

# PROJECT DEVELOPMENT PHASE

## SPRINT-4

<b>TEAM ID</b> :	PNT2022TMID26511
<b>PROJECT NAME</b> :	INDUSTRY SPECIFIC INTELLIGENT FIRE MANGEMENT SYSTEM

```
#include <WiFi.h>
#include <PubSubClient.h>
#define temp_pin 15
void callback(char* subscribetopic,byte* payload, unsigned int payloadLength);
#define ORG "dvo306"
#define DEVICE_TYPE "sona22devicetype"
#define DEVICE_ID "sona22"
#define TOKEN "sona22102001" String
data3;

char server[]= ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[]="iot-2/evt/Data/fmt/json";
char subscribeTopic[]="iot-2/cmd/test/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);


// should match the Beta Coefficient of the thermistor

void setup() {
  Serial.begin(9600);
  analogReadResolution(10);
  pinMode(32,INPUT);
  pinMode(14,OUTPUT);

  wificonnect();
  mqttconnect();
}

void loop() {
  const float BETA = 3950; // should match the Beta Coefficient of the
thermistor int analogValue = analogRead(A4);
float temp = 1 / (log(1 / (1023. / analogValue - 1)) / BETA + 1.0 / 298.15) - 273.15;
```

```

    //float temp = 1 / (log(1 / (1023. / analogValue - 1)) / BETA + 1.0 / 298.15) - 273.15;
    Serial.print("Temperature:
"); Serial.print(temp);
    Serial.println(" °C");
    if(temp>=35){
        PublishData2(temp);
        digitalWrite(14, HIGH);
    }else{
        digitalWrite(14, LOW);
        PublishData1(temp);
    }
    delay(1000);
    if(!client.loop()){
        mqttconnect();
    }

    //delay(2000);
}

void PublishData1(float
tem){ mqttconnect();
String payload=
"{\"temp\":"; payload +=
tem;
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 PublishData2(float
tem){ mqttconnect();
String payload= "{\"ALERT\":";
payload += tem;
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 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"); 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(14,HIGH);
    }else{

```

```

Serial.println(data3)
;
digitalWrite(14,LOW);
}
data3="";
}

```

## DIAGRAM:

The screenshot shows the Wokwi web IDE interface. On the left, the 'sketch.ino' file contains the following code:

```

1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 #define temp_pin 15
4 void callback(char* topic, byte* payload, unsigned int payloadLength);
5 #define ORG "dvo306"
6 #define DEVICE_TYPE "sona22devicetype"
7 #define DEVICE_ID "sona22"
8 #define TOKEN "sona22102001"
9 String data3;
10
11 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
12 char publishTopic[] = "iot-2/evt/Data/fmt/json";
13 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
14 char authMethod[] = "use-token-auth";
15 char token[] = TOKEN;
16 char clientId[] = "d:" + ORG + ":" + DEVICE_TYPE + ":" + DEVICE_ID;
17
18 WiFiClient wifiClient;
19 PubSubClient client(server, 1883, callback, wifiClient);
20
21
22
23
24 // should match the Beta Coefficient of the thermistor
25
26 void setup() {
27   Serial.begin(9600);
28   analogReadResolution(10);
29   pinMode(32, INPUT);
30   pinMode(14, OUTPUT);
31
32   wifiConnect();
33   mqttConnect();
34

```

On the right, the 'Simulation' window shows a visual representation of the ESP32 board connected to a thermistor and an LED. The temperature is displayed as 23.99 °C, and the MQTT payload being sent is {"temp":23.99}. The status bar at the bottom indicates the temperature is 29°C and the date is 17-11-2022.

This screenshot shows the same Wokwi web IDE interface, but with the MQTT log expanded. The log shows multiple successful publish events, each with the temperature 23.99 °C and the payload {"temp":23.99}. The status bar at the bottom indicates the temperature is 29°C and the date is 17-11-2022.

Node-RED : node-red-uwrpn-20... MIT App Inventor MIT App Inventor IBM Watson IoT Platform

dvo306.internetofthings.ibmcloud.com/dashboard/security

211719106007@smartinternz.com ID: dvo306

← Back Close Save

### Connection Security

Use the Connection Security policy to set the default security level that is applied to all devices. You can then add custom rules for specific devices.

#### Default Rule

Define the default connection security level to use for all device types that do not have custom rules defined.

Scope	Security Level	# of Devices
Default	TLS Optional	0 devices

#### Custom Rules

You can define custom connection rules for specific device types. Custom rules overwrite the default rule for the specified device types.

1 Simulation running

Wowki link:

<https://wokwi.com/projects/347767927273947730>