

Assignment -4

Python
Programming

Assignment Date	28 OCTOBER 2022
Student Name	BHARATH O
Student Roll Number	113019106007
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 in the 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!qy0Y" //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()// configuring 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");
  }
}
}

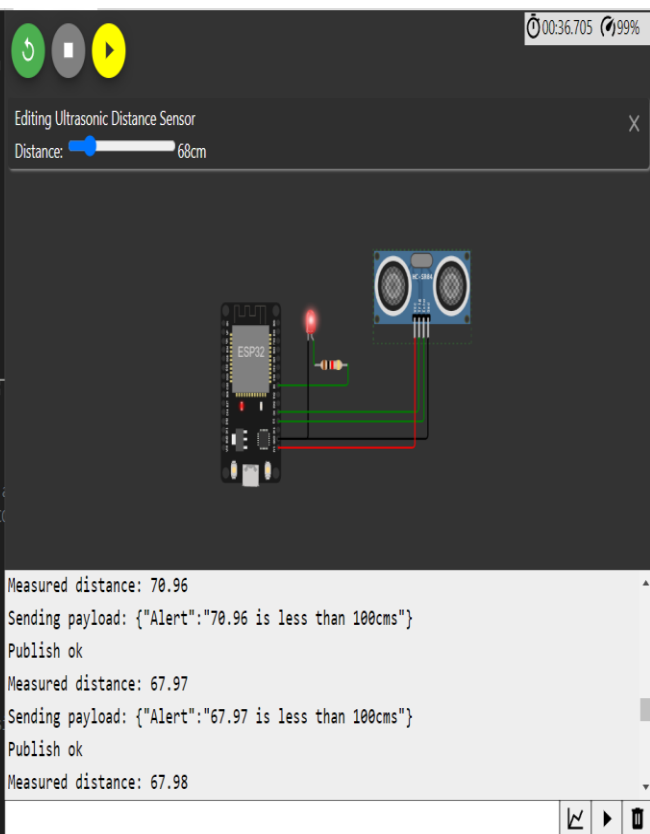
```

Picture:

```

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 ORGANITION ID
11 #define DEVICE_TYPE "ESP32" //Device type mentioned in ibm watson IOT Platform
12 #define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
13 #define TOKEN "5xp6Zc74hThvClqy0Y" //Token
14
15
16
17 //----- Customise the above values -----
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
19 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform
20 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type AND CC
21 char authMethod[] = "use-token-auth"; // authentication method
22 char token[] = TOKEN;
23 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
24
25
26 //-----
27 WiFiClient wifiClient; // creating the instance for wificlient
28 PubSubClient client(server, 1883, wifiClient); //calling the predefined client id by pass
29 void setup() // configuring the ESP32
30 {
31   Serial.begin(115200);
32   pinMode(TRIG_PIN, OUTPUT);

```



The screenshot shows the Arduino IDE interface. The left pane contains the C++ code for an ESP32 microcontroller. The right pane shows the serial monitor output, which includes distance measurements from an ultrasonic sensor and JSON payloads being sent to the IBM Watson IoT Platform. The code defines various constants like pins, credentials, and topics, and includes functions for initializing the WiFi client and subscribing to MQTT topics. The serial monitor shows the device successfully connecting to the network and sending data.

Link:

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

Cloud Output:

IBM Watson IoT Platform

bb5744652@gmail.com
ID: 4gh14s

Browse

Action

Device Types

Interfaces

Add Device

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
1234	Connected	ESP32	Device	Nov 8, 2022 3:25 PM	

Identity

Device Information

Recent Events

State

Logs

The recent events listed show the live stream of data that is coming and going from this device.

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

