Delivery Sprint-4

Industry-Specific Intelligent Fire Management System

TEAM ID: PNT2022TMID45189

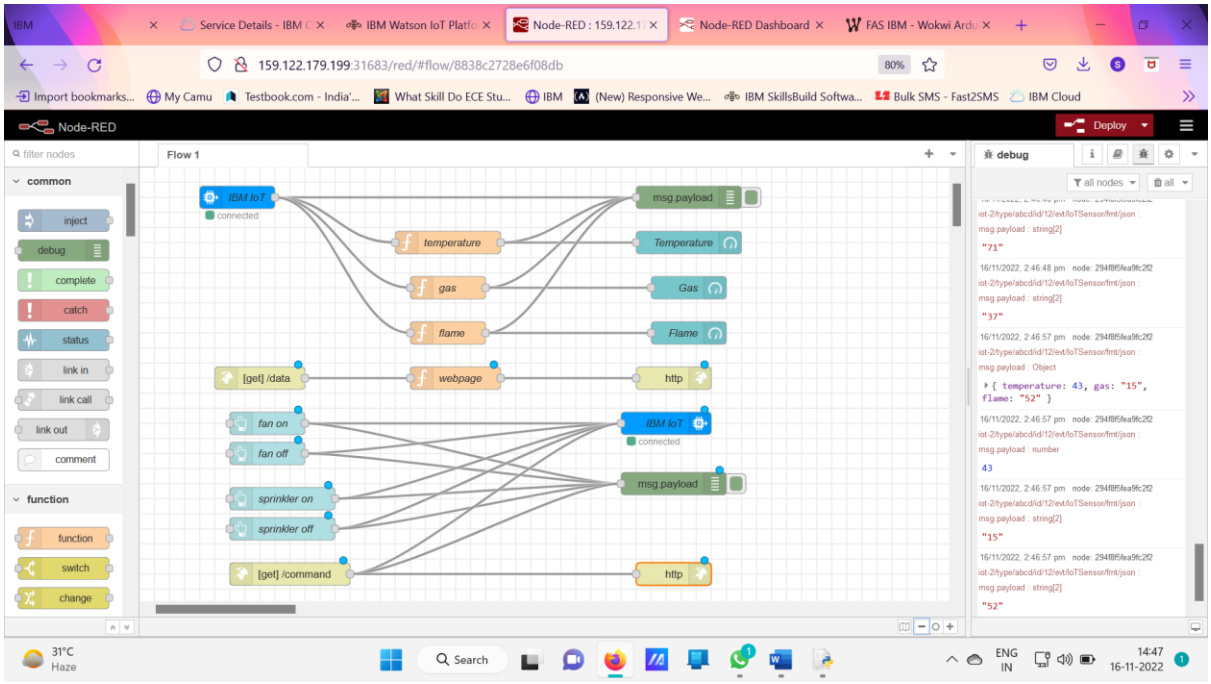
Create a smart fire management system that includes a Gas sensor, Flame sensor and temperature sensors to detect any changes in the environment. Based on the temperature readings and if any Gases are present the exhaust fans are powered ON. If any flame is detected the sprinklers will be switched on automatically. Emergency alerts are notified to the authorities and Fire station.

