

## Project Deveopment Phase

### Sprint 2

Date	07 November 2022
Team ID	PNT2022TMID30240
Project Name	Hazardous Area Monitoring for Industrial Plant powered by IoT

In this sprint, we are getting Temperature and Humidity as input from the ESP32 using wokwi which is considered to get the input from the environment condition.

#### Solution:

```
#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 22
#define LED 5
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 "6yafic"//IBM ORGANITION ID
#define DEVICE_TYPE "Sensor"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "Sensorid"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "VghKTVPaSlbz+vICyz" //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/test/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("Temperature:");
  Serial.println(t);
  Serial.print("Humidity:");
  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 = "{\"Temperature\":\"";
  payload += temp;
  payload += "," "\"Humidity\":\"";
  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 definition 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=="Alarmon")
{
    Serial.println(data3);
    digitalWrite(LED,HIGH);

}

else
{
    Serial.println(data3);
    digitalWrite(LED,LOW);

}

data3="";
}

```

## Data gathering from sensors (wokwi )

The screenshot displays the Wokwi online 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 22
6 #define LED 5
7 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of
8
9 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
10
11 //-----credentials of IBM Accounts-----
12
13 #define ORG "6yafic" //IBM ORGANITION ID
14 #define DEVICE_TYPE "Sensor" //Device type mentioned in ibm watson IOT Platform
15 #define DEVICE_ID "Sensorid" //Device ID mentioned in ibm watson IOT Platform
16 #define TOKEN "VghKTVPaS!bz+vlCyz" //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/test/fmt/String"; // cmd REPRESENT command
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

```

On the right, the 'Simulation' window shows a visual representation of the hardware: an ESP32 microcontroller board connected to a DHT22 digital temperature and humidity sensor. Below the simulation, a console log displays the following output:

```

Sending payload: {"Temperature":47.80,"Humidity":43.00}
Publish ok
Temperature:47.80
Humidity:43.00
Sending payload: {"Temperature":47.80,"Humidity":43.00}
Publish ok
Reconnecting client to 6yafic.messaging.internetofthings.ibmcloud.com

```

## Uploaded data in Cloud from sensors (wokwi - ESP32)

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes tabs for 'Service Details - IBM Cloud' and 'IBM Watson IoT Platform'. The browser address bar shows the URL: <https://6yafic.internetofthings.ibmcloud.com/dashboard/devices/browse>. The user is logged in as '2k19ece005@kiot.ac.in' with ID '6yafic'.

The main dashboard area is titled 'IBM Watson IoT Platform' and features a sidebar with navigation icons. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A blue 'Add Device' button is located on the right.

The 'Browse' tab is active, showing a list of devices. The first device is 'PNT2022TMID30240', which is 'Disconnected' and a 'Monitoringarea' device, last updated on 'Oct 30, 2022 2:17 PM'. The second device is 'Sensorid', which is 'Connected' and a 'Sensor' device, last updated on 'Nov 4, 2022 12:24 PM'. This device is selected, and its details are shown in a modal window.

The modal window for 'Sensorid' has tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, displaying a table of recent events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. The events are listed as 'Data' with a value of `{"Temperature":47.8,"Humidity":43}` in 'json' format, received 'a few seconds ago'.

Event	Value	Format	Last Received
Data	<code>{"Temperature":47.8,"Humidity":43}</code>	json	a few seconds ago
Data	<code>{"Temperature":47.8,"Humidity":43}</code>	json	a few seconds ago
Data	<code>{"Temperature":47.8,"Humidity":43}</code>	json	a few seconds ago
Data	<code>{"Temperature":47.8,"Humidity":43}</code>	json	a few seconds ago
Data	<code>{"Temperature":47.8,"Humidity":43}</code>	json	a few seconds ago