

Sprint3

Team ID	PNT2022TMID04760
Topic	Industry specific intelligent fire management system

```
#include "DHTesp.h"
#include <cstdlib>
#include <time.h>
#include <WiFi.h>
#include <PubSubClient.h>

#define ORG "w9vo0w"
#define DEVICE_TYPE "Vicky"
#define DEVICE_ID "5678"
#define TOKEN "123456789"

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/data/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

WiFiClient wifiClient;
PubSubClient client(server, 1883, wifiClient);

const int DHT_PIN = 15;

bool is_exhaust_fan_on = false;
bool is_sprinkler_on = false;

float temperature = 0;

int gas_ppm = 0;
int flame = 0;
int flow = 0;

String flame_status = "";
String accident_status = "";
String sprinkler_status = "";

DHTesp dhtSensor;
```

```

void setup() {
    Serial.begin(99900);

    /*** sensor pin setups ***/
    dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
    //if real gas sensor is used make sure the sensor is heated up for accurate
    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.
    */

    wifiConnect();
    mqttConnect();
}

void loop() {

    TempAndHumidity data = dhtSensor.getTempAndHumidity();

    //setting a random seed
    srand(time(0));

    //initial variable activities like declaring , assigning
    temperature = data.temperature;
    gas_ppm = rand()%1000;
    int flamereading = rand()%1024;
    flame = map(flamereading,0,1024,0,1024);
    int flamerange = map(flamereading,0,1024,0,3);
    int flow = ((rand()%100)>50?1:0);

    //set a flame status based on how close it is.....
    switch (flamerange) {
    case 2:    // A fire closer than 1.5 feet away.
        flame_status = "Close Fire";
        break;
    case 1:    // A fire between 1-3 feet away.
        flame_status = "Distant Fire";
        break;
    case 0:    // No fire detected.
        flame_status = "No Fire";
        break;
    }

    //toggle the fan according to gas in ppm in the room
    if(gas_ppm > 100){
        is_exhaust_fan_on = true;
    }
}

```

```

}
else{
    is_exhaust_fan_on = false;
}

//find the accident status 'cause fake alert may be caused by some mischief
activities
if(temperature < 40 && flamerange ==2){
    accident_status = "need auditing";
    is_sprinkler_on = false;
}
else if(temperature < 40 && flamerange ==0){
    accident_status = "nothing found";
    is_sprinkler_on = false;
}
else if(temperature > 50 && flamerange == 1){
    is_sprinkler_on = true;
    accident_status = "moderate";
}
else if(temperature > 55 && flamerange == 2){
    is_sprinkler_on = true;
    accident_status = "severe";
}else{
    is_sprinkler_on = false;
    accident_status = "nil";
}

//send the sprinkler status
if(is_sprinkler_on){
    if(flow){
        sprinkler_status = "working";
    }
    else{
        sprinkler_status = "not working";
    }
}
else if(is_sprinkler_on == false){
    sprinkler_status = "now it shouldn't";
}
else{
    sprinkler_status = "something's wrong";
}

//Obviously the output.It is like json format 'cause it will help us for
future sprints
String payload = "{\"senor_values\":{\"";
payload+="\"gas_ppm\":";
payload+=gas_ppm;

```

```

payload+=",";
payload+="\"temperature\":\"";
payload+=(int)temperature;
payload+=",";
payload+="\"flame\":\"";
payload+=flame;
payload+=",";
payload+="\"flow\":\"";
payload+=flow;
payload+="},";
payload+="\"output\":{\"";
payload+="\"is_exhaust_fan_on\":" +String((is_exhaust_fan_on)?"true":"false")
+",";
payload+="\"is_sprinkler_on\":" +String((is_sprinkler_on)?"true":"false")+"";
payload+="},";
payload+="\"messages\":{\"";
payload+="\"fire_status\":" +""+flame_status+"\"";
payload+="\"flow_status\":" +""+sprinkler_status+"\"";
payload+="\"accident_status\":" +""+accident_status+"\"";
payload+="}";
payload+="}";
//Serial.println(payload);

```

```

if(client.publish(publishTopic, (char*) payload.c_str()))
{
    Serial.println("Publish OK");
}
else{
    Serial.println("Publish failed");
}
delay(1000);

```

```

if (!client.loop())
{
    mqttConnect();
}

```

```

}

```

```

void wifiConnect()
{
    Serial.print("Connecting to ");
    Serial.print("Wifi");
    WiFi.begin("Wokwi-GUEST", "", 6);
}