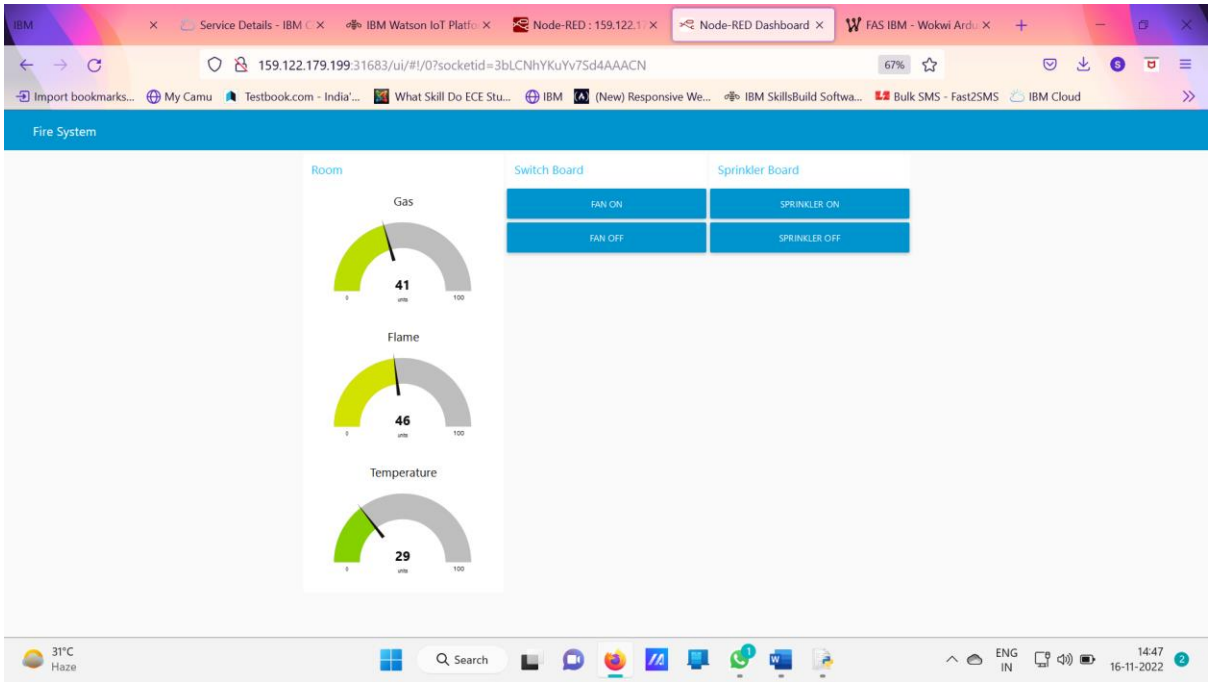
Sprint-4

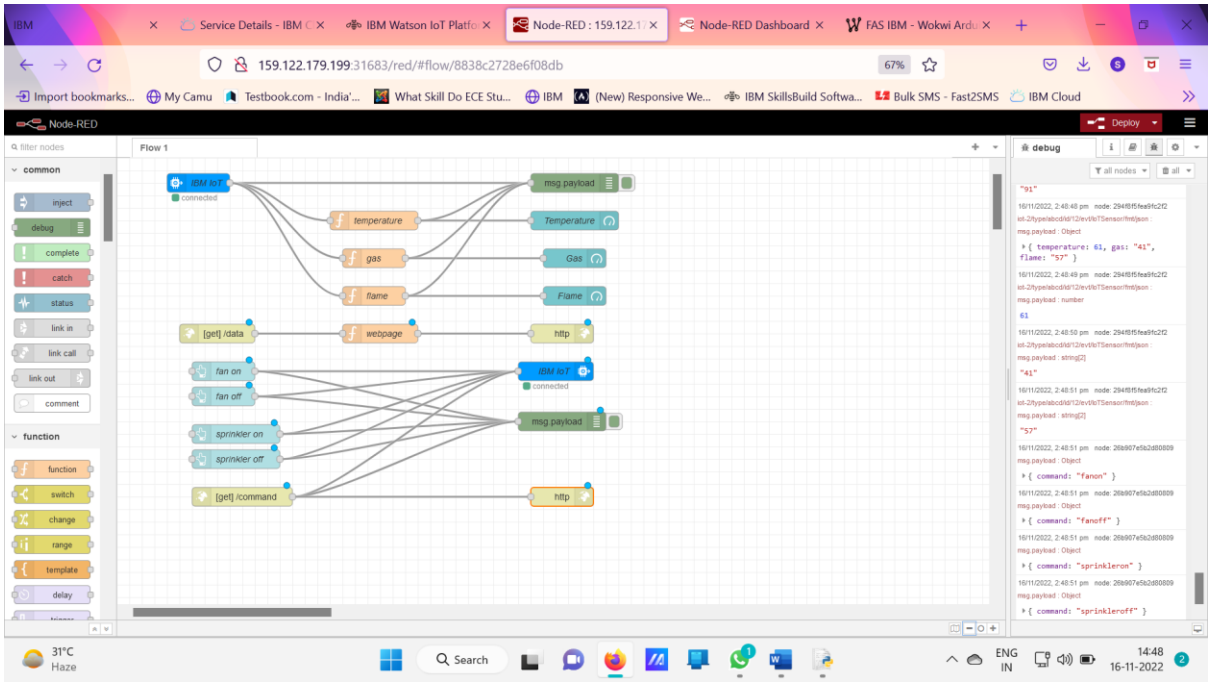
Web UI (to make the user interact with the software) / Run a simulation using the wokwi online platform

