

Project Development Phase Sprint 3

Signs with smart connectivity for Better road safety

Team ID	PNT2022TMID17080
Project Name	Project - Signs with smart connectivity for Better road safety
Marks	8 Marks

WOWKI LINK : <https://wokwi.com/projects/348426120386839123>

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQTT
#include "DHT.h"// Library for dht11
#define DHTPIN 5 // what pin we're connected to
#define DHTTYPE DHT22 // define type of sensor DHT 11
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "b59mry"//IBM ORGANITION ID
#define DEVICE_TYPE "Event_1"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1111"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678" //Token
String data3;
float h, t;
```

```

//----- Customise the above values -----
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT command type AND COMMAND IS TEST
OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
//-----
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by passing parameter like server
id,portand wificredential
void setup()// configuring the ESP32
{
  Serial.begin(115200);
  dht.begin();
  pinMode(33, INPUT); //North
  pinMode(25, INPUT); // South
  pinMode(26, INPUT); // East
  pinMode(27, INPUT); // West
  delay(10);
  Serial.println();
  wificonnect();
  mqttconnect();
}
int n, s, e, w;
void loop()// Recursive Function
{
  h = dht.readHumidity();
  t = dht.readTemperature();
  Serial.print("temp:");
  Serial.println(t);
}

```

```
Serial.print("humidity:");  
Serial.println(h);
```

```
n = digitalRead(33);  
s = digitalRead(25);  
e = digitalRead(26);  
w = digitalRead(27);
```

```
PublishData(t, h, n, s, e, w);  
delay(1000);  
if (!client.loop()) {  
    mqttconnect();  
}  
}  
/*.....retrieving to Cloud.....*/  
void PublishData(float temp, float humid, int n, int s, int e, int w) {  
    mqttconnect();//function call for connecting to ibm  
    /*  
        creating the String in in form JSon to update the data to ibm cloud  
    */  
    String payload = "{\"temp\":";  
    payload += temp;  
    payload += ", \"humidity\":";  
    payload += humid;  
    payload += ", \"North\":";  
    payload += n;  
    payload += ", \"South\":";  
    payload += s;  
    payload += ", \"East\":";  
    payload += e;  
    payload += ", \"West\":";  
    payload += w;
```

```

payload += "}";
Serial.print("Sending payload: ");
Serial.println(payload)
if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print publish ok in Serial monitor or else
it will print publish failed
} else {
    Serial.println("Publish failed");
}
}
void mqttconnect() {
    if (!client.connected()) {
        Serial.print("Reconnecting client to ");
        Serial.println(server);
        while (!!!client.connect(clientId, authMethod, token)) {
            Serial.print(".");
            delay(500);
        }
        initManagedDevice();
        Serial.println();
    }
}
void wificonnect() //function defination for wificonnect
{
    Serial.println();
    Serial.print("Connecting to ");

    WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the connection
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
}

```

```

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}
void initManagedDevice() {
    if (client.subscribe(subscribetopic)) {
        Serial.println(subscribetopic);
        Serial.println("subscribe to cmd OK");
    } else {
        Serial.println("subscribe to cmd FAILED");
    }
}
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{

    Serial.print("callback invoked for topic: ");
    Serial.println(subscribetopic);
    for (int i = 0; i < payloadLength; i++) {
        //Serial.print((char)payload[i]);
        data3 += (char)payload[i];
    }
    Serial.println("data: " + data3);
    DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht connected

}

```

WOWKI :

wokwi.com/projects/348426120386839123

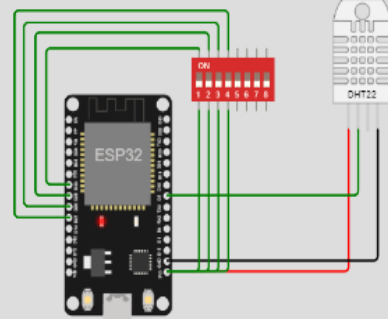
WOKWI SAVE SHARE IOT ROAD SAFETY Docs

sketch.ino diagram.json libraries.txt Library Manager

```
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
8
9 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
10
11 //-----credentials of IBM Accounts-----
12
13 #define ORG "b59mry" //IBM ORGANIZATION ID
14 #define DEVICE_TYPE "Event_1" //Device type mentioned in ibm watson IOT Platform
15 #define DEVICE_ID "1111" //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
24 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command
25 char authMethod[] = "use-token-auth"; // authentication method
26 char token[] = TOKEN;
```

Simulation

00:03.833 61%



Connecting to ..
WiFi connected
IP address:
10.10.0.2

sketch.ino • diagram.json libraries.txt Library Manager

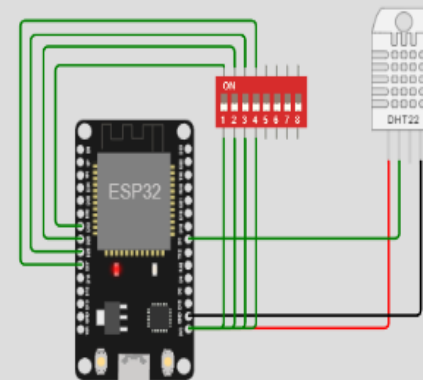
```

33
34
35 void setup()// configuring the ESP32
36 {
37     Serial.begin(115200);
38     dht.begin();
39     pinMode(33, INPUT); //North
40     pinMode(25, INPUT); // South
41     pinMode(26, INPUT); // East
42     pinMode(27, INPUT); // West
43     delay(10);
44     Serial.println();
45     wificonnect();
46     mqttconnect();
47 }
48
49 int n, s, e, w;
50
51 void loop()// Recursive Function
52 {
53
54     h = dht.readHumidity();
55     t = dht.readTemperature();
56     Serial.print("temp:");
57     Serial.println(t);
58     Serial.print("humidity:");
59     Serial.println(h);
60

```

Simulation

00:17.099 65%



```

{"temp":3.00,"humidity":86.00,"North":0,"South":0,"East":0,"West":0}
Publish ok
temp:3.00
humidity:86.00
Sending payload:
{"temp":3.00,"humidity":86.00,"North":0,"South":0,"East":0,"West":0}
Publish ok

```

NODE RED:

The screenshot displays the Node-RED web interface in a browser. The address bar shows the URL: `node-red-obbsa-2022-11-10.eu-gb.mybluemix.net/red/#flow/fbbcf9207dc9f0d8`. The interface is divided into several sections:

- Left Panel (Node Palette):** Contains a search bar "filter nodes" and a list of nodes categorized by type. The "function" category is expanded, showing nodes like "function", "switch", "change", "range", "template", "delay", and "trigger".
- Flow Editor:** The central workspace where flows are built. It shows three tabs: "Flow 1", "Flow 3", and "Flow 2". The "Flow 2" tab is active, displaying a flow diagram with the following components:
 - An "INPUT DATA" node connected to a "function" node.
 - The "function" node connected to an "http" node.
 - A "[get] /source" node connected to three destinations: an "http" node, an "IBM IoT" node (labeled "connected"), and a "msg payload" node.
- Right Panel (Dashboard Sidebar):** Contains a "dashboard" tab and a "Site" configuration section. The "Title" is "Node-RED Dashboard". The "Options" section includes several dropdown menus: "Show the title bar", "Click to show side menu", "No swipe between tabs", and "Node-RED theme everywhere". The "Date Format" is set to "DD/MM/YYYY". The "Sizes" section has a table for widget dimensions:

	Horizontal	Vertical
1x1 Widget Size	48	48
Widget Spacing	6	6
Group Padding	0	0
Group Spacing	6	6

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
```