INDUSTRY – SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM

Sprint 4: Web UI (to make the user interact with the software)

CODE:

```
#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 "sms611"

#define DEVICE_TYPE "3114"

#define DEVICE_ID "14"

#define TOKEN "98765432"


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()// Recursive Function
{
  //printf("\n%li\n",time(0));


  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";
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);
printf("\n%li\n",cooldown);
  if(accidentstatus=="severe" && cooldown >= 600){
cooldown = 0;     sendalert();
    PublishData(temp,gas,flame,flow,isfanon,issprinkon);
cansentalert = false;


  }


  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 += "}";


  //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(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");
  }
}

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);
```

```
  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;
```

```
} void sendalert(){
cansentalert = true;
cooldown = 0;
 }
```

diagram.json:

```
{
 "version": 1,
 "author": "Anonymous maker",
 "editor": "wokwi",
 "parts": [
  { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 2.13, "left": -176.36, "attrs": {} },
  {
   "type": "wokwi-dht22",
   "id": "dht1",
   "top": -72.84,
   "left": -48.9,
   "attrs": { "temperature": "10.9" }
  }
 ],
 "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" ] ]
 ]
}
```