

## Sprint-1

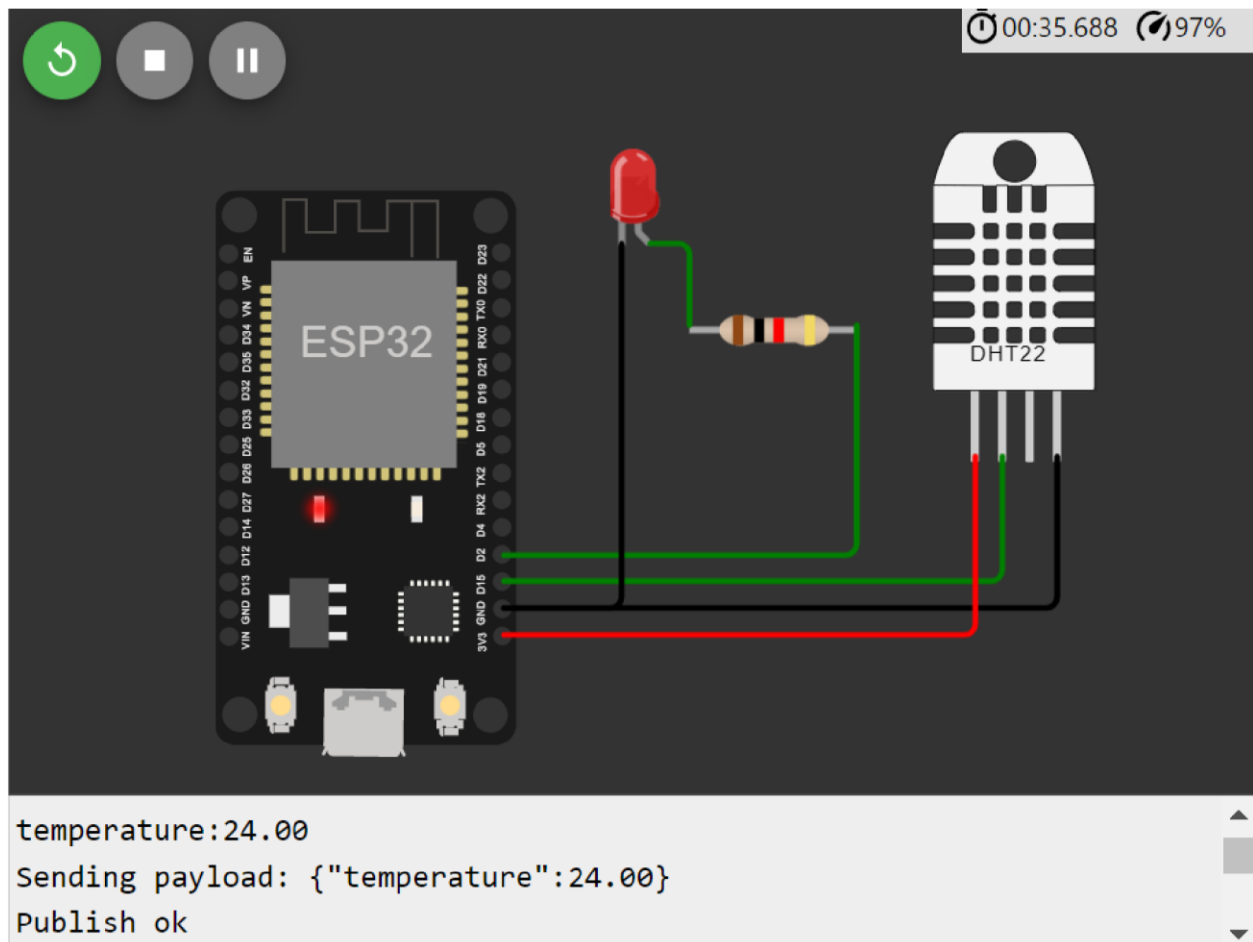
Date	7 November 2022
Team ID	PNT2022TMID34904
Project Name	Industry-Specific Intelligent Fire Management System

### Display the temperature values:

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

Student Roll number:810419104067,810419104078,810419104091,810419104094,810419104-100

Wokwi link: <https://wokwi.com/projects/347571790373978706>



## 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:Godsy D,Aswini A

Student Roll no:962819106016,962819106008

The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various functions. The main content area displays a table of recent events. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. The events listed are 'event\_1' with a JSON value, and four 'Data' events with temperature values.

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 :

The screenshot shows the configuration page for a device type named 'fershidevicetype'. It includes sections for 'Events', 'Schedule', and 'Payload'. The 'Events' section shows a single event type 'event\_1' with a 'Send' button. The 'Schedule' section shows a frequency of 'Every Minute'. The 'Payload' section shows a JSON payload with 'Gas' and 'Flame' values generated by random functions.

Device Type: fershidevicetype

**Events** 1 New event type +

Event type name: event\_1 Send Trash

**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: Fershia G Geona, Reshma Xavier

Student Roll number: 962819106014, 962819106033

IBM Watson IoT Platform

962819106014@smartinternz.com  
ID: zbgr67

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	
<div style="display: flex; justify-content: space-between; align-items: center;"><div><b>Events</b> <span style="background-color: #e0e0e0; padding: 2px 5px;">1</span></div><div style="background-color: #005596; color: white; padding: 5px 10px; border-radius: 3px;">New event type <span style="font-size: 1.2em;">+</span></div></div>	
▼ <b>Event type name</b>	<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;">event_1</div> <div style="display: flex; justify-content: flex-end; align-items: center; gap: 10px;"><div style="border: 2px solid #005596; padding: 5px 15px; font-weight: bold; color: #005596;">Send</div><div style="font-size: 1.2em;">🗑</div></div>
<b>Schedule</b> <div style="display: flex; margin-top: 5px;"><div style="border: 1px solid #ccc; padding: 5px; width: 150px; text-align: center;">20</div><div style="border: 1px solid #ccc; padding: 5px; width: 150px; text-align: center;">Every Minute ▼</div></div>	
<b>Payload</b> <p style="margin-top: 5px;">Specify the event payload in the editor window or by uploading a <a href="#">CSV file</a>.</p> <div style="border: 1px solid #ccc; padding: 10px; min-height: 150px; margin-top: 5px;"><div style="background-color: #e0f0ff; padding: 2px 5px; margin-bottom: 5px;">0 {</div><div style="padding: 2px 5px; margin-bottom: 5px;">1    "Gas": random(0, 100),</div><div style="padding: 2px 5px; margin-bottom: 5px;">2    "Flame": random(0,1)</div><div style="padding: 2px 5px; margin-bottom: 5px;">3    }</div><div style="padding: 2px 5px;">4</div></div>	