

Assignment -4
Wokwi & IBM Cloud

Assignment Date	28 October 2022
Student Name	Nandha kumar P
Student Roll Number	732219CS070
Maximum Marks	2 Marks

Question-1:

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

Solution:

Code:

```
#include <WiFi.h>
#include <PubSubClient.h>
WiFiClient wifiClient;
String data3;
#define ORG "it4o0t"
#define DEVICE_TYPE "ultras_sensor"
#define DEVICE_ID "us_sensor"
#define TOKEN "123654987"
#define speed 0.034
#define led 14
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/manimd/fmt/json";
char topic[] = "iot-2/cmd/led/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
PubSubClient client(server, 1883, wifiClient);

const int trigpin=5;
const int echopin=18;
String command;
String data="";

long duration;
float dist;

void setup()
{
  Serial.begin(115200);
  pinMode(led, OUTPUT);
```

```

    pinMode(trigpin, OUTPUT);
    pinMode(echopin, INPUT);
    wifiConnect();
    mqttConnect();
}

void loop() {
    bool isNearby = dist < 100;
    digitalWrite(led, isNearby);

    publishData();
    delay(500);

    if (!client.loop()) {
        mqttConnect();
    }
}

void wifiConnect() {
    Serial.print("Connecting to "); Serial.print("Wifi");
    WiFi.begin("Wokwi-GUEST", "", 6);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.print("WiFi connected, IP address: ");
    Serial.println(WiFi.localIP());
}

void mqttConnect() {
    if (!client.connected()) {
        Serial.print("Reconnecting MQTT client to "); Serial.println(server);
        while (!client.connect(clientId, authMethod, token)) {
            Serial.print(".");
            delay(500);
        }
        initManagedDevice();
        Serial.println();
    }
}

void initManagedDevice() {
    if (client.subscribe(topic)) {
        // Serial.println(client.subscribe(topic));
        Serial.println("IBM subscribe to cmd OK");
    } else {
        Serial.println("subscribe to cmd FAILED");
    }
}

void publishData()
{
    digitalWrite(trigpin, LOW);

```

```

digitalWrite(trigpin,HIGH);
delayMicroseconds(10);
digitalWrite(trigpin,LOW);
duration=pulseIn(echopin,HIGH);
dist=duration*speed/2;
if(dist<100){
    String payload = "{\"Alert Distance\":\"";
    payload += dist;
    payload += "\"}";

    Serial.print("\n");
    Serial.print("Sending payload: ");
    Serial.println(payload);
    if (client.publish(publishTopic, (char*) payload.c_str())) {
        Serial.println("Publish OK");
    }
}

if(dist>100){
    String payload = "{\"Distance\":\"";
    payload += dist;
    payload += "\"}";

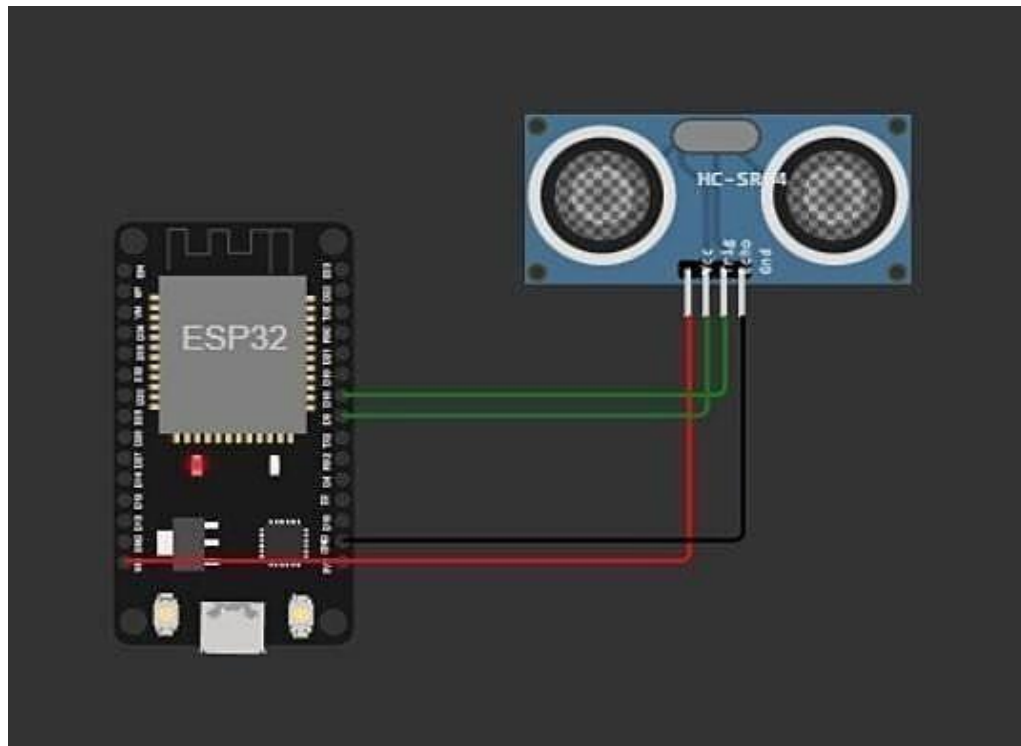
    Serial.print("\n");
    Serial.print("Sending payload: ");
    Serial.println(payload);
    if(client.publish(publishTopic, (char*) payload.c_str())) {
        Serial.println("Publish OK");
    }else {
        Serial.println("Publish FAILED");
    }
}

}

}