```

```

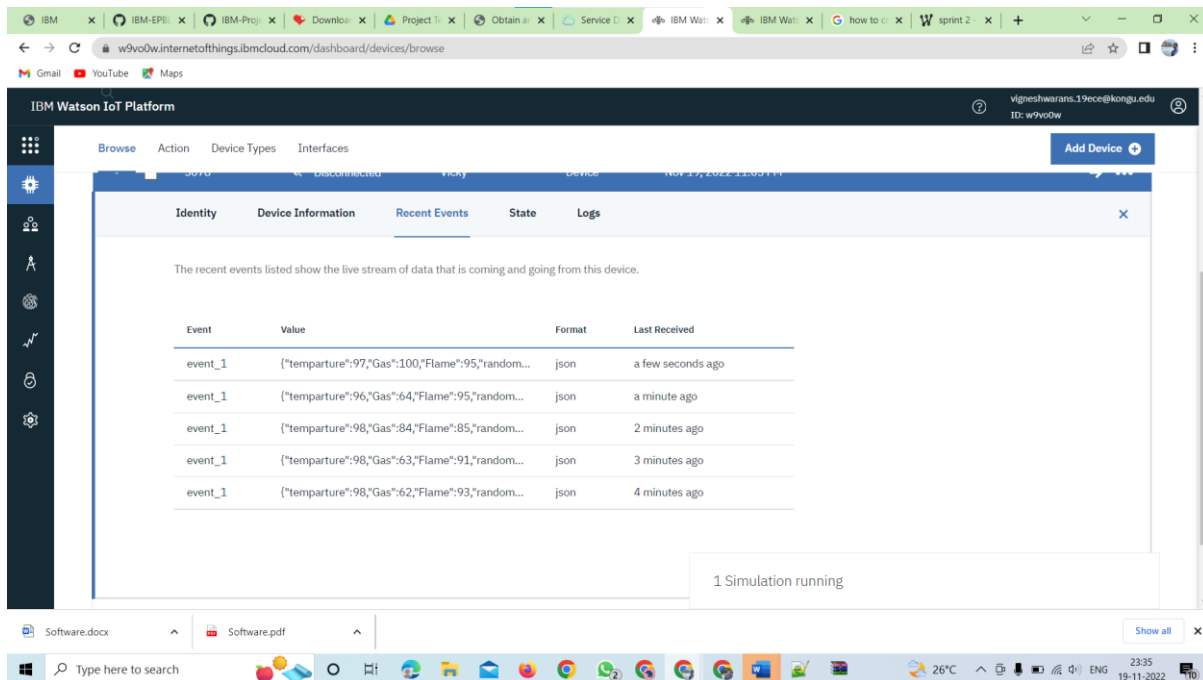
while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
    Serial.print(".");
}
Serial.print("WiFi connected, IP address: ");
Serial.println(WiFi.localIP());

}

void mqttConnect()
{
    if (!client.connected())
    {
        Serial.print("Reconnecting MQTT client to ");
        Serial.println(server);
        while (!client.connect(clientId, authMethod, token))
        {
            Serial.print(".");
            delay(500);
        }

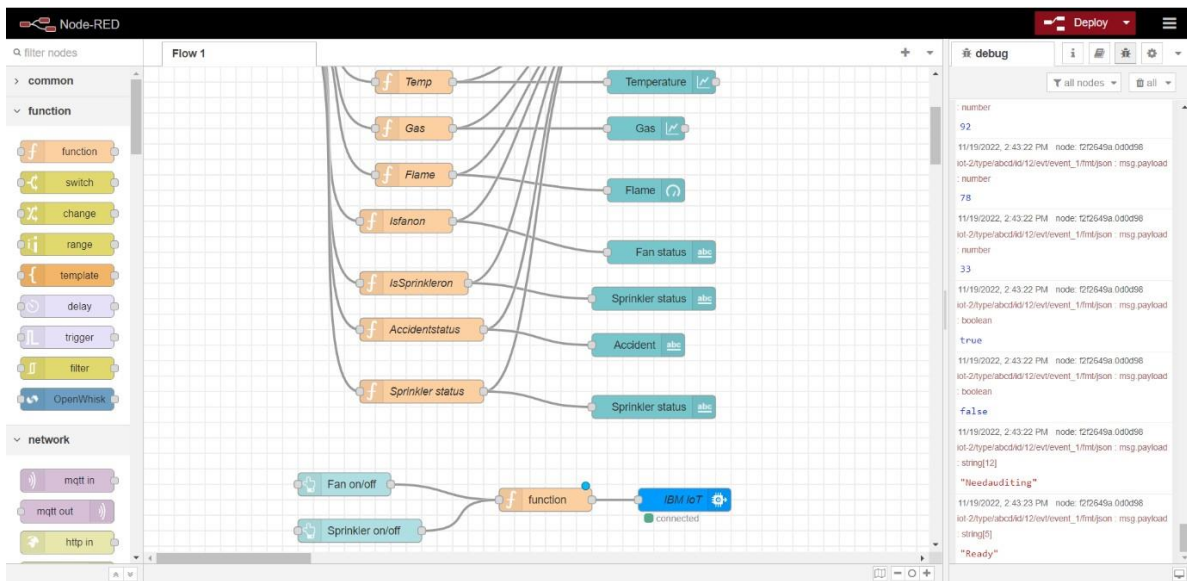
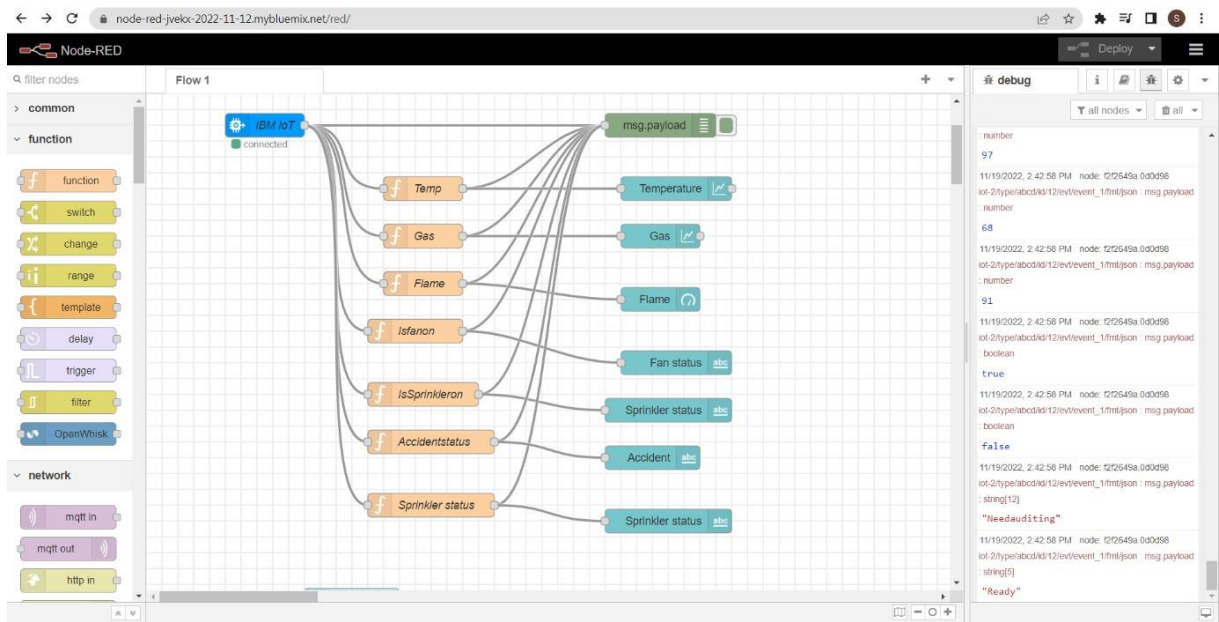
        Serial.println();
    }
}

