

Project Development

Phase Sprint - 3

Team id	PNT2022TMID33212
Project name	Signs with smart connectivity for better road safety

Wokwi Simulation Link: <https://wokwi.com/projects/348599350304178770>

The screenshot displays the Wokwi simulation interface. On the left, the Arduino IDE editor shows 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 connected
8
9 void callback(char* topic, byte* payload, unsigned int payloadLength);
10
11 //-----credentials of IBM Accounts-----
12
13 #define ORG "03e16" // IBM ORGANIZATION ID
14 #define DEVICE_TYPE "HAYAN24" // Device type mentioned in ibm watson IOT Platform
15 #define DEVICE_ID "HAYAN24id" // Device ID mentioned in ibm watson IOT Platform
16 #define TOKEN "hayanthika2411" // Token
17 String data;
18 float h, t;
19
20 //----- Customise the above values -----
21 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
22 char publishTopic[] = "iot-2/evr/Data/fmt/json"; // topic name and type of event perform and format in which
23 char subscribeTopic[] = "iot-2/cmd/command/fmt/string"; // cmd REPRESENT command type AND COMMAND IS TEST OF
24 char authMethod[] = "use-token-auth"; // authentication method
25 char token[] = TOKEN;
26 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; // client id
27
28 //-----
29
30 WiFiClient wificlient; // creating the instance for wificlient
31 PubSubClient client(server, 1883, callback, wificlient); // calling the predefined client id by passing param
32
33 void setup() // configuring the ESP32
34 {
35   Serial.begin(115200);
36   dht.begin();
37   pinMode(A3, OUTPUT); // /North
38 }
```

On the right, the simulation window shows a circuit diagram with an ESP32 microcontroller connected to a DHT22 sensor. The sensor's VCC pin is connected to the ESP32's 5V pin, and its GND pin is connected to the ESP32's GND pin. The sensor's data pin is connected to the ESP32's D0 pin. The simulation output shows the following sequence of events:

