

Sprint 2

Team ID	PNT2022TMID15161
Project Name	Industry-specific Intelligent Fire Management System

Configuring IBM IoT Platform and sending data to IBM cloud

The IBM cloud details included in 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 #define BUZZER 4
8
9 #include <Wire.h>
10 #include <LiquidCrystal_I2C.h>
11
12 LiquidCrystal_I2C LCD = LiquidCrystal_I2C(0x27, 20, 4);
13
14 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin number and type
15
16 void callback(char* topic, byte* payload, unsigned int length) {
17
18 //-----credentials of IBM Accounts-----
19
20 #define ORG "pol6f4" //IBM ORGANIZATION ID
21 #define DEVICE_TYPE "ESP-32IoT" //Device type mentioned in IBM Cloud
22 #define DEVICE_ID "100100C40A24" //Device ID mentioned in IBM Cloud
23 #define TOKEN "EnuF+Tgx4Q@YwJmsq" //Token
24 String data;
25 float h, t, f;
26
27 //----- Customise the above values -----
28 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
29 char publishTopic[] = "iot-2/evt/data/fmt/json"; // topic name as per IBM Cloud
30 char subscribeTopic[] = "iot-2/cmd/command/fmt/string"; // cmd
31 char authMethod[] = "use-token-auth"; // authentication method
32 char token[] = TOKEN;
33 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //clientId
```

Fire Detection & Indication in LCD Screen

Sending Sensor Data to IBM Cloud

```
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 #define BUZZER 4
8
9 #include <Wire.h>
10 #include <LiquidCrystal_I2C.h>
11
12 LiquidCrystal_I2C LCD = LiquidCrystal_I2C(0x27, 20, 4);
13
14 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin number and type
15
16 void callback(char* topic, byte* payload, unsigned int length) {
17
18 //-----credentials of IBM Accounts-----
19
20 #define ORG "pol6f4" //IBM ORGANIZATION ID
21 #define DEVICE_TYPE "ESP-32IoT" //Device type mentioned in IBM Cloud
22 #define DEVICE_ID "100100C40A24" //Device ID mentioned in IBM Cloud
23 #define TOKEN "EnuF+Tgx4Q@YwJmsq" //Token
24 String data;
25 float h, t, f;
26
27 //----- Customise the above values -----
28 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
29 char publishTopic[] = "iot-2/evt/data/fmt/json"; // topic name as per IBM Cloud
30 char subscribeTopic[] = "iot-2/cmd/command/fmt/string"; // cmd
31 char authMethod[] = "use-token-auth"; // authentication method
32 char token[] = TOKEN;
33 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //clientId
```

Publishing Sensor Data & Receiving in IBM Cloud

The screenshot displays the IBM Watson IoT Platform interface. A device with ID 100100C40A24 is shown as 'Connected'. The 'Recent Events' tab is selected, showing a table of sensor data. A red box highlights the table, and a callout points to it with the text 'Sensor Data Received in IBM Cloud'.

Event	Value	Format	Last Received
Data	{"temp":12.3,"Humid":62,"Smoke":454}	json	a few seconds ago
Data	{"temp":12.3,"Humid":62,"Smoke":482}	json	a few seconds ago
Data	{"temp":12.3,"Humid":62,"Smoke":816}	json	a few seconds ago
Data	{"temp":12.3,"Humid":62,"Smoke":788}	json	a few seconds ago
Data	{"temp":12.3,"Humid":62,"Smoke":663}	json	a few seconds ago

Sprint 2 - Coding:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
#define DHTPIN 15
#define DHTTYPE DHT22
#define LED 2
#define BUZZER 4

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C LCD = LiquidCrystal_I2C(0x27, 20, 4);

DHT dht (DHTPIN, DHTTYPE);

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

#define ORG "pol6f4"
#define DEVICE_TYPE "ESP-32IoT"
#define DEVICE_ID "100100C40A24"
#define TOKEN "EnuF+Tgx40@Y!wJWsq"
String data3;
float h, t, f;
```

```
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/command/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
```

```
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback ,wifiClient);
```

```
void setup()
{
    Serial.begin(115200);
    dht.begin();
    pinMode(LED,OUTPUT);
    pinMode(BUZZER,OUTPUT);
    digitalWrite(LED,LOW);
    digitalWrite(BUZZER,LOW);
    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
    LCD.init();
    LCD.backlight();
    LCD.setCursor(0, 0);
    LCD.print("Connecting to ");
    LCD.setCursor(0, 1);
    LCD.print("WiFi ");
    delay(1000);
    LCD.clear();
}
```

```
void loop()
{
    LCD.setCursor(0,2);
    LCD.print("Smoke: ");
    LCD.setCursor(0, 0);
    LCD.print("Temp: ");
    LCD.setCursor(14, 0);
    LCD.print("C");
    LCD.setCursor(0, 1);
    LCD.print("Humid: ");
    LCD.setCursor(14, 1);
    LCD.print("%");
    h = dht.readHumidity();
    t = dht.readTemperature();
```

```

f = random(0,900);
if (f>300)
{
    Serial.print("Smoke: ");
    Serial.println("Detected");
    digitalWrite(LED,HIGH);
    digitalWrite(BUZZER,HIGH);
    LCD.setCursor(7, 2);
    LCD.print("YES");
    LCD.setCursor(0, 3);
    LCD.print("WARNING! FIREACCIDENT");
}
else{
    Serial.print("Smoke: ");
    Serial.println("Not Detected");
    digitalWrite(LED,LOW);
    digitalWrite(BUZZER,LOW);
    LCD.setCursor(7, 2);
    LCD.print(" NO");
    LCD.setCursor(0, 3);
    LCD.print("                ");
}
Serial.print("temp:");
Serial.println(t);
LCD.setCursor(7, 0);
LCD.print(t);
Serial.print("Humid:");
Serial.println(h);
LCD.setCursor(7, 1);
LCD.print(h);

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

void PublishData(float temp, float humid, float smoke) {

    mqttconnect();
    String payload = "{\"temp\":";
    payload += temp;
    payload += "," " \"Humid\":";
    payload += humid;
    payload += "," " \"Smoke\":";
    payload += smoke;
    payload += "}";

```

```

Serial.print("Sending payload: ");
Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");
} 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()
{
    Serial.println();
    Serial.print("Connecting to ");

    WiFi.begin("Wokwi-GUEST", "", 6);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    LCD.setCursor(0, 0);
    LCD.print("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++) {
        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="";
}

```