

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[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
29
30 //-----
31 WiFiClient wifiClient; // creating the instance for wifiClient
32 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client
33

```

On the right, the 'Simulation' window shows a visual representation of the hardware: an ESP32 microcontroller board connected to a red LED 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. 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 a table of devices, with the first device (ID 123) selected. Below the table, the "Recent Events" tab is active, displaying a list of events for the selected device.

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
123	Connected	Deepak	Device	Sep 29, 2022 7:54 PM	

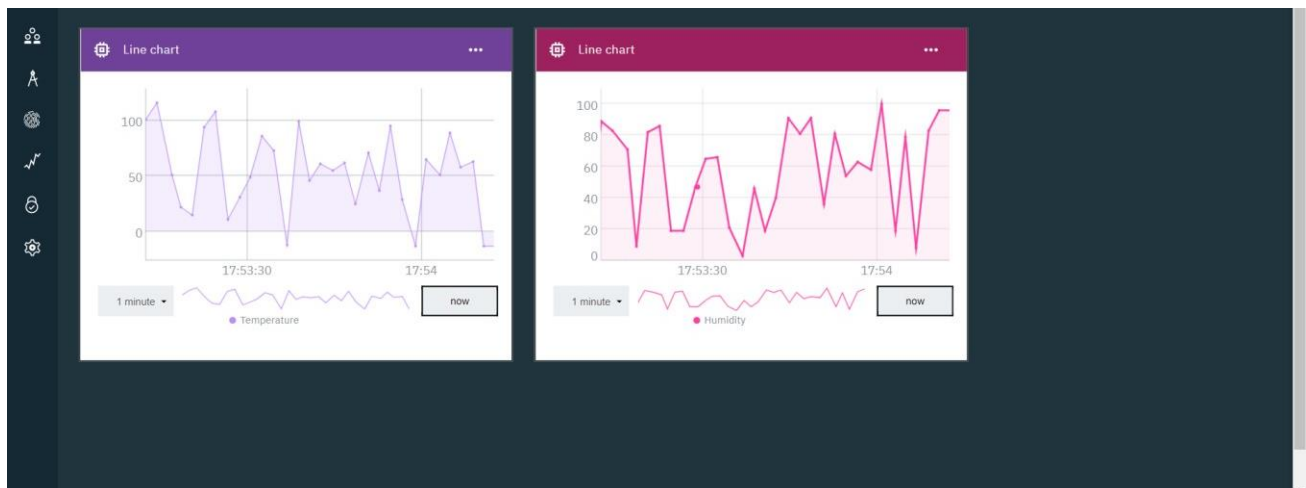
Event	Value	Format	Last Received
IoTSensor	{"temp":32,"Humid":33}	json	a few seconds ago
IoTSensor	{"temp":58,"Humid":87}	json	a few seconds ago
IoTSensor	{"temp":35,"Humid":6}	json	a few seconds ago
IoTSensor	{"temp":86,"Humid":24}	json	a few seconds ago

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 "IoTSensor" and the time received "Nov 14, 2022 11:31 AM". The event payload is displayed as a JSON object.

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

DEVICE- BOARD:



IBM CLOUDANT DB LOG:

The screenshot shows the IBM Cloudant DB interface. On the left is a sidebar with navigation options: 'All Documents', 'Query', 'Permissions', 'Changes', 'Design Documents', and 'library'. The main area displays a table of documents. The table has columns for '_id', 'humidity', and 'temperature'. The data is sorted by '_id'. At the bottom, there is a 'Showing 3 of 4 columns' message and a 'Showing document 1 - 20' message.

_id	humidity	temperature
0096ab1244940360661f0bce73051181	9	79
0096ab1244940360661f0bce730520d0	68	122
0096ab1244940360661f0bce730d0d19	6	109
0096ab1244940360661f0bce730d1a9b	72	39
0096ab1244940360661f0bce730d380a	44	105
0096ab1244940360661f0bce731556d4	37	12
0096ab1244940360661f0bce73156b2b	18	-5
0096ab1244940360661f0bce731b7048	81	5
0096ab1244940360661f0bce731bbf33	25	90
0096ab1244940360661f0bce7320722f	87	11
0096ab1244940360661f0bce73207f8c	48	49
0096ab1244940360661f0bce7325d136	56	107