd_number	{"Lane_1":70,"Lane_2":11,"Lane_3":69,"Lane_4":...}	json	a few seconds ago

A message at the bottom indicates '1 Simulation running'.

## Node Red – Connect with MIT AppInventor



## Edit function node

Delete

Cancel

Properties



'g• Name      Name



Setup

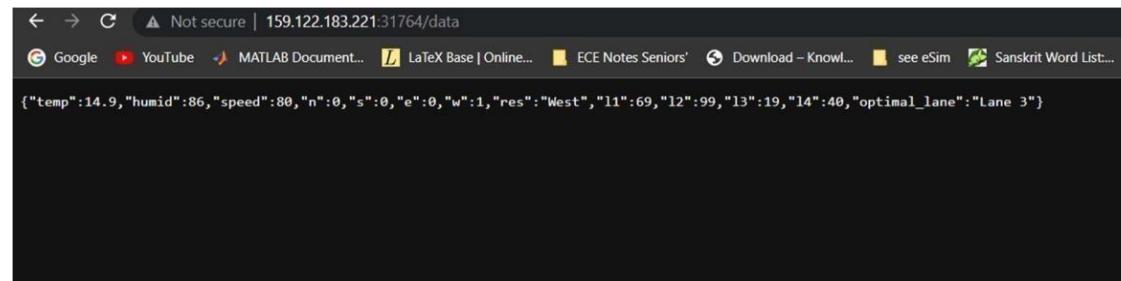
On Start

On Message

On Stop

```
• msg.payload = {  
2      "temp":global.get("temp"),  
3      "humid":global.get("humid"),  
p      "speed":global.get("speed"),  
s      "n":global.get("n"),  
6      "s":global.get("s"),  
7      "e":global.get("e"),  
8      "w":global.get("w"),  
g      "res":global.get("res"),  
16     "11":global.get("11"),  
tt     "12":global.get("12"),  
12     "13":global.get("13"),  
13     "14":global.get("14"),  
t4     "optimal lane":global.get("optional Jane")  
15  
16};  
17  
ig  return msg;
```

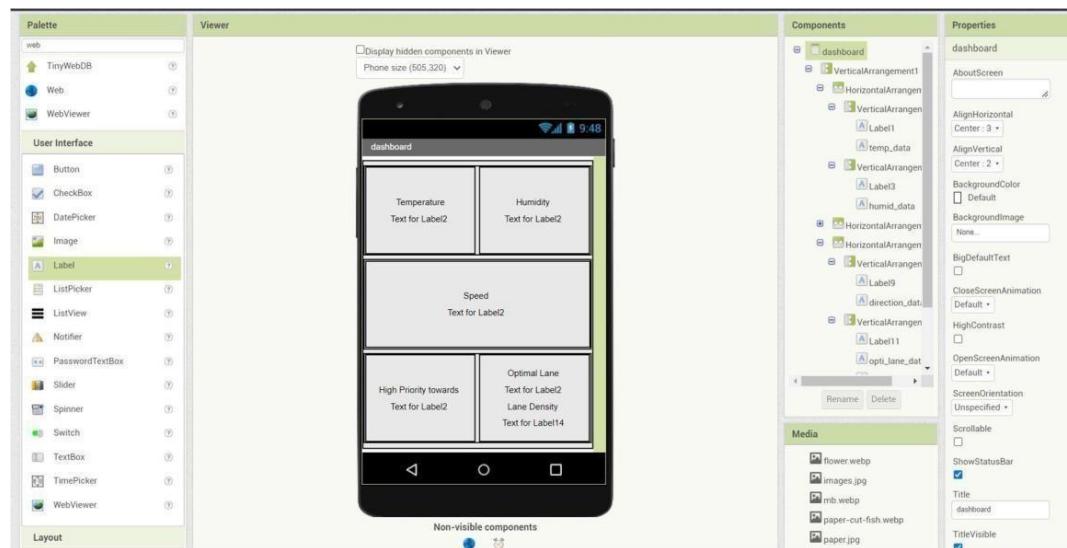
## Output from Node red:



