

Assignment -4
ESP32 Programming with IBM Cloud

Assignment Date	1 November 2022
Student Name	R.Sakthipriya
Student Roll Number	621719106040
Maximum Marks	2 Marks

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100cms send "alert" to ibm cloud and display in device recent events.

Upload document with wokwi share link and images of ibm cloud.

Solution:

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT

#define ECHO_GPIO 12
#define TRIGGER_GPIO 13
#define MAX_DISTANCE_CM 100 // Maximum of 5 meters
#include "Ultrasonic.h"

Ultrasonic ultrasonic(13, 12);
int distance;

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

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

#define ORG "8zenx2" //IBM ORGANITION ID
#define DEVICE_TYPE "raspberrypi" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "dhaya29" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "Dhaya@29" //Token
String data3;
float h, t;

//----- 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, port and wificredential

void setup() // configureing the ESP32
{
```

```

Serial.begin(115200);
delay(10);
Serial.println();
wificonnect();
mqttconnect();
}

void loop()// Recursive Function
{

    distance = ultrasonic.read(CM);
    if(distance < 100){
        Serial.print("Distance in CM: ");
        Serial.println(distance);
        PublishData(distance);
        delay(1000);
        if (!client.loop()) {
            mqttconnect();
        }

    }

    delay(1000);

}

/*.....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 = "{\"Alert Distance\":\"";
    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(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);
    }
    else
    {
        Serial.println(data3);
    }
    data3="";
}

```

WOKWI

SAVE

SHARE

Docs

sketch.ino

diagram.json

libraries.txt

Ultrasonic.h

Ultrasonic.cpp

Library Manager

```

1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3
4 #define ECHO_GPIO 12
5 #define TRIGGER_GPIO 13
6 #define MAX_DISTANCE_CM 100 // Maximum of 5 meters
7 #include "Ultrasonic.h"
8
9 Ultrasonic ultrasonic(13, 12);
10 int distance;
11
12 void callback(char* subscribtopic, byte* payload, unsigned int payloadLength);
13
14 //-----credentials of IBM Accounts-----
15
16 #define ORG "bxd09" //IBM ORGANIZATION ID
17 #define DEVICE_TYPE "ESP32" //Device type mentioned in ibm watson IOT Platform
18 #define DEVICE_ID "Assign4" //Device ID mentioned in ibm watson IOT Platform
19 #define TOKEN "45625689713" //Token
20 String data3;
21 float h, t;
22
23 //----- Customise the above values -----
24 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
25 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform
26 char subscribtopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND
27 char authMethod[] = "use-token-auth"; // authentication method
28 char token[] = TOKEN;
29 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
30
31 //-----
32
33 //-----
34 WiFiClient wificlient; // creating the instance for wificlient
35 PubSubClient client(server, 1883, callback ,wificlient); //calling the predefined client

```

Simulation

01:29.787

98%

Publish ok

Distance in CM: 28

Sending payload: {"Alert Distance":28.00}

Publish ok

Distance in CM: 28

Sending payload: {"Alert Distance":28.00}

Publish ok

Browse

Action

Device Types

Interfaces

Add Device

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 Distance":28}	json	a few seconds ago
Data	{"Alert Distance":28}	json	a few seconds ago
Data	{"Alert Distance":28}	json	a few seconds ago
Data	{"Alert Distance":28}	json	a few seconds ago
Data	{"Alert Distance":28}	json	a few seconds ago

Items per page 50 | 1-2 of 2 items

1 of 1 page

1

0 Simulations running

