

### Project Development PhaseSprint III

Date	11 November 2022
Team ID	PNT2022TMID12305
Project Name	Signs with Smart Connectivity for better road safety

#### Sprint Targets :

Sprint	Functional Requirement (Epic)	UserStory Number	UserStory/Task	Story Points	Priority	Team Members
Sprint-3	Login	USN-5	As an administrator , I should have an account of the website	7	Low	Ghavanamani Naveen Maheshwaran Hariharan
Sprint-3	Dashboard	USN-6SSS	As an admin , I should be able to monitor and add sign nodes	13	Medium	Ghavanamani Naveen Maheshwaran Hariharan

Wokwi Simulation: <https://wokwi.com/projects/348178332935782994>

The screenshot displays the Wokwi web-based IDE interface. The left pane shows the 'sketch.ino' file with the following code:

```
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[] = "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
```

The right pane shows the 'Simulation' view with a visual representation of an ESP32 microcontroller connected to a DHT22 temperature and humidity sensor. Below the simulation, the serial monitor displays the following output:

```
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 of the image shows a Windows taskbar with various application icons and a system tray indicating the date and time as 08:23 on 13-11-2022.

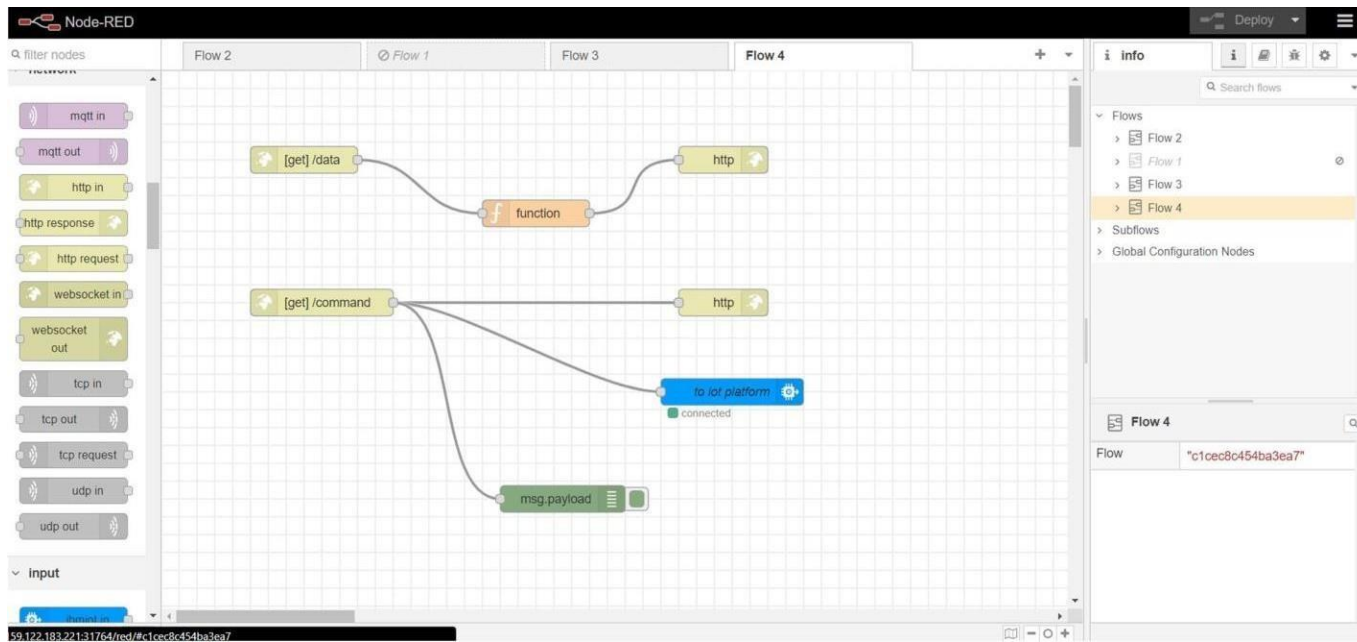
## IoT Device – IoT Platform

The screenshot displays the 'Recent Events' tab for a device named 'edge-device-1' (ID: 0001). The interface includes a top navigation bar with 'Browse', 'Action', 'Device Types', and 'Interfaces', and an 'Add Device' button. A left sidebar contains various icons for navigation. The main content area shows a table of recent events, with a note stating: 'The recent events listed show the live stream of data that is coming and going from this device.'

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

At the bottom right, a status box indicates '1 Simulation running'.

## Node Red – Connect with MIT AppInventor



## Edit function node

Delete

Cancel

Done

### Properties

Name

Name

Setup

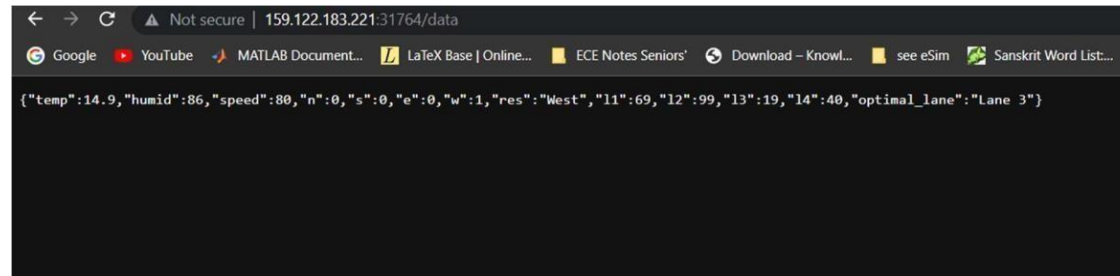
On Start

On Message

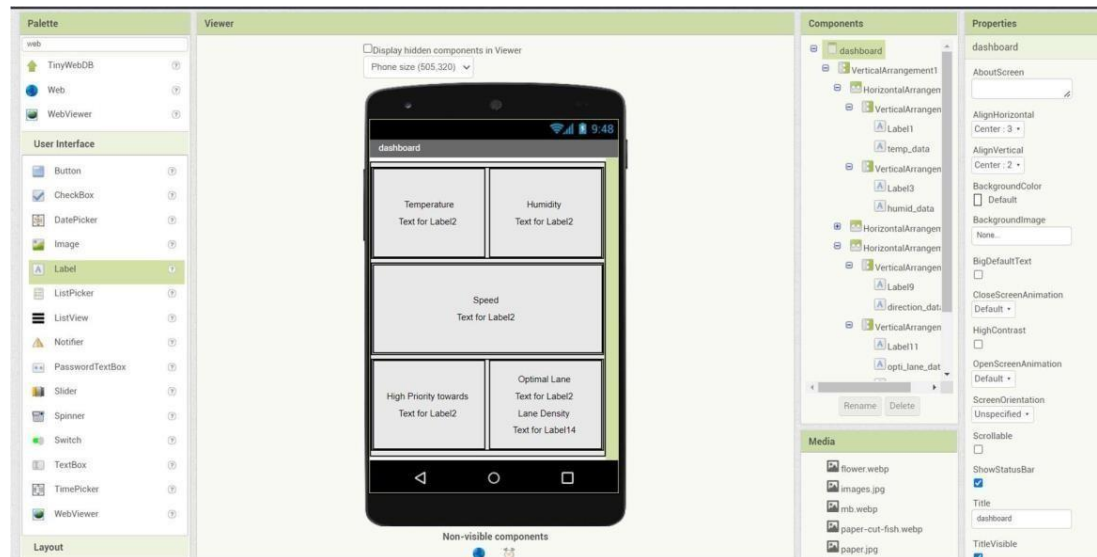
On Stop