PROCESS

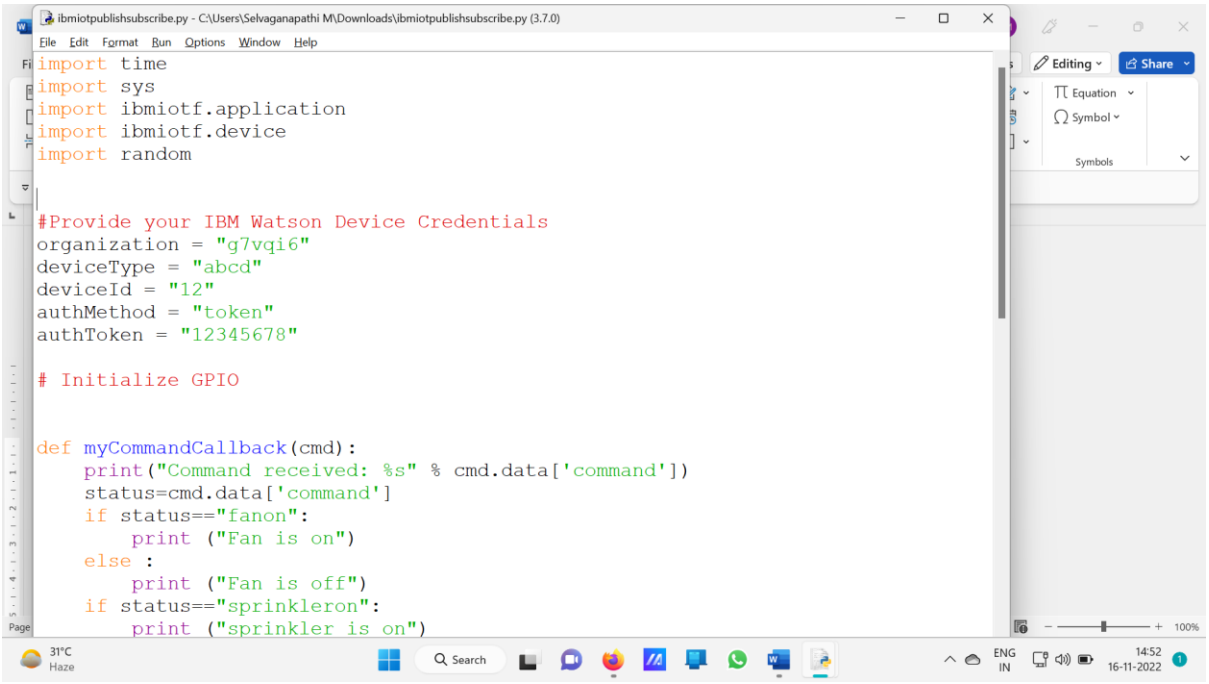
Using Node-Red for the Web UI Process



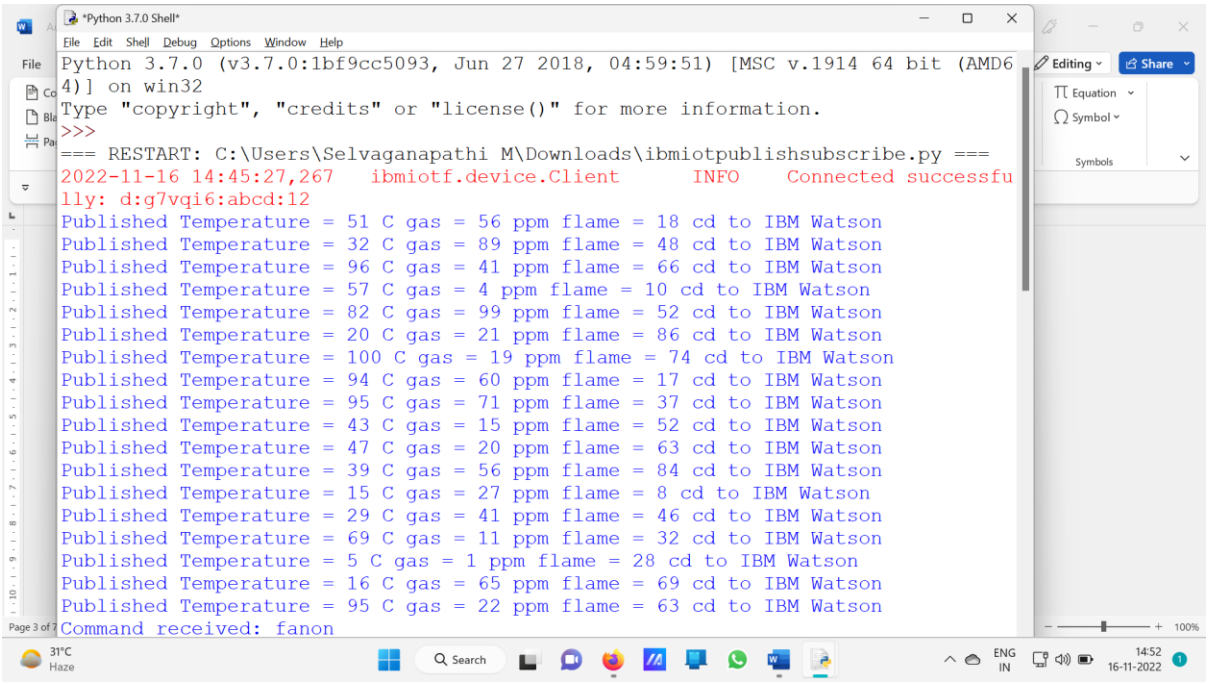

Fire System
Room
Gas
41
Flame
46
Temperature
29
Switch Board
FAN ON
FAN OFF
Sprinkler Board
SPRINKLER ON
SPRINKLER OFF


Node-RED
Flow 1
IBM IoT
connected
temperature
gas
flame
webpage
[get] /data
[get] /command
fan on
fan off
sprinkler on
sprinkler off
msg.payload
Temperature
Gas
Flame
http
debug

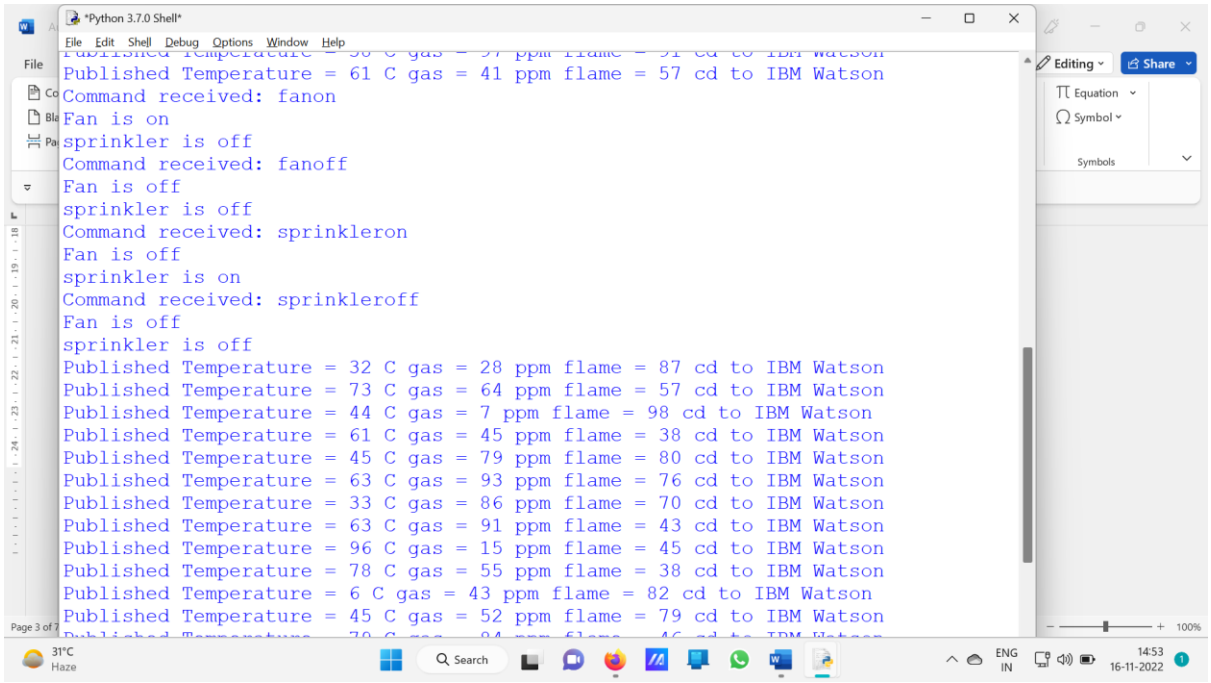
PROGRAM



```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random


#Provide your IBM Watson Device Credentials
organization = "g7vqi6"
deviceType = "abcd"
deviceId = "12"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO


def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="fanon":
        print ("Fan is on")
    else :
        print ("Fan is off")
    if status=="sprinkleron":
        print ("sprinkler is on")
```



