Sprint 3

Team ID:PNT2022TMID27908

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include <ESP32Servo.h>
#include "DHT.h"// Library for dht11
#include <Stepper.h>
#define DHTPIN 5     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define SERVO_PIN 22 //servo motor connection
#define BUZZER_PIN 2//buffer connecton
DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of dht
connected
Servo servoMotor;
Servo servoMotor2;
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

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

#define ORG "py0epl"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"     //Token
const int tempHigh=50;
const int firingHigh = 90;
const int gasHigh=400;
String gasData;
String flameData;
String tempData;
 float templevel=0;
  float flamelevel;
const int stepsPerRevolution = 200;  //
Stepper myStepper(stepsPerRevolution, 13, 12, 14, 26);

//-------- 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);
 myStepper.setSpeed(60);
   pinMode(BUZZER_PIN, OUTPUT);
   servoMotor.attach(SERVO_PIN);
   dht.begin();
  delay(10);
  Serial.println();


  wificonnect();
  mqttconnect();
}

void loop()// Recursive Function
{
  //int steps=200000;
  templevel= dht.readTemperature();
  float analogValue = analogRead(36);
  float gaslevel=0;
  gaslevel = random(100,900);
  Serial.print(gaslevel);
  Serial.println("Sensor RAW: ");
  Serial.println(analogValue, 0);
 flamelevel = map(analogValue, 0, 1024, 100, 0);
  Serial.print(flamelevel, 0);
  Serial.println("%");
  if (flamelevel >= firingHigh ) { // stoker is fully firing
  tone(BUZZER_PIN,2000);
   servoMotor.write(180);
    delay(300);
    flameData="alert";
  }
  else{
    flameData="safe";
    noTone(BUZZER_PIN);
     servoMotor.write(0);
  }
  Serial.print("Flame Message sending to authority :");
  Serial.println(flameData);
  if(gaslevel>= gasHigh){
    tone(BUZZER_PIN,2000);
      myStepper.step(stepsPerRevolution);
    delay(300);
    gasData="alert";
  }
  else{
    gasData="safe";
         myStepper.step(-stepsPerRevolution);
    noTone(BUZZER_PIN);

  }
  Serial.print("Gas Message sending to authority :");
  Serial.println(gasData);
```

```
   if(templevel>= tempHigh){
    tone(BUZZER_PIN,2000);
    delay(300);
    tempData="alert";
  }
  else{
    tempData="safe";
    noTone(BUZZER_PIN);

  }
  Serial.print("Temperature Message sending to authority :");
  Serial.println(tempData);

  PublishData(gaslevel,flamelevel,templevel);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}



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

void PublishData(float gaslevel,float flamelevel,float templevel) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"gaslevel\":";
  payload += gaslevel;
  //payload += "," "\"GasMsg\":";
  //payload += gasData;
  payload += "," "\"flamelevel\":";
  payload += flamelevel;
  //payload += "," "\"FlameMsg\":";
  //payload += flameData;
  payload += "," "\"templevel\":";
  payload += templevel;
  //payload += "," "\"TemperatureMsg\":";
  //payload += tempData;
  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);

}
```