```
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;
```

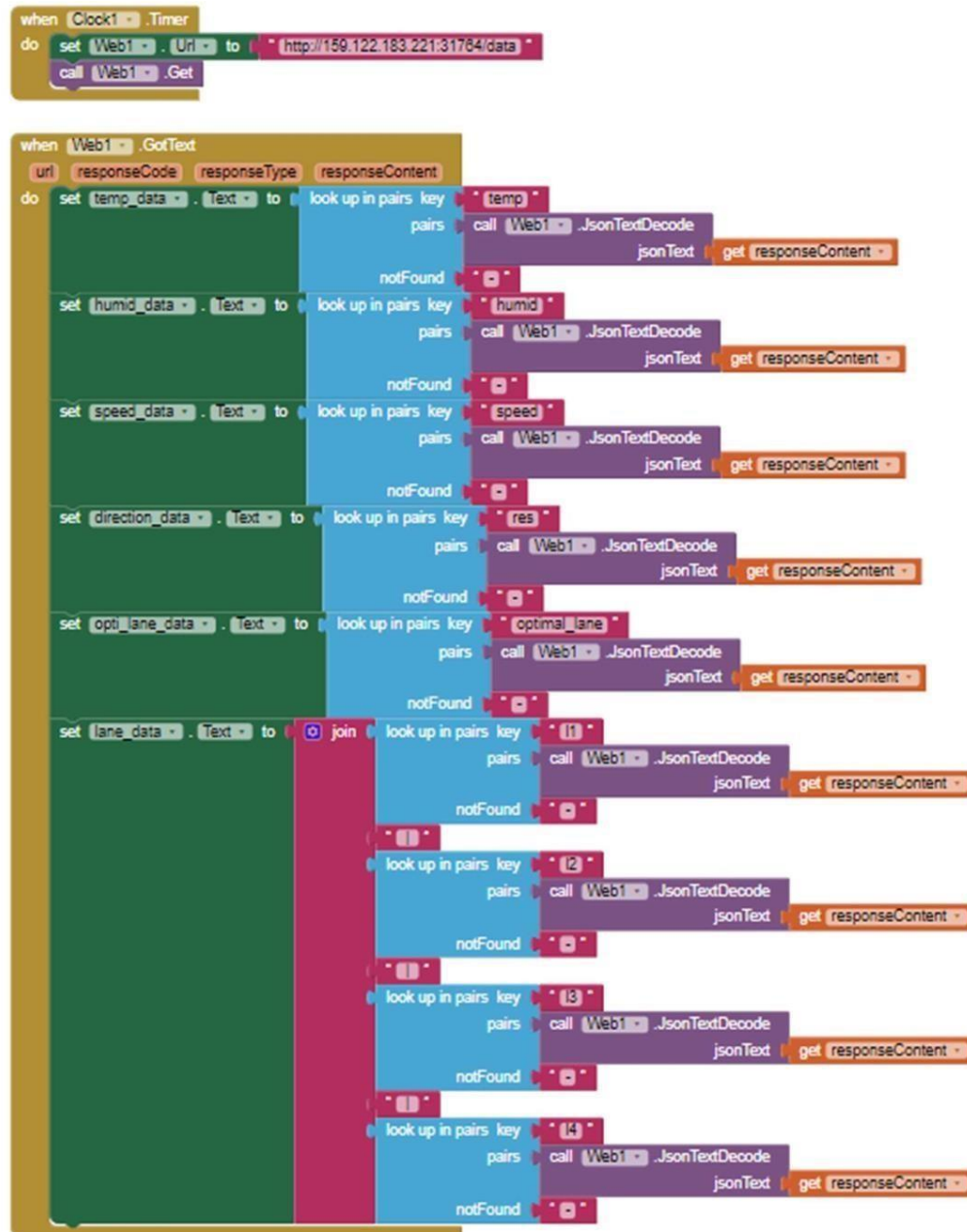
## Output from Node red:



## MIT App Inventor UI design:



## MIT App Inventor Backend design:



**Sprint 3 delivery:**

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