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: C:\Users\Selvaganapathi M\Downloads\ibmiotpublishsubscribe.py ===
2022-11-16 14:45:27,267   ibmiotf.device.Client      INFO    Connected successfully: d:g7vqi6:abcd:12
Published Temperature = 51 C gas = 56 ppm flame = 18 cd to IBM Watson
Published Temperature = 32 C gas = 89 ppm flame = 48 cd to IBM Watson
Published Temperature = 96 C gas = 41 ppm flame = 66 cd to IBM Watson
Published Temperature = 57 C gas = 4 ppm flame = 10 cd to IBM Watson
Published Temperature = 82 C gas = 99 ppm flame = 52 cd to IBM Watson
Published Temperature = 20 C gas = 21 ppm flame = 86 cd to IBM Watson
Published Temperature = 100 C gas = 19 ppm flame = 74 cd to IBM Watson
Published Temperature = 94 C gas = 60 ppm flame = 17 cd to IBM Watson
Published Temperature = 95 C gas = 71 ppm flame = 37 cd to IBM Watson
Published Temperature = 43 C gas = 15 ppm flame = 52 cd to IBM Watson
Published Temperature = 47 C gas = 20 ppm flame = 63 cd to IBM Watson
Published Temperature = 39 C gas = 56 ppm flame = 84 cd to IBM Watson
Published Temperature = 15 C gas = 27 ppm flame = 8 cd to IBM Watson
Published Temperature = 29 C gas = 41 ppm flame = 46 cd to IBM Watson
Published Temperature = 69 C gas = 11 ppm flame = 32 cd to IBM Watson
Published Temperature = 5 C gas = 1 ppm flame = 28 cd to IBM Watson
Published Temperature = 16 C gas = 65 ppm flame = 69 cd to IBM Watson
Published Temperature = 95 C gas = 22 ppm flame = 63 cd to IBM Watson
Command received: fanon
```


Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Published Temperature = 61 C gas = 41 ppm flame = 57 cd to IBM Watson
Command received: fanon
Fan is on
sprinkler is off
Command received: fanoff
Fan is off
sprinkler is off
Command received: sprinkleron
Fan is off
sprinkler is on
Command received: sprinkleroff
Fan is off
sprinkler is off
Published Temperature = 32 C gas = 28 ppm flame = 87 cd to IBM Watson
Published Temperature = 73 C gas = 64 ppm flame = 57 cd to IBM Watson
Published Temperature = 44 C gas = 7 ppm flame = 98 cd to IBM Watson
Published Temperature = 61 C gas = 45 ppm flame = 38 cd to IBM Watson
Published Temperature = 45 C gas = 79 ppm flame = 80 cd to IBM Watson
Published Temperature = 63 C gas = 93 ppm flame = 76 cd to IBM Watson
