

# INDUSTRY – SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM

Team ID: PNT2022TMID53736

Sprint 3: Mit app inventor, dashboard (application for your project using Mit app, design the model and test the app)

## **CODE:**

```
#include <WiFi.h>//library for wifi
```

```
#include <PubSubClient.h>//library for MQTT
```

```
#include "DHT.h"// Library for dht11
```

```
#include <cstdlib>
```

```
#include <time.h>
```

```
#define DHTPIN 15    // what pin we're connected to
```

```
#define DHTTYPE DHT22 // define type of sensor DHT 11
```

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

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

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

```
#define ORG "fvs923"
```

```
#define DEVICE_TYPE "zenabc"
```

```
#define DEVICE_ID "221"
```

```
#define TOKEN "12345678"
```

```
String data3 = "";
```

```
String accidentstatus = "";
```

```
String sprinkstatus = "";
```

```
float temp =0;
```

```
bool isfanon = false;
```

```
bool issprinkon = false;
```

```
int gas = 0;
```

```
int flame = 0;
```

```
int flow = 0;
```

```
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; // 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();
```

```
  //if real gas sensor is used make sure the senor is heated up for acurate readings
```

```
  /*
```

```
  - Here random values for readings and stdout were used to show the  
    working of the devices as physical or simulated devices are not  
    available.
```

```
  */
```

```

delay(10);
Serial.println();
wificonnect();
mqttconnect();
}

void loop()// Recursive Function
{

temp = dht.readTemperature();
//setting a random seed
srand(time(0));

//initial variable activities like declaring , assigning
gas = rand()%400;
int flamereading = rand()% 1024;
flame = map(flamereading,0,1024,0,1024);
int flow = ((rand()% 100)>50?1:0);

//find the accident status 'cause fake alert may be caused by some mischief activities
if(temp < 45 ){
    if(flame > 650){
        accidentstatus = "Need Auditing";
        isfanon = true;
        issprinkon = false;
    }
}

```

```
else if(flame <= 10){  
    accidentstatus = "nothing happened";  
    isfanon = false;  
    issprinkon = false;  
}
```

```
}else if(temp >= 45 && temp <= 55 ){  
    if(flame <=650 && flame >100 ){  
        issprinkon = true;  
        accidentstatus = "moderate";  
        if(gas > 150){  
            isfanon = true;  
        }  
        else{  
            isfanon = false;  
        }  
    }  
}
```

```
else if(flame <= 100 && flame > 10){  
    issprinkon = true;  
    isfanon = false;  
    accidentstatus = "moderate";  
}
```

```
}else if(temp > 55){  
    if(flame > 650){  
        gas = 500 + rand()%500;  
        accidentstatus = "severe";  
        issprinkon = true;  
        isfanon = true;  
    }  
}
```

```
else if(flame < 650 && flame > 400 ){  
    gas = 300 + rand()%500;  
    accidentstatus = "severe";  
    issprinkon = true;  
    isfanon = true;  
}  
}  
else {  
    accidentstatus = "Need Auditing";  
    isfanon = false;  
    issprinkon = false;  
}
```

```
if(issprinkon){  
    if(flow){  
        sprinkstatus = "working";  
    }  
    else{  
        sprinkstatus = "not working";  
    }  
}  
else if(!issprinkon){  
    sprinkstatus = "ready";  
}  
else {  
    sprinkstatus = "something's wrong";  
}
```

```

PublishData(temp,gas,flame,flow,isfanon,issprinkon);
delay(1000);
if (!client.loop()) {
    mqttconnect();
}
}

```

```

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

```

```

void PublishData(float temp, int gas ,int flame ,int flow,bool isfanon,bool issprinkon) {
    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 += "," "\"gas\":";
    payload += gas;
    payload += "," "\"flame\":";
    payload += flame;
    payload += "," "\"flow\":";
    payload += ((flow)?"true":"false");
    payload += "," "\"isfanon\":";
    payload += ((isfanon)?"true":"false");
    payload += "," "\"issprinkon\":";
    payload += ((issprinkon)?"true":"false");
    payload += "," "\"accidentstatus\":";
    payload += "\"" + accidentstatus + "\"";
}

```

```
payload += "," "\"sprinkstatus\":";
payload += "\"" + sprinkstatus + "\"";
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=="foo")
{
Serial.println(data3);

}
else
{
Serial.println(data3);

}
data3="";
}

```

### **diagram.json:**

```

{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 4.8, "left": -127.69, "attrs": { } },
    { "type": "wokwi-dht22", "id": "dht1", "top": -76.72, "left": 137.76, "attrs": { } }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],

```

```
[ "esp:RX0", "$serialMonitor:TX", "", [ ] ],  
[ "dht1:VCC", "esp:3V3", "red", [ "v0" ] ],  
[ "dht1:GND", "esp:GND.1", "black", [ "v0" ] ],  
[ "dht1:SDA", "esp:D15", "green", [ "v101.76", "h-2.06" ] ]  
]  
}
```