

Assignment -4
ESP32 Programming with IBM
Cloud

Assignment Date	29 October 2022
Student Name	ABIMANYU M
Student Roll Number	820419106002
Maximum Marks	2 Marks

Question-1:

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100cm 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* subscribtopic, byte* payload, unsigned int payloadLength);

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

#define ORG "uo60re" //IBM ORGANITION ID
#define DEVICE_TYPE "AQUAMEN1" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "AQUAMEN1_1" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "AQUA1234" //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 subscribtopic[] = "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);
  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="";  
}
```

```

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* topic, byte* payload, unsigned int payloadLength);
13
14 //-----credentials of IBM Accounts-----
15
16 #define ORG "us60re" //IBM ORGANIZATION ID
17 #define DEVICE_TYPE "AQUAMEN1" //Device type mentioned in ibm watson IOT Platform
18 #define DEVICE_ID "AQUAMEN1_1" //Device ID mentioned in ibm watson IOT Platform
19 #define TOKEN "AQUA1234" //Token
20 String data3;
21 float h, t;
22
23 //----- Customise the above values -----
24
25 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
26 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform
27 char subscribTopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND
28 char authMethod[] = "use-token-auth"; // authentication method
29 char token[] = TOKEN;
30 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
31
32 //-----
33
34 WiFiClient wifiClient; // creating the instance for wifiClient
35 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client
36
37 void setup() // configuring the ESP32
38 {

```

Simulation console output:

```

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

```

Device ID: AQUAMEN1_1, Status: Disconnected, Device Type: AQUAMEN1, Class ID: Device, Date Added: Nov 3, 2022 8:33 PM, Added By: 820419106034@smartinternz.com

Event	Value	Format	Last Received
data	{"Alert Distance":28}	json	a few seconds ago
data	{"Alert Distance":28}	json	a minute ago
data	{"Alert Distance":28}	json	2 minutes ago

1 Simulation running

Wokwi share link:
<https://wokwi.com/projects/347322997381530195>

