

SPRINT-4

PROJECT	INDUSTRY-SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM
TEAM ID	PNT2022TMID15035

PYTHON CODE:

```
#include <time.h>
#include <WiFi.h>
#include <PubSubClient.h>

#define ORG "ksgtfi"
#define DEVICE_TYPE "123"
#define DEVICE_ID "123_1"
#define TOKEN "12345678"

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);

float temperature = 0;
int gas = 0;
int flame = 0;

String flame_status = "";
String Gas_status = "";
String exhaust_fan_status = "";
String sprinkler_status = "";

void setup() {
    Serial.begin(99900);
    wifiConnect();
    mqttConnect();
}

void loop() {

    srand(time(0));
```

```

//initial variables and random generated data

temperature = random(-20,125);
gas = random(0,1000);
int flamereading = random(200,1024);
flame = map(flamereading,200,1024,0,2);

//set a flame status

switch (flame) {
case 0:
    flame_status = "No Fire";
    break;
case 1:
    flame_status = "Fire is Detected";
    break;
}

//send the sprinkler status

if(flame==1){
    sprinkler_status = "Working";
}
else{
    sprinkler_status = "Not Working";
}

//toggle the fan according to gas reading

if(gas > 100){
    Gas_status = "Gas Leakage is Detected";
    exhaust_fan_status = "Working";
}
else{
    Gas_status = "No Gas Leakage is Detected";
    exhaust_fan_status = "Not Working";
}

//json format for IBM Watson

String payload = "{";
payload+="\"gas\":";
payload+=gas;
payload+=",";
payload+="\"temperature\":";
payload+=(int)temperature;

```

```

    payload+=",";
    payload+="\"flame\":";
    payload+=flamereading;
    payload+=",";
    payload+="\"fire_status\":"+"\""+flame_status+"\"",";
    payload+="\"sprinkler_status\":"+"\""+sprinkler_status+"\"",";
    payload+="\"Gas_status\":"+"\""+Gas_status+"\"",";
    payload+="\"exhaust_fan_status\":"+"\""+exhaust_fan_status+"\""}";

    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))

```

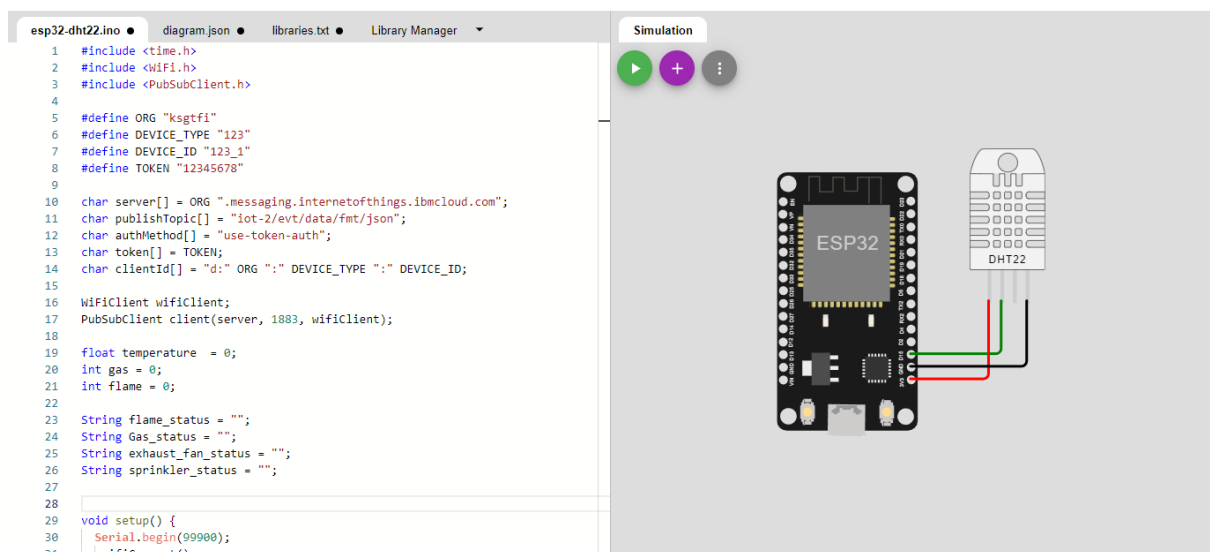
```

    {
        Serial.print(".");
        delay(500);
    }

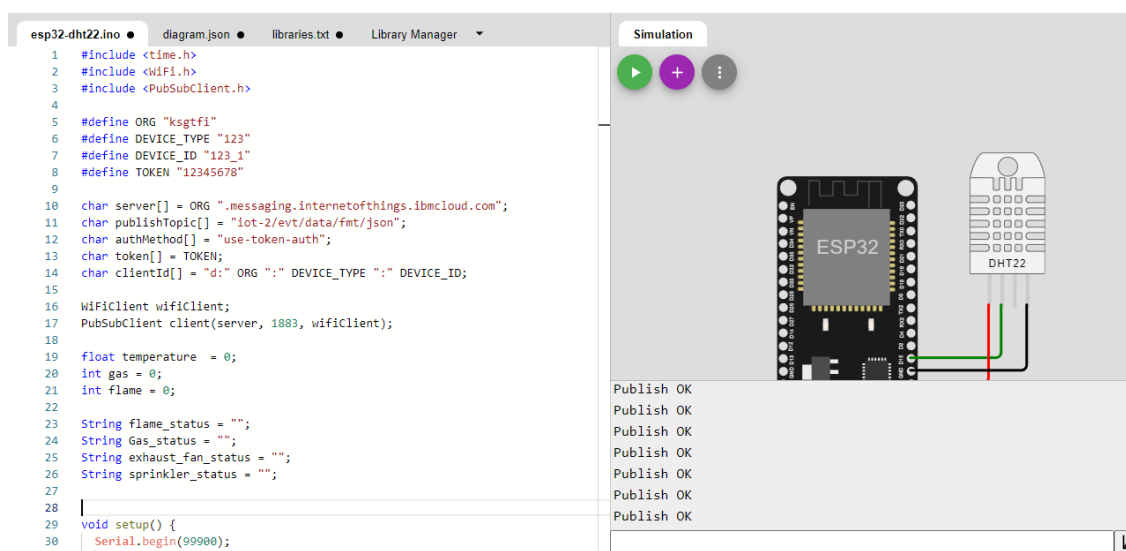
    Serial.println();
}
}

