

## Assignment - 4

### IOT - SMART WASTE MANAGEMENT SYSTEM

Assignment Date	17 October 2022
Student Name	PRIYADHARSHINI R
Student Roll Number	49621915013
Maximum Marks	2 Marks

#### Question-1:

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

#### CODE 1 :

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribtopic, byte* payload, unsigned int
payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "x9oggs"//IBM ORGANITION ID
#define DEVICE_TYPE "iot_device"//Device type mentioned in ibm watson IOT
Platform#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT
Platform#define TOKEN "12345678" //Token
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribtopic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback ,wifiClient);
const int trigPin = 5;
const int echoPin = 18;
#define SOUND_SPEED 0.034
long duration;
float distance;
void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  wificonnect();
  mqttconnect();
}
void loop()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * SOUND_SPEED/2;
```

```

Serial.print("Distance (cm): ");
Serial.println(distance);
if(distance<100)
{
Serial.println("ALERT!!");
delay(1000);
PublishData(distance);
delay(1000);
if (!client.loop()) {
mqttconnect();
}
}
delay(1000);
}
void PublishData(float dist) {
mqttconnect();
String payload = "{\"Distance\": ";
payload += dist;
payload += ", \"ALERT!!\": \"\"Distance less than 100cms\"";
payload += "}";
Serial.print("Sending payload: ");
Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish ok");
} 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()
{
Serial.println();
Serial.print("Connecting to ");
WiFi.begin("Wokwi-GUEST", "", 6);
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);
data3="";
}
```

## Wokwi Link :

<https://wokwi.com/projects/347021585567187540>

## Output and Simulation :

The screenshot shows the Wokwi IDE interface. On the left, the 'sketch.ino' file is open, displaying C++ code for an ESP32 microcontroller. The code includes libraries for WiFi and PubSubClient, and defines variables for server, topic, device type, device ID, and token. It also defines pins for an ultrasonic sensor and a buzzer. The main loop triggers the sensor and publishes data to the cloud. On the right, the 'Simulation' window shows a visual representation of the ESP32 and an HC-SR04 ultrasonic sensor connected by wires. A status bar at the top right of the simulation window shows '00:10.276' and '99%'. Below the simulation, a console window displays the output of the program, showing 'ALERT!!' messages and the distance measured by the sensor (40.97 cm).

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int
4 payloadLength);
5 //-----credentials of IBM Accounts-----
6 #define ORG "cbsej1"//IBM ORGANITION ID
7 #define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT Platform
8 #define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
9 #define TOKEN "12345678" //Token
10 String data;
11 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
12 char publishTopic[] = "iot-2/evt/Data/fmt/json";
13 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
14 char authMethod[] = "use-token-auth";
15 char token[] = TOKEN;
16 char clientId[] = "d: " ORG ":" DEVICE_TYPE ":" DEVICE_ID;
17 WiFiClient wifiClient;
18 PubSubClient client(server, 1883, callback, wifiClient);
19 const int trigPin = 5;
20 const int echoPin = 18;
21 #define SOUND_SPEED 0.034
22 long duration;
23 float distance;
24 void setup() {
25   Serial.begin(115200);
26   pinMode(trigPin, OUTPUT);
27   pinMode(echoPin, INPUT);
28   wifiConnect();
29   mqttConnect();
30 }
31 void loop()
32 {
33   digitalWrite(trigPin, LOW);
34   delayMicroseconds(2);
35   digitalWrite(trigPin, HIGH);
36   delayMicroseconds(10);
37   digitalWrite(trigPin, LOW);
38   duration = pulseIn(echoPin, HIGH);
39   distance = duration * SOUND_SPEED / 2;
40   if (distance < 100) {
41     Serial.println("Distance: " + String(distance) + "cm");
42     data = "{\"Distance\":\"" + String(distance) + "\",\"ALERT!!\":\"Distance less than 100cms\"}";
43     client.publish(publishTopic, data);
44     Serial.println("Publish ok");
45   }
46 }
```

Simulation: Editing Ultrasonic Distance Sensor  
Distance: 41cm

ALERT!!  
Sending payload: {"Distance":93.94,"ALERT!!":"Distance less than 100cms"}  
Publish ok  
Distance (cm): 40.97  
ALERT!!  
Sending payload: {"Distance":40.97,"ALERT!!":"Distance less than 100cms"}  
Publish ok

Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display in the device recent events.

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. The main content area displays a table of devices. The first device, with ID '1234', is shown in detail. The 'Recent Events' tab is selected, showing a list of events. The events table has columns for 'Event', 'Value', 'Format', and 'Last Received'. The events listed are 'Data' with values like '({"Distance":93.94,"ALERT!!":"Distance less than ...}' and 'json' format, received 'a few seconds ago'.

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	Added By
1234	Connected	abcd	Device	Oct 10, 2022 9:38 PM		910619106034@smartinternz.com

Event	Value	Format	Last Received
Data	({"Distance":93.94,"ALERT!!":"Distance less than ...}	json	a few seconds ago
Data	({"Distance":93.96,"ALERT!!":"Distance less than ...}	json	a few seconds ago