```
humidity:86.00
Sending payload: {"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}
Publish ok
```

IOT-DEVICE :

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for navigation. The main content area shows details for a device named 'HAYAN24id', which is 'Connected'. The 'Recent Events' tab is active, displaying a table of events.

Event	Value	Format	Last Received
Data	{\"temp\":37.4,\"humidity\":86,\"North\":0,\"South\":0,...}	json	a few seconds ago
Data	{\"temp\":37.4,\"humidity\":86,\"North\":0,\"South\":0,...}	json	a few seconds ago
Data	{\"temp\":37.4,\"humidity\":86,\"North\":0,\"South\":0,...}	json	a few seconds ago
Data	{\"temp\":37.4,\"humidity\":86,\"North\":0,\"South\":0,...}	json	a few seconds ago
Data	{\"temp\":37.4,\"humidity\":86,\"North\":0,\"South\":0,...}	json	a few seconds ago

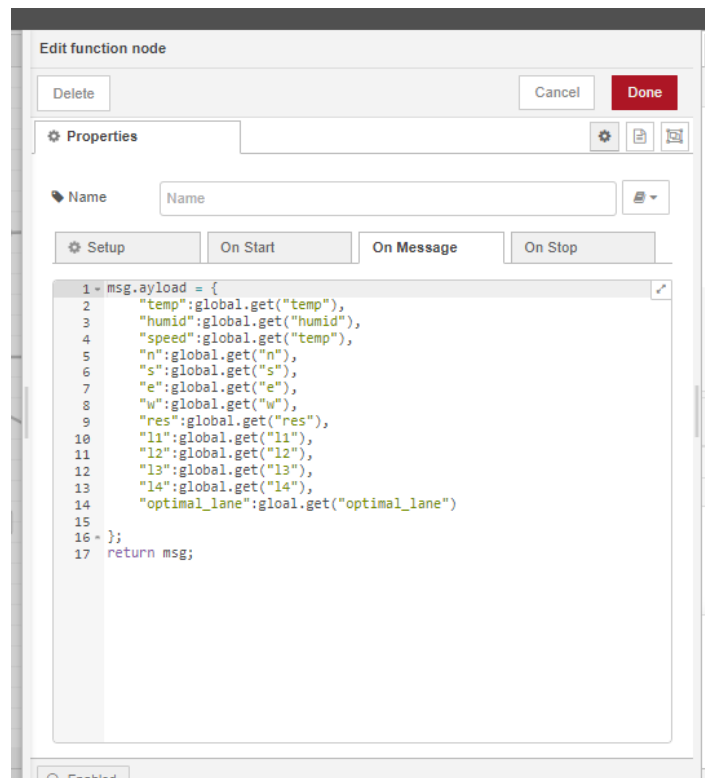
0 Simulations running

NODE-RED CONNECT WITH MIT APP INVENTOR:

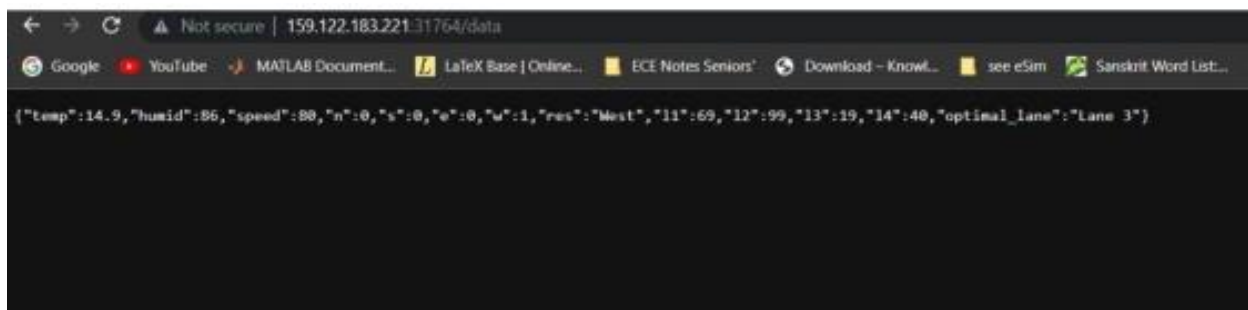
The screenshot shows the Node-RED interface with a flow named 'Flow 1'. The flow consists of the following nodes:

- [get] /data**: A GET endpoint node.
- function**: A function node that processes the data.
- http**: An HTTP response node.
- [get] /command**: A GET endpoint node for commands.
- http**: An HTTP response node for commands.
- IBM IoT**: A node representing the IBM IoT device, which is 'connected'.
- msg.payload**: A message payload node.

The right sidebar shows the 'info' tab, displaying details for the selected 'http' node, including its ID and type.



OUTPUT:



MIT APP INVENTOR UI DESIGN:



```
when Clock.Timer
do
  set Web.Url to "http://59.22.15.22/0-1764/data"
  call Web.Get
```

```
when Web.GetTest
  uri responseCode responseType responseContent
do
  set temp_data.Text to
    look up in pairs key temp
    pairs call Web.JsonTextDecode
    jsonText get responseContent
    notFound
  set humid_data.Text to
    look up in pairs key humid
    pairs call Web.JsonTextDecode
    jsonText get responseContent
    notFound
  set speed_data.Text to
    look up in pairs key speed
    pairs call Web.JsonTextDecode
    jsonText get responseContent
    notFound
  set direction_data.Text to
    look up in pairs key dir
    pairs call Web.JsonTextDecode
    jsonText get responseContent
    notFound
  set opt_lane_data.Text to
    look up in pairs key opt_lane
    pairs call Web.JsonTextDecode
    jsonText get responseContent
    notFound
  set lane_data.Text to
    join
    look up in pairs key E1
    pairs call Web.JsonTextDecode
    jsonText get responseContent
    notFound
    E2
    look up in pairs key E2
    pairs call Web.JsonTextDecode
    jsonText get responseContent
    notFound
    E3
    look up in pairs key E3
    pairs call Web.JsonTextDecode
    jsonText get responseContent
    notFound
    E4
    look up in pairs key E4
    pairs call Web.JsonTextDecode
    jsonText get responseContent
    notFound
```

OUTPUT DISPLAY FROM MIT APP:

