```
Include <WiFi.h>//library for wifi

#include <PubSubClient.h>//library for MQtt

#include “DHT.h”// Library for dht11

#include <cstdlib>

#include <time.h>

#include <mjson.h>


//#include <HTTPClient.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 typr of dht connected


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


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


#define ORG “odvjnt”

#define DEVICE_TYPE “1312”

#define DEVICE_ID “22”

#define TOKEN “12345678”


String data3 = “”;

String accidentstatus =””;

String sprinkstatus = “”;

Float temp =0;

Bool isfanon = false;

Bool issprinkon = false;
```

```
Bool cansprinkoperate = true;

Bool canfanoperate = true;

Bool canalertsent = true;

Bool cansentalert = false;

Int gas = 0;

Int flame = 0;

Int flow = 0;

Long int cooldown= 600;




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()
{
 Temp = dht.readTemperature();
 //setting a random seed (only for random values not in real life scenarios)
 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";
```

```
    If(canfanoperate)
      Isfanon = true;
    Else
      Isfanon = false;
    Issprinkon = false;
  }
  Else if(flame <= 10){
    Accidentstatus = “nothing happened”;
    Isfanon = false;
    Issprinkon = false;
  }
}else if(temp >= 45 && temp <= 55 ){
  If(flame <=650 && flame >100  ){

    If(cansprinkoperate)
      Issprinkon = true;
    Else
      Issprinkon = false;
    Accidentstatus = “moderate”;
    If(gas > 160 && canfanoperate ){
      Isfanon = true;
    }
    Else{
      Isfanon = false;
    }
  }
  Else if(flame <= 100 && flame > 10){
   If(cansprinkoperate)
      Issprinkon = true;
```

```
    Else
      Issprinkon = false;
    Isfanon = false;
    Accidentstatus = “moderate”;
  }

}else if(temp > 55){
  If(flame > 650){
    Gas = 500 + rand()%500;
    Accidentstatus = “severe”;
    If(cansprinkoperate)
      Issprinkon = true;
    Else
      Issprinkon = false;
    If(canfanoperate)
      Isfanon = true;
    Else
      Isfanon = false;
  }
  Else if(flame < 650 && flame > 400 ){
    Gas = 300 + rand()%500;
    Accidentstatus = “severe”;
    If(cansprinkoperate)
      Issprinkon = true;
    Else
      Issprinkon = false;

    If(canfanoperate)
      Isfanon = true;
```

```
    Else
      Isfanon = false;
    }
  }
  Else {
   Accidentstatus = “Need moderate 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);
}
```

```
  If(cooldown > 999999){
    Cooldown = 601;
  }



Delay(1000);
  ++cooldown;
  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 += “,” “\”cansentalert\”:”;
 Payload += ((cansentalert)?”true”:”false”);
 Payload += “,” “\”accidentstatus\”:”;
 Payload += “\””+accidentstatus+”\””;
 Payload += “,” “\”sprinkstatus\”:”;
 Payload += “\””+sprinkstatus+”\””;
 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);
  While (WiFi.status() != WL_CONNECTED) {
    Delay(100);
    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");
  }
}

//handles commands from user side
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
}
Serial.println(“data: “+ data3);

  Const char *s =(char*) data3.c_str();
  Double pincode = 0;
  If(mjson_get_number(s, strlen(s), “$.pin”, &pincode)){
   If(((int)pincode)==67993){
      Const char *buf;
      Int len;

      If (mjson_find(s, strlen(s), “$.command”, &buf, &len))  // And print it
      {
        String command(buf,len);
        If(command==”\”cantfan\””){

          Canfanoperate = !canfanoperate;
        }
        Else if(command==”\”cantsprink\””){
          Cansprinkoperate = !cansprinkoperate;
        }else if(command==”\”sentalert\””){
          Resetcooldown();
```

```
        }
       }

 }
}


      Data3="";
}

Void resetcooldown(){
  Cooldown = 0;
}

//sent alert request to node-red
Void sendalert(){
 cansentalert=ture;
Cooldown  = 0;

}
```