```

WOKWI CONNECTION:



WOKWI OUTPUT:



WATSON IOT PLATFORM:

The screenshot shows the IBM Watson IoT Platform interface. The main page is titled 'Browse Devices' and includes a table of devices. A modal window is open for creating a new event type for device '123'.

Device Table:

Device ID	Status	Device Type
123456	Disconnected	123
123_1	Connected	123

Event Type Modal (Device Type: 123):

- Event type name:** eventtest
- Schedule:** 2 Every Minute
- Payload:**

```
{
  "gas": random(200, 1000),
  "temp": random(100, 800),
  "flame": random(200, 800),
  "firestatus": "No Fire",
  "sprinklerstatus": "Not Working",
  "gasstatus": "Working",
  "exhaustfanstatus": "Gas Leakage is Detected"
}
```

OUTPUT:

The screenshot shows the 'Device Drilldown - 123_1' page. It displays connection information and a list of recent events.

Recent Events:

Event	Value	Format	Last Received
eventtest	{"gas":725,"temp":453,"flame":264,"firestatus": "...	json	a few seconds ago
eventtest	{"gas":791,"temp":427,"flame":577,"firestatus": "...	json	a few seconds ago
eventtest	{"gas":580,"temp":110,"flame":796,"firestatus": "...	json	a minute ago
eventtest	{"gas":339,"temp":409,"flame":228,"firestatus": "...	json	a minute ago
eventtest	{"gas":293,"temp":748,"flame":665,"firestatus": "...	json	2 minutes ago

5 Simulations running

⌵

🔧

👤

🏠

🌐

📶

🔒

⚙️

← Back

Device Drilldown - 123_1

Connection Information

Recent Events

State

Device Information

Metadata

Diagnostics

Connection Logs

Device Actions

Property	Value	Type	Event	Last Received
gas	791	Number	eventtest	a few seconds ago
temp	427	Number	eventtest	a few seconds ago
flame	577	Number	eventtest	a few seconds ago
firestatus	No Fire	String	eventtest	a few seconds ago
sprinklerstatus	Not Working	String	eventtest	a few seconds ago
gasstatus	Working	String	eventtest	a few seconds ago
exhaustfanstatus	Gas Leakage is Detected	String	eventtest	a few seconds ago

5 Simulations running

NODE-RED :

