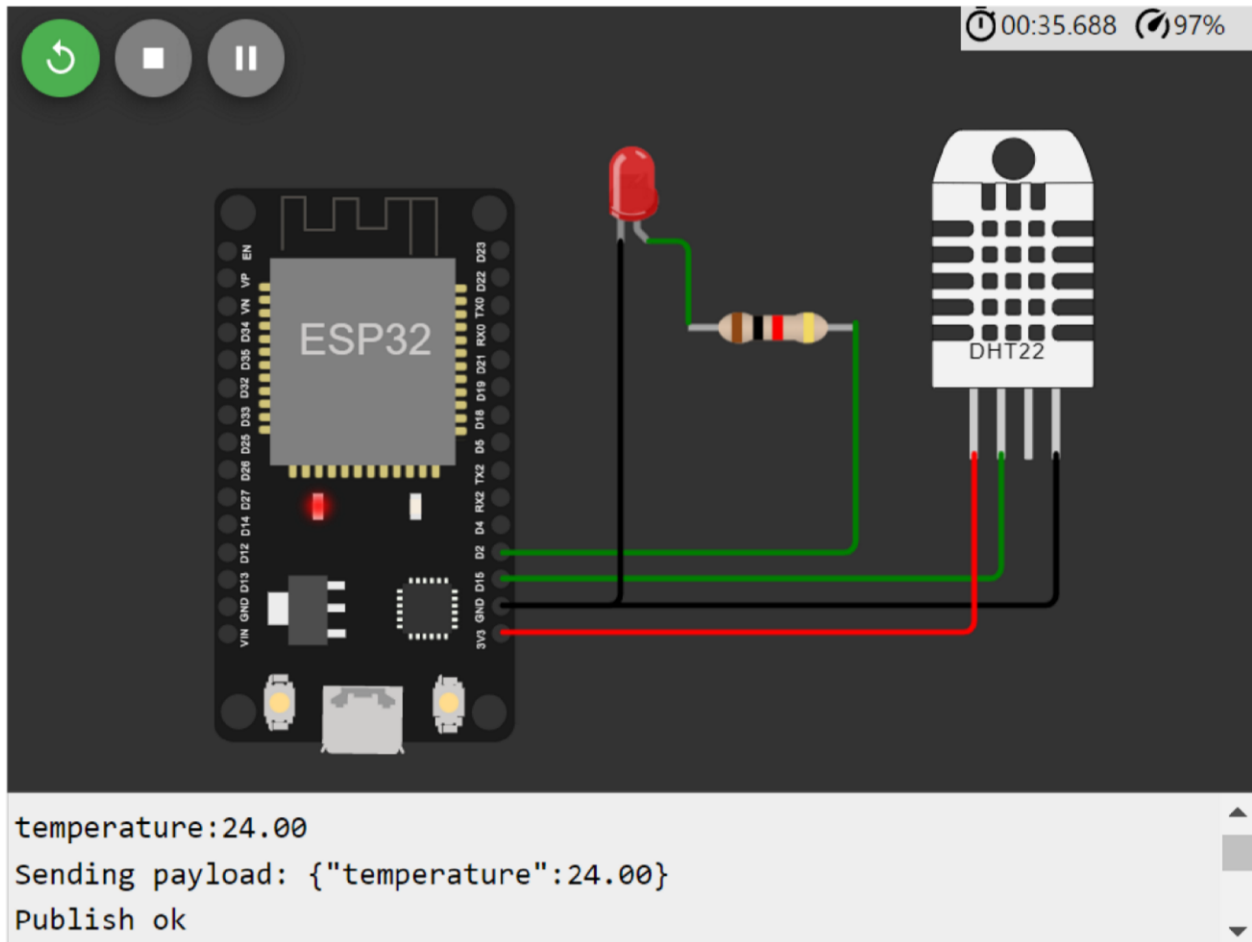


Sprint-1

Date	7 November 2022
Team ID	PNT2022TMID53943
Project Name	Industry-Specific Intelligent Fire Management System

Display the temperature values:

Submitted by: CROSSLIN V, AJAY.C, KARPAGARAJA M, MICHAEL JEMIN GEORGE F



Program:

```
#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

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 "zbgr67"//IBM ORGANITION ID

#define DEVICE_TYPE "fershidevicetype"//Device type mentioned in ibm watson
IOT Platform

#define DEVICE_ID "fershideviceid"//Device ID mentioned in ibm watson IOT
Platform

#define TOKEN "fershiageona"    //Token
String data3; float t;

//----- 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);    delay(10);
```

```
    Serial.println();    wificonnect();
```

```
    mqttconnect();
```

```
} void loop()// Recursive
```

```
Function
```

```
{
```

```
    t = dht.readTemperature();
```

```
    Serial.print("temperature:");
```

```
    Serial.println(t);
```

```
    PublishData(t);    delay(1000);    if  
(!client.loop()) {    mqttconnect();
```

```
}
```

```
}
```

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

```
void PublishData(float temp) {    mqttconnect();//function  
call for connecting to ibm
```

```
/*      creating the String in in form JSon to update the data to ibm  
cloud    */
```

```
String payload = "{\"temperature\":\"";    payload  
+= temp;    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=="lighton")

{

```

```
Serial.println(data3); digitalWrite(LED,HIGH);
```

```
} else
```

```
{
```

```
Serial.println(data3); digitalWrite(LED,LOW);
```

```
  } data3="";
```

```
}
```

Displaying flame sensor values:

Submitted by: Nandhini,parwin,preethi,Rathidevi,Sathya

Student Roll no: 810419104067,810419104078,810419104091,810419104094,810419104100

IBM Watson IoT Platform

962819106014@smartinternz.com
ID: zbg67

Browse

Action

Device Types

Interfaces

Add Device +

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
event_1	{"Gas":100,"Flame":0}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago

Code :

Device Type: fershidevicetype

Events 1

New event type +

Event type name

event_1

Send

Schedule

20

Every Minute

Payload

Specify the event payload in the editor window or by uploading a [CSV file.](#)

0 {

1 "Gas": random(0, 100),

2 "Flame": random(0,1)

3 }

4

Displaying gas sensor values:

Submitted by: Rathidevi,Nandhini,Parwin,Preethi,Sathya

Student Roll number:810419104094,810419104067,810419104078,810419104091,810419104100



Browse

Action

Device Types

Interfaces

Add Device +

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
event_1	{"Gas":100,"Flame":0}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago
Data	{"temperature":24}	json	a few seconds ago

Code:

Device Type: fershidevicetype

Events1

New event type +

Event type name

event_1

Send

Schedule

20

Every Minute

Payload

Specify the event payload in the editor window or by uploading a [CSV file.](#)

0

{

1

"Gas": random(0, 100),

2

"Flame": random(0,1)

3

}

4