```

Connections:



Output:(wokwi):

A screenshot of the Wokwi web interface. The left pane shows the sketch code for an ESP32, which includes headers for WiFi and PubSubClient, defines for device type, ID, token, and speed, and a loop that publishes distance data to an MQTT topic. The right pane shows the simulation output, which displays a series of messages: "Sending payload: {'Alert Distance':99.98}" followed by "Publish OK". The interface also includes a "Simulation" tab with play, stop, and pause buttons, and a status bar at the bottom showing the time and battery level.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 WiFiClient wifiClient;
4 String data3;
5 #define ORG "IoT08t"
6 #define DEVICE_TYPE "ultras_sensor"
7 #define DEVICE_ID "us_sensor"
8 #define TOKEN "123654987"
9 #define speed 0.034
10 #define led 14
11 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
12 char publishTopic[] = "iot-2/evt/manimd/fmt/json";
13 char topic[] = "iot-2/cmd/led/fmt/String";
14 char authMethod[] = "use-token-auth";
15 char token[] = TOKEN;
16 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
17 PubSubClient client(server, 1883, wifiClient);
18
19
20
21 const int trigpin=5;
22 const int echopin=18;
23 String command;
24 String data="";
25
26 long duration;
27 float dist;
28
```

Simulation

Sending payload: {'Alert Distance':99.98}
Publish OK

Sending payload: {'Alert Distance':99.98}
Publish OK

Sending payload: {'Alert Distance':99.98}
Publish OK

Sending payload: {'Alert Distance':99.98}
Publish OK

Sending payload: {'Alert Distance':99.98}
Publish OK

Sending payload: {'Alert Distance':99.98}
Publish OK

Sending payload: {'Alert Distance':99.98}

Link: <https://wokwi.com/projects/347209761694941779>

Output:(IBM Cloud)

The screenshot displays the IBM Watson IoT Platform interface. At the top, a dark navigation bar shows the platform name and a user profile. Below this, a secondary navigation bar contains tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. The main content area features a table of events for a device named 'manimd'. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. Five rows of data are shown, with the fourth row highlighted. At the bottom of the table, there is a pagination control showing 'Items per page 50' and '1 of 1 item'. The left sidebar contains various icons for navigation, and the bottom of the screen shows a Windows taskbar with a search bar and system icons.

Event	Value	Format	Last Received
manimd	{"Alert Distance":99.94}	json	a few seconds ago
manimd	{"Alert Distance":99.98}	json	a few seconds ago
manimd	{"Alert Distance":99.98}	json	a few seconds ago
manimd	{"Alert Distance":99.98}	json	a few seconds ago
manimd	{"Alert Distance":54.94}	json	a few seconds ago