```



The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various IoT functions. The main content area shows a 'Recent Events' table for a device named 'w9vo0w'. The table lists five events, each with a timestamp and a JSON payload containing temperature, gas, and flame data. A status bar at the bottom indicates '1 Simulation running'.

Event	Value	Format	Last Received
event_1	{"temperature":97,"Gas":100,"Flame":95,"random..."	json	a few seconds ago
event_1	{"temperature":96,"Gas":64,"Flame":95,"random..."	json	a minute ago
event_1	{"temperature":98,"Gas":84,"Flame":85,"random..."	json	2 minutes ago
event_1	{"temperature":98,"Gas":63,"Flame":91,"random..."	json	3 minutes ago
event_1	{"temperature":98,"Gas":62,"Flame":93,"random..."	json	4 minutes ago



Industry

Control unit

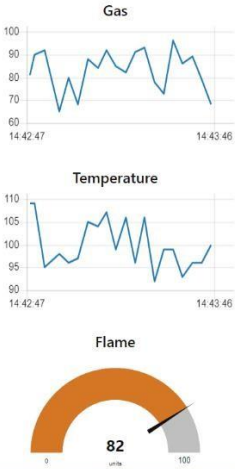
FAN ON/OFF

SPRINKLER ON/OFF

Accident: **Needauditing**

Sprinkler status: **Ready**

Sensor values



Status

Fan status	true
Sprinkler status	false