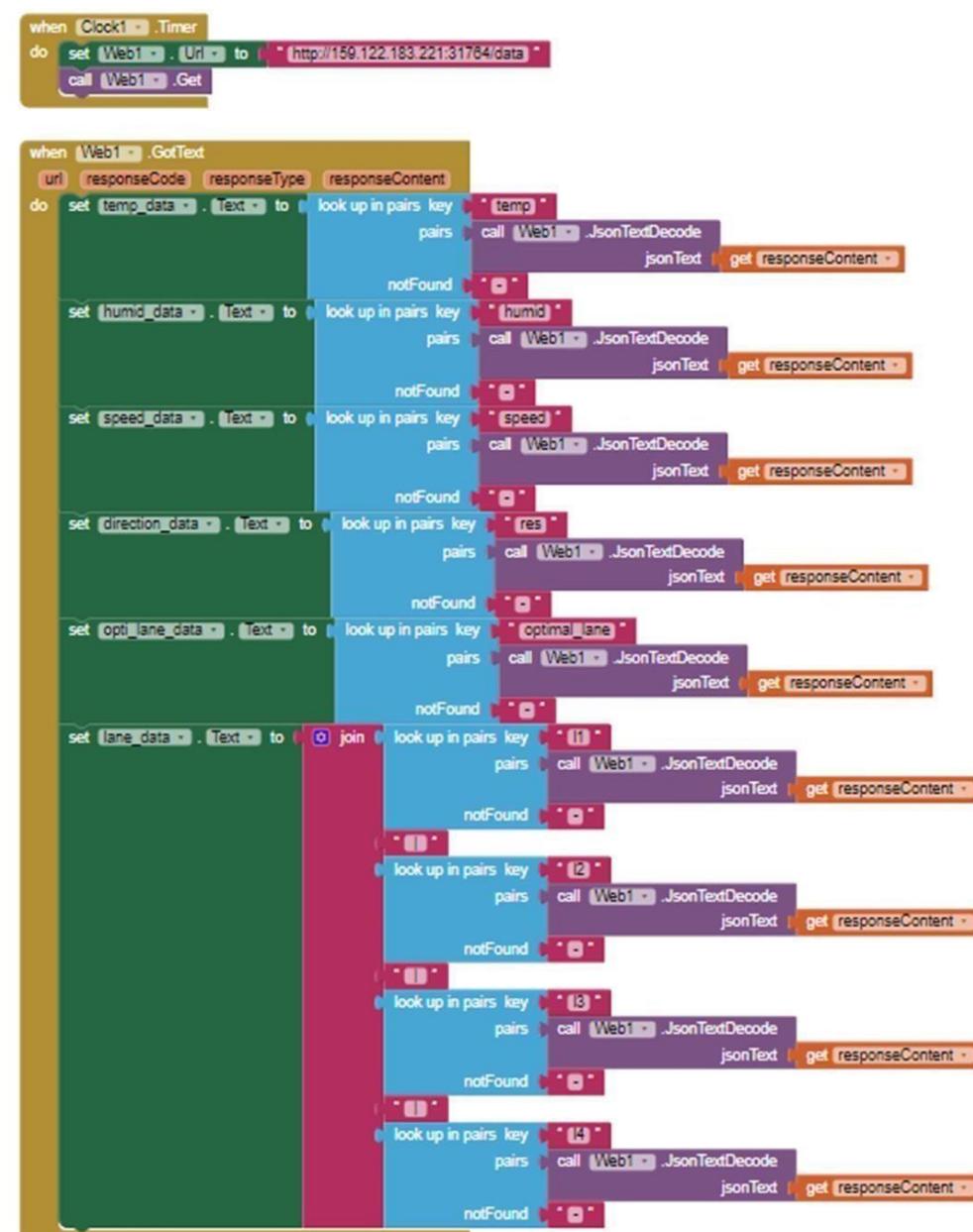
A screenshot of a web browser window. The address bar shows the URL `159.122.183.221:31764/data`. The page content is a JSON object:

```
{"temp":14.9,"humid":86,"speed":80,"n":0,"s":0,"e":0,"w":1,"res":"West","l1":69,"l2":99,"l3":19,"l4":40,"optimal_lane":"Lane 3"}
```

## MIT App Inventor UI design:



## MIT App Inventor Backend design:



**Sprint 3 delivery:**

**(OUTPUT) Display from MIT App:**

