Sprint 4

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

Project 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
#define BUZZER 4
#define EXFAN1 13
#define EXFAN2 14
#define SPRKLR 12

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

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

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 "pol6f4"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP-32IoT"//Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "100100C40A24"//Device ID mentioned in ibm watson IOT
Platform
#define TOKEN "EnuF+Tgx4O@Y!wJWsq"     //Token
String data3;
float h, t, s,f,fan1=0,fan2=0,spr=0;


//-------- 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);
  pinMode(BUZZER,OUTPUT);
  pinMode(EXFAN1,OUTPUT);
  pinMode(EXFAN2,OUTPUT);
  digitalWrite(LED,LOW);
  digitalWrite(BUZZER,LOW);
  digitalWrite(EXFAN1,LOW);
  digitalWrite(EXFAN2,LOW);
  pinMode(SPRKLR,OUTPUT);
  digitalWrite(SPRKLR,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()// Recursive Function
{
    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();
s = random(0,900);
f = random(0,255);
if (s>400)
{
  Serial.print("Smoke: ");
  Serial.println("Detected");
  digitalWrite(BUZZER,HIGH);
   LCD.setCursor(7, 2);
  LCD.print("YES");
  LCD.setCursor(0, 3);
  LCD.print("WARNING!FIREACCIDENT");
  digitalWrite(EXFAN1,HIGH);
  digitalWrite(EXFAN2,HIGH);
   fan1=1;
    fan2=1;
}
else{
    Serial.print("Smoke: ");
    Serial.println("Not Detected");
    digitalWrite(BUZZER,LOW);
    LCD.setCursor(7, 2);
   LCD.print(" NO");
   LCD.setCursor(0, 3);
  LCD.print("                    ");
   digitalWrite(EXFAN1,LOW);
  digitalWrite(EXFAN2,LOW);
   fan1=0;
    fan2=0;
}
if(f>80)
{
    Serial.print("Flame: ");
    Serial.println("Detected");
    spr=1;
    digitalWrite(SPRKLR,HIGH);
}
else
{
    Serial.print("Flame: ");
    Serial.println("nOT Detected");
    spr=0;
    digitalWrite(SPRKLR,LOW);
}
```

```
  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, s,fan1,fan2,spr);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}



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

void PublishData(float temp, float humid, float smoke,float fan1,float
fan2,float spr) {
  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 += "," "\"Smoke\":";
  payload += smoke;
   payload += "," "\"F1STATUS\":";
  payload += fan1;
   payload += "," "\"F2STATUS\":";
  payload += fan2;
     payload += "," "\"SPRKLR\":";
  payload += spr;
  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");
  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++) {
    //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="";
}
```