Node-RED on IBM Cloud

Node-RED

Flow-based programming for the Internet of Things

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

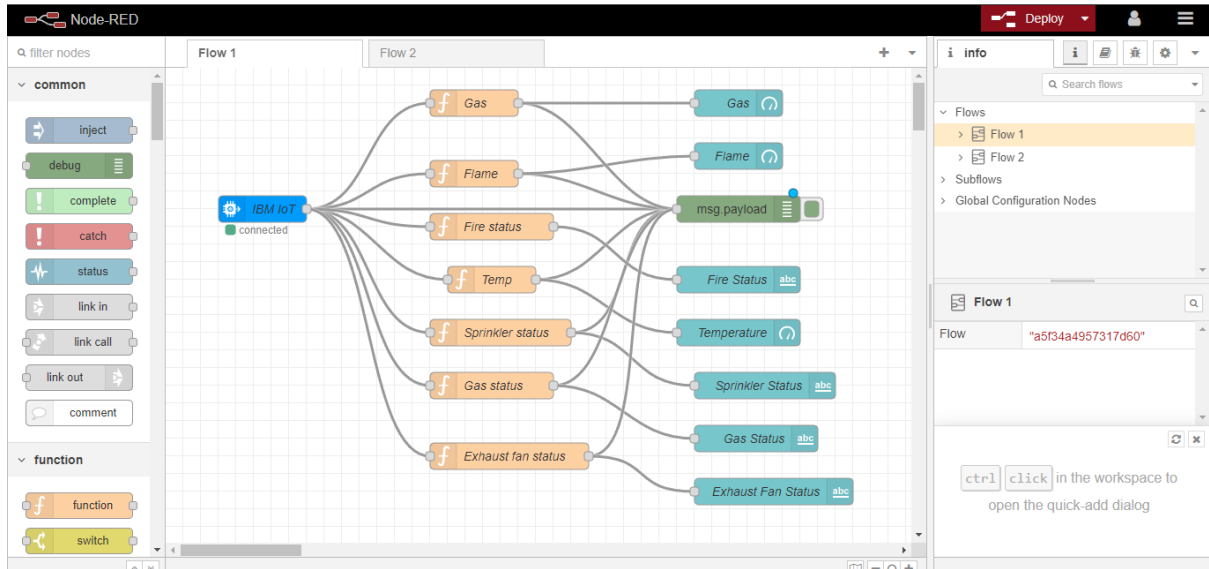
This instance is running as an IBM Cloud application, giving it access to the wide range of services available on the platform.

More information about Node-RED, including documentation, can be found at nodered.org.

