SOURCE CODE

Wokwi :

```
#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 22

#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 "x6rbso"//IBM ORGANITION ID

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

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

#define TOKEN "Q&hrS52r0@Qs5)xh@+"     //Token

String data3;

float h;

//-------- 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

{

  //h = dht.readHumidity();

  t = dht.readTemperature();

  Serial.print("Temperature:");

  Serial.println(t);

 // Serial.print("Humidity:");

  //Serial.println(h);

  PublishData(t);
```

```
  delay(6000);
  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(1000);
    }

     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(1000);
    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=="ALERT!!! Your industry got fired")
  {
  Serial.println(data3);
  pinMode(LED,HIGH);
  tone(LED,67);
  delay(20000);
  }
  else
```

```
 {
Serial.println(data3);
pinMode(LED,LOW);
noTone(LED);
 }
data3="";
}
```

Python code :

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

#Provide your IBM Watson Device Credentials
organization = "x6rbso" # repalce it with organization ID
deviceType = "project" #replace it with device type
deviceId = "projectid" #repalce with device id
authMethod = "token"
authToken = "Q&hrS52r0@Qs5)xh@+"#repalce with token

def myCommandCallback(cmd):
        print("Command received: %s" % cmd.data)
        if cmd.data['command']=='Sprinkler On':
                print("sprinkler On")
        elif cmd.data['command'] == 'Sprinkler Off':
```

```
        print("sprinkler Off")


def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data)
    if cmd.data['command']=='ExhaustFan On':
        print("ExhaustFan On")




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()


deviceCli.connect()


while True:
    F=155;
    G=255;


    #Send Temperature & Humidity to IBM Watson
```

```
        data = { 'Flame' : F,'Gas': G }

        #print data

        def myOnPublishCallback():

            print ("Published Flame = %s C" % F, "Gas = %s %%" % G,  "to IBM Watson")


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

        if not success:

            print("Not connected to IoTF")

        time.sleep(60)


        deviceCli.commandCallback = myCommandCallback


# Disconnect the device and application from the cloud

deviceCli.disconnect()
```