

DEVELOP A PYTHON SCRIPT

PUBLISH DATA TO IBM CLOUD

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQTT
#define EchoPIN 4      // what pin we're connected to
#define TrigPIN 2
#define LED 5
//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 "2piqlm"//IBM ORGANIZATION ID
#define DEVICE_TYPE "Vijay"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "12345"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"      //Token
String data3;

//----- 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
float dist,dur;
String data;
//-----
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);
```

```

    pinMode(TrigPIN, OUTPUT);
    digitalWrite(TrigPIN, LOW);
    pinMode(EchoPIN, INPUT);
    pinMode(LED, OUTPUT);
    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

void loop()// Recursive Function
{

    digitalWrite(TrigPIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TrigPIN, LOW);

    dur = pulseIn(EchoPIN, HIGH);

    dist= dur *0.034 / 2;
    if(dist<100)
    {
        data="alert";
        digitalWrite(LED, HIGH);
    }
    else{
        data="safe";
        digitalWrite(LED, LOW);
    }

    PublishData(dist);
    delay(1000);
    if (!client.loop()) {
        mqttconnect();
    }
}

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

void PublishData(float dist) {
    mqttconnect();//function call for connecting to ibm
    /*
        creating the String in in form JSon to update the data to ibm cloud
    */

```

```

String payload = "{\"distance\": ";
payload += dist;
payload += ", \"msg\": \"";
payload += data;
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)
{
}

```

DATA PUBLISH TO IBM CLOUD

The screenshot displays 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 shows a device named '12345' with a status of 'Connected' and a last update time of 'Nov 7, 2022 3:17 PM'. Below this, the 'Recent Events' tab is selected, showing a table of live data streams. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. It lists five data events, each with a JSON payload containing distance and message status. A notification at the bottom right indicates '0 Simulations running'.

Event	Value	Format	Last Received
Data	{"distance":399.96,"msg":"safe"}	json	a few seconds ago
Data	{"distance":399.92,"msg":"safe"}	json	a few seconds ago
Data	{"distance":399.94,"msg":"safe"}	json	a few seconds ago
Data	{"distance":399.92,"msg":"safe"}	json	a few seconds ago
Data	{"distance":399.96,"msg":"safe"}	json	a few seconds ago

0 Simulations running