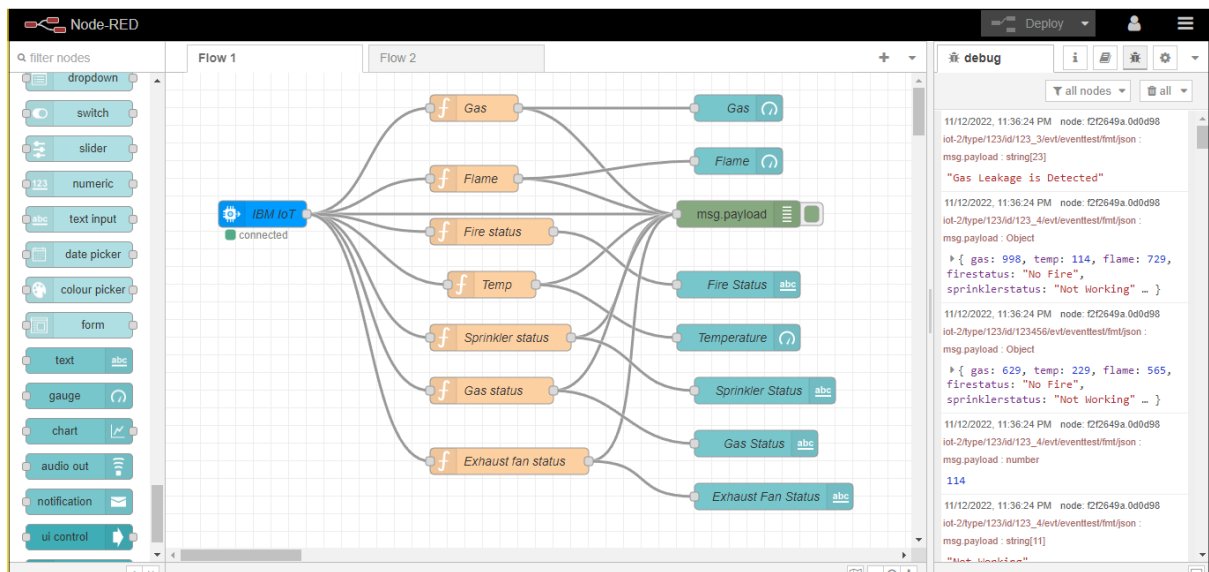
Go to your Node-RED flow editor

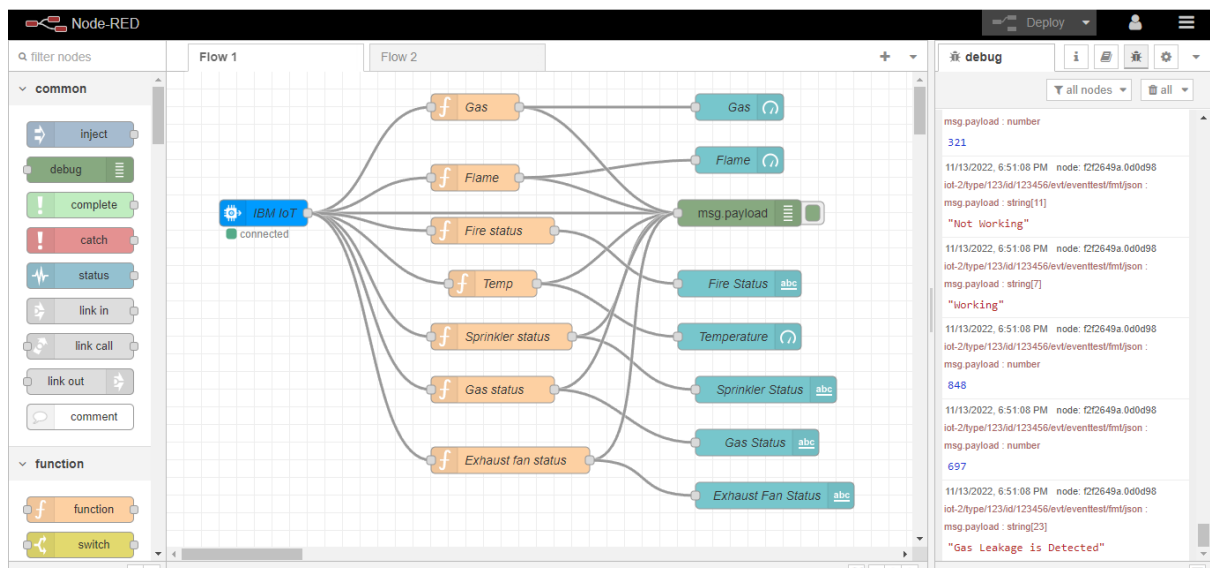
[Learn how to customise Node-RED](#)

WEB APPLICATION USING NODE-RED :

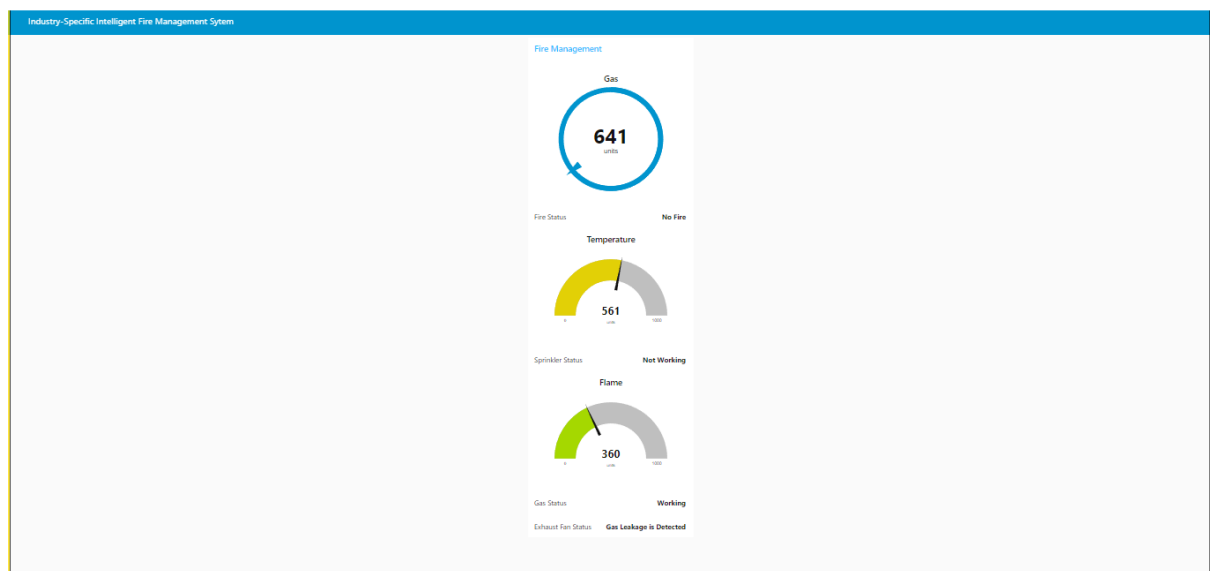


OUTPUT:





NODE-RED DASHBOARD STATUS:



Successfully get the status for Flame, Fire, Gas ,
Temperture in Node-red Dashboard Using Node-Red
Platform.