

Assignment -4
Python
Programming

Assignment Date	28 OCTOBER 2022
Student Name	ABISHEK RONJAN RC
Student Roll Number	113019106003
Maximum Marks	2 Marks

Question:

Write code and connections in wokwi for the ultrasonic sensor.Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display inthe device recent events.Upload document with wokwi share link and images of IBM cloud.

Program:

```
#include <WiFi.h>/library for wifi
#include <PubSubClient.h>/library for MQTT
#define ECHO_PIN 2
#define TRIG_PIN 4
#define LED 5

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

#define ORG "4gh14s"/IBM ORGANITION ID
#define DEVICE_TYPE "ESP32"/Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234"/Device ID mentioned in ibm watson IOT Platform
#define TOKEN "5xp6Zc74hThvC!qyOY" //Token

//----- 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/test/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,wifiClient); //calling the predefined client
id by passing parameter like server id,portand wificredential
void setup()// configueing the ESP32
{
    Serial.begin(115200);
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    pinMode(LED,OUTPUT);
    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

float readDistanceCM() {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    int duration = pulseIn(ECHO_PIN, HIGH);
    return duration * 0.034 / 2;
}

void loop()// Recursive Function
{
    float distance = readDistanceCM();
    bool isNearby = distance < 100;
    digitalWrite(LED, isNearby);
    Serial.print("Measured distance: ");
    Serial.println(distance);
    delay(100);
    if (isNearby == 1){
        PublishData(distance);
    }
    delay(1000);
    if (!client.loop()) {
        mqttconnect();
    }
}

/*
.....retrieving to
Cloud....*/
void PublishData(float distance) {

```

```

mqttconnect();//function call for connecting to ibm
/*
    creating the String in in form JSON to update the data to ibm cloud
*/
String payload = "{\"Alert\":\"\"}";
payload += distance;
payload += " is less than 100cms\"";
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");
    }
}

```

Picture:

The screenshot displays a development environment for an ESP32-based project. On the left, the code editor shows the following C++ code:

```

1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #define ECHO_PIN 2
4 #define TRIG_PIN 4
5 #define LED 5
6
7
8 //-----credentials of IBM Accounts-----
9
10 #define ORG "4gh14s" //IBM ORGANIZATION ID
11 #define DEVICE_TYPE "ESP32" //Device type mentioned in Watson IOT Platform
12 #define DEVICE_ID "1234" //Device ID mentioned in Watson IOT Platform
13 #define TOKEN "5xp6Zc74hThvC1qyOY" //Token
14
15
16 //----- Customise the above values -----
17 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
18 char publishTopic[] = "iot-2/evt/data/fmt/json"; // topic name and type of event performed
19 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type AND COMMAND
20 char authMethod[] = "use-token-auth"; // authentication method
21 char token[] = TOKEN;
22 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
23
24
25 //-----
26 WiFiClient wifiClient; // creating the instance for wifi client
27 PubSubClient client(server, 1883, wifiClient); //calling the predefined client id by passing
28 void setup() // configuring the ESP32
29 {
30 {
31 Serial.begin(115200);
32 pinMode(TRIG_PIN, OUTPUT);

```

The central part of the interface shows a schematic diagram of the hardware setup. An ESP32 module is connected to a breadboard. On the breadboard, there is an ultrasonic distance sensor (HC-SR04) connected to pins 4 (TRIG) and 2 (ECHO). A red LED is also connected to digital pin 5. The right side of the interface is a terminal window showing the output of the code execution:

```

Measured distance: 70.96
Sending payload: {"Alert":"70.96 is less than 100cms"}
Publish ok
Measured distance: 67.97
Sending payload: {"Alert":"67.97 is less than 100cms"}
Publish ok
Measured distance: 67.98

```

Link:

<https://wokwi.com/projects/347766595898049107>

Cloud Output:

The screenshot shows the IBM Watson IoT Platform interface. On the left is a sidebar with various icons. The main area has a header with 'IBM Watson IoT Platform' and user information. Below the header is a search bar and a 'Add Device' button. A table lists a single device: '1234' (Device ID), 'Connected' (Status), 'ESP32' (Device Type), 'Device' (Class ID), and 'Nov 8, 2022 3:25 PM' (Date Added). Below the table are tabs for 'Identity', 'Device Information', 'Recent Events' (which is selected), 'State', and 'Logs'. A message says 'The recent events listed show the live stream of data that is coming and going from this device.' Below this is a table of recent events with columns 'Event', 'Value', 'Format', and 'Last Received'. All entries show 'Data' as the event, a JSON value indicating an alert, 'json' as the format, and 'a few seconds ago' as the last received time. At the bottom, there are pagination controls and a note about simulations.

Event	Value	Format	Last Received
Data	{"Alert":"81.96 is less than 100cms"}	json	a few seconds ago
Data	{"Alert":"81.96 is less than 100cms"}	json	a few seconds ago
Data	{"Alert":"81.96 is less than 100cms"}	json	a few seconds ago
Data	{"Alert":"81.96 is less than 100cms"}	json	a few seconds ago
Data	{"Alert":"81.96 is less than 100cms"}	json	a few seconds ago

Items per page: 50 | 1–1 of 1 item 0 Simulations running