Published Temperature = 33 C gas = 86 ppm flame = 70 cd to IBM Watson
Published Temperature = 63 C gas = 91 ppm flame = 43 cd to IBM Watson
Published Temperature = 96 C gas = 15 ppm flame = 45 cd to IBM Watson
Published Temperature = 78 C gas = 55 ppm flame = 38 cd to IBM Watson
Published Temperature = 6 C gas = 43 ppm flame = 82 cd to IBM Watson
Published Temperature = 45 C gas = 52 ppm flame = 79 cd to IBM Watson
Editing
Share
Equation
Symbol
Symbols
Page 3 of 7
100%
31°C
Haze
Search
ENG
IN
14:53
16-11-2022

CODE

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random



#Provide your IBM Watson Device Credentials
organization = "g7vqi6"
deviceType = "abcd"
deviceId = "12"
authMethod = "token"
authToken = "12345678"

# Initialize GPIO



def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="fanon":
        print ("Fan is on")
    else :
        print ("Fan is off")
    if status=="sprinkleron":
        print ("sprinkler is on")
    else :
        print ("sprinkler is off")
    #print(cmd)
```

```
try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}
        deviceCli = ibmiotf.device.Client(deviceOptions)
        #.....................................


except Exception as e:
        print("Caught exception connecting device: %s" % str(e))
        sys.exit()


# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times
deviceCli.connect()


while True:
    #Get Sensor Data from DHT11


    temperature=random.randint(0,100)
    gas=str(random.randint(0,100))
    flame=str(random.randint(0,100))


    data = { 'temperature' : temperature, 'gas': gas, 'flame' : flame }
    #print data
    def myOnPublishCallback():
       print ("Published Temperature = %s C" % temperature, "gas = %s ppm" % gas,"flame = %s cd" % flame, "to IBM Watson")


    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
```

```
        if not success:
            print("Not connected to IoTF")
        time.sleep(10)

        deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

WOKWI

Create your project in the online platform of the WOKWI and execute it using the IBM Credential.

CODE

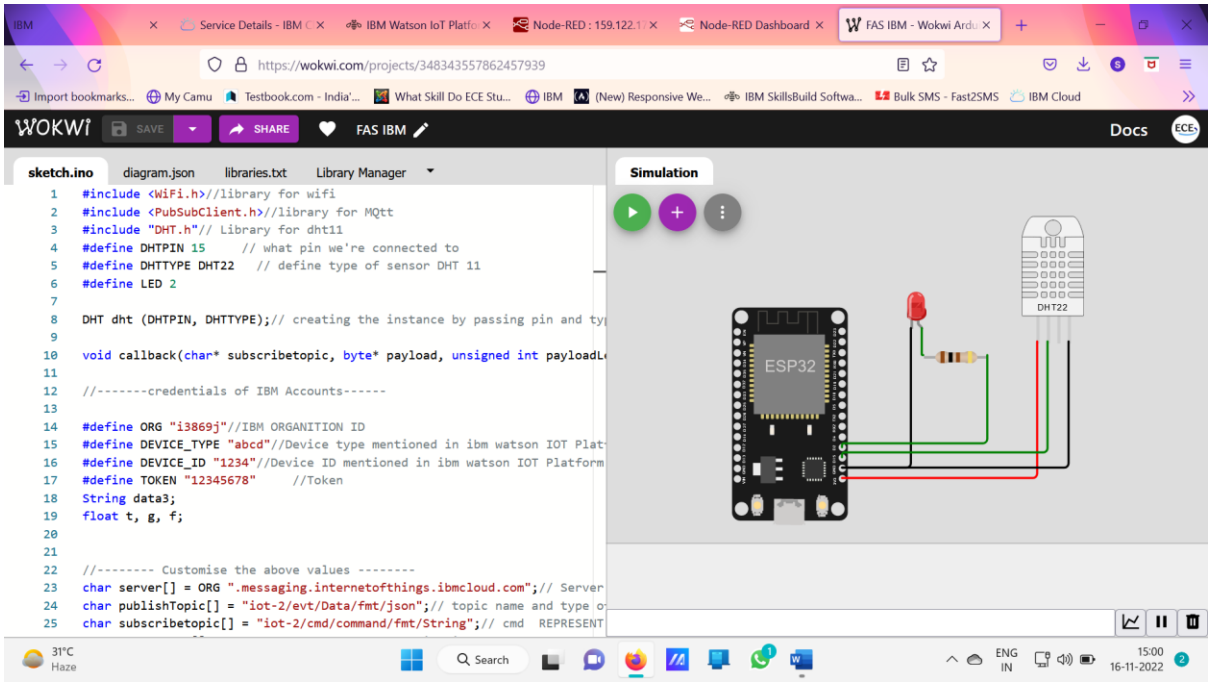
```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 15     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define LED 2

DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and typr of
dht connected

void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);

//-------credentials of IBM Accounts------

#define ORG "i3869j"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"     //Token
String data3;
float t, g, f;


//-------- 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);
```

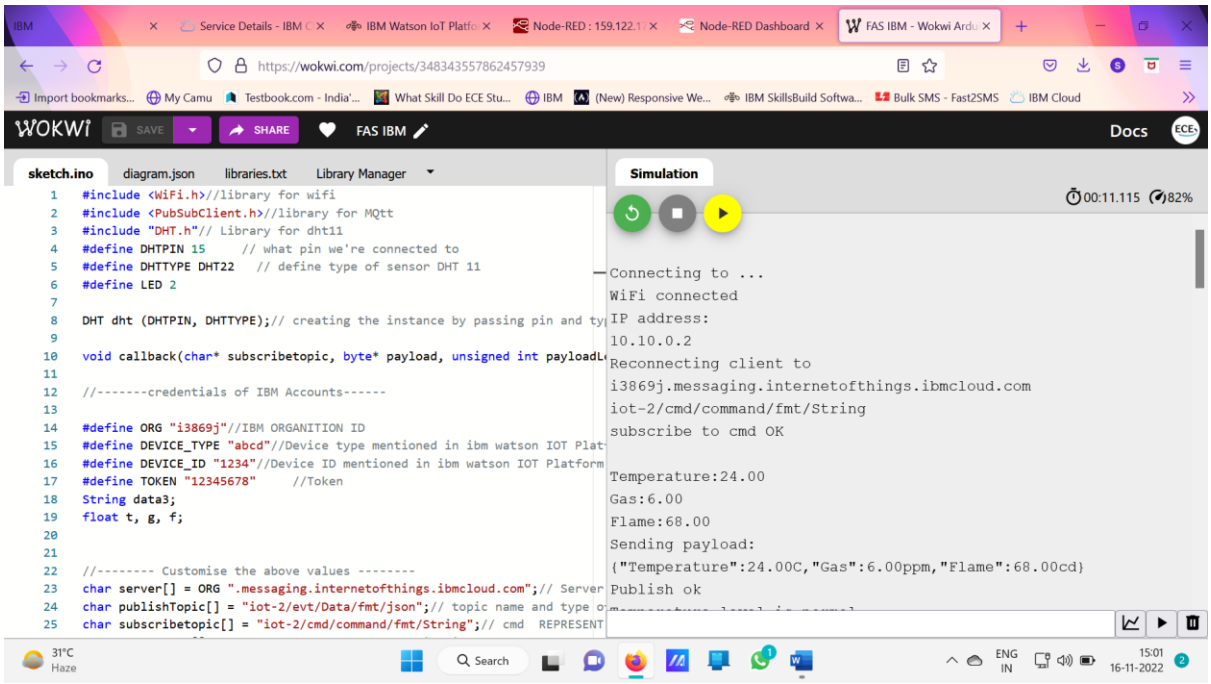
```
  dht.begin();
  pinMode(LED,OUTPUT);
  delay(10);
  Serial.println();
  randomSeed(analogRead(0));
  wificonnect();
  mqttconnect();
}

void loop()// Recursive Function
{

  t = dht.readTemperature();
  g = random(80);
  f = random(80);
  Serial.print("Temperature:");
  Serial.println(t);
  Serial.print("Gas:");
  Serial.println(g);
  Serial.print("Flame:");
  Serial.println(f);

  PublishData(t, g, f);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
  {
    if(t>=60){
      Serial.print("Fan is ON Automatically When HIGH Temperature in
Degree(Temp>=60C) ");//C-Celsius
      Serial.print("\n");
    }
    else{
      Serial.println("Temperature level is normal");
      }
    }
    {
    if(g>=40){
      Serial.print("Fan is ON Automatically When HIGH Gas in (Gas>=40ppm)
");//ppm-Patrs Per Million
      Serial.print("\n");
    }
    else{
      Serial.println("Gas level is normal");
      }
    }  {
    if(f>=30){
```

```
      Serial.print("Sprinkler is ON Automatically When detected the flame
limit reached the value>=30cd");//cd-candela
      Serial.print("\n");
    }
    else{
      Serial.println("No flame is Detected");
      }
    }
}



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

void PublishData(float temp, float gas, float flame) {
  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 += "C," "\"Gas\":";
  payload += gas;
  payload += "ppm," "\"Flame\":";
  payload += flame;
  payload += "cd}";


  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="";
}
```





Link:

https://wokwi.com/projects/348343557862457939

Note:

(In this project the used credentials are given by the tutorial instructors because of some issues in the other credential id running the project in wokwi platform)