

## SPRINT 4

Date	14 November 2022
Team ID	PNT2022TMID33005
Project Name	Hazardous Area Monitoring for Industrial Plant powered by IoT

### WOKWI CODE:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQTT
#include "DHT.h"// Library for dht11
#define DHTPIN 15 // what pin we're connected to
#define DHTTYPE DHT22 // define type of sensor DHT 11
#define LED 2

DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht
connected

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-----credentials of IBM Accounts-----

#define ORG "iagqzu"//IBM ORGANITION ID
#define DEVICE_TYPE "Deepak"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "123"//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()// configureing the ESP32
{
  Serial.begin(115200);
```

```

dht.begin();
pinMode(LED,OUTPUT);
delay(10);
Serial.println();
wificonnect();
mqttconnect();
}

void loop()// Recursive Function
{

    h = dht.readHumidity();
    t = dht.readTemperature();
    Serial.print("temp:");
    Serial.println(t);
    Serial.print("Humid:");
    Serial.println(h);

    PublishData(t, h);
    delay(1000);
    if (!client.loop()) {
        mqttconnect();
    }
}

/*.....retrieving to
Cloud. .... */

void PublishData(float temp, float humid) {
    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 += "," " \"Humid\":";
    payload += humid;
    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);
if(data3=="lighton")
{
Serial.println(data3);
digitalWrite(LED,HIGH);
}
else
{
Serial.println(data3);
digitalWrite(LED,LOW);
}
data3="";
}

```

## WOKWI OUTPUT:

The screenshot displays the WOKWI IDE interface. On the left, the 'sketch.ino' file contains 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 15 // what pin we're connected to
5 #define DHTTYPE DHT22 // define type of sensor DHT 11
6 #define LED 2
7
8 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of dht connect
9
10 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
11
12 //-----credentials of IBM Accounts-----
13
14 #define ORG "i3869j" //IBM ORGANIZATION ID
15 #define DEVICE_TYPE "abcd" //Device type mentioned in ibm watson IOT Platform
16 #define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
17 #define TOKEN "12345678" //Token
18 String data3;
19 float h, t;
20
21
22 //----- Customise the above values -----
23 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
24 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform a
25 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND
26 char authMethod[] = "use-token-auth"; // authentication method
27 char token[] = TOKEN;
28 char clientId[] = "dt:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
29
30
31 //-----
32 WiFiClient wificlient; // creating the instance for wificlient
33 PubSubClient client(server, 1883, callback, wificlient); //calling the predefined client

```

On the right, the 'Simulation' window shows a visual representation of the hardware: an ESP32 microcontroller, a red LED, a resistor, and a DHT22 temperature and humidity sensor. Below the simulation, the serial output log shows the following sequence of events:

```

Humid:40.00
Sending payload: {"temp":24.00,"Humid":40.00}
Publish ok
temp:24.00
Humid:40.00
Sending payload: {"temp":24.00,"Humid":40.00}
Publish ok

```

## IBM WATSON PLATFORM →

### DEVICE EVENT LOG:

The screenshot displays the IBM Watson IoT Platform interface. At the top, the header shows the platform name, a user profile icon, and the email address 820419106011@smartinternz.com with the ID: lagqzu. Below the header, a navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for navigation. The main content area shows a search bar for 'Search by Device ID' and a 'Device Simulator' toggle. Below this is a table listing devices. The selected device has ID 123, is 'Connected', and is of type 'Deepak'. The 'Recent Events' tab is active, showing a list of events. The events table has columns for 'Event', 'Value', 'Format', and 'Last Received'.

Event	Value	Format	Last Received
IoTSensor	{"temp":32,"Humid":33}	json	a few seconds ago
IoTSensor	{"temp":58,"Humid":87}	json	a few seconds ago
IoTSensor	{"temp":35,"Humid":6}	json	a few seconds ago
IoTSensor	{"temp":86,"Humid":24}	json	a few seconds ago

### DEVICE EVENT PAYLOAD:

The screenshot displays the IBM Watson IoT Platform interface with an 'Event Payload' modal window open. The modal shows the 'Event Name' as 'IoTSensor' and the 'Time Received' as 'Nov 14, 2022 11:31 AM'. The payload is shown in a code editor with line numbers 1 through 4. The payload is a JSON object: {"temp": 32, "Humid": 33}.

```
1 | {  
2 |   "temp": 32,  
3 |   "Humid": 33  
4 | }
```

## DEVICE- BOARD:

