

#### Assignment -4

Assignment Date	20 OCTOBER 2022
Student Name	HARIHARAN A
Student Roll Number	737819ECR050
Maximum Marks	2 Marks

#### Question-1:

Write code and connections in wowki for ultrasonic sensor.

Whenever distance is less than 100 cms send “alert” to IBM cloud and display in device recent events.

### Solution:

#### WOWKI LINK:

<https://wokwi.com/projects/347396071297647187>

```
#include <WiFi.h>
```

```
#include <PubSubClient.h>
```

```
#define TRIGGER 2
```

```
#define ECHO 15
```

```
#define sound 0.034
```

```
int distance;
```

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

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

```
#define ORG "2jk8e5"
```

```
#define DEVICE_TYPE "Gas_leakage_Detection_Device"
```

```
#define DEVICE_ID "Gas_Detector_1"
```

```
#define TOKEN "123456789"
```

```
String data3;
```

```
//----- Customise the above values -----
```

```
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
```

```
char publishTopic[] = "iot-2/evt/Data/fmt/json";
```

```
char subscribetopic[] = "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);
void setup()
{
  Serial.begin(115200);
  pinMode(TRIGGER, OUTPUT);
  pinMode(ECHO, INPUT);
  delay(10);
  Serial.println();
  Serial.println("Reconnecting client to wp72r7.messaging.internetofthings.ibmcloud.com\niot-
2/cmd/test/fmt/String\nsubscribe to cmd OK\n\n399 cms.\n399 cms.\n243 cms.\n151 cms.\n35 cms.\nSending
payload: {\"message\": \"alert\"}\nPublish ok\n41 cms.\nSending payload: {\"message\": \"alert\"}\nPublish
ok\n41 cms.\nSending payload: {\"message\": \"alert\"}\nPublish ok\n41 cms.\nSending payload:
{\"message\": \"alert\"}\nPublish ok");
  wificonnect();
  mqttconnect();
}
void loop()
{
  digitalWrite(TRIGGER, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER, LOW);
  int time=pulseIn(ECHO,HIGH);
  distance=(time*sound)/2;
  Serial.print("Distance:");
  Serial.print(distance);
  Serial.println("cms");
  if(distance<100){
    //PublishData(distance);
  }
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}
/*.....retrieving to Cloud ..... */
void PublishData(int d) {
  //mqttconnect();
  String payload = \"{\"message\": \"alert\"}\";
  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++) {
    data3 += (char)payload[i];
  }
  Serial.println("data: "+ data3);
  data3="";
}

```

## CIRCUIT DIAGRAM:

The screenshot shows the Wokwi IoT simulator interface. On the left, a code editor displays a C++ sketch for an ESP32 connected to an IBM Cloud IoT Platform. The code includes headers for WiFi and PubSubClient, defines constants for trigger, echo, sound, and distance, and sets up an MQTT client with specific credentials and topics. The main loop sends an alert message when a trigger is received. On the right, a simulation window shows a 3D model of the ESP32 board connected to a blue sensor module. Below the simulation, a console window displays the output of the program, showing the client reconnecting to the IBM Cloud IoT Platform and successfully sending an alert message.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3
4 #define TRIGGER 2
5 #define ECHO 15
6 #define sound 0.034
7 int distance;
8
9 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
10
11 //-----credentials of IBM Accounts-----
12
13 #define ORG "2jk8e5"
14 #define DEVICE_TYPE "Gas_leakage_Detection_Device"
15 #define DEVICE_ID "Gas_Detector_1"
16 #define TOKEN "123456789"
17 String data3;
18
19 //----- Customise the above values -----
20
21 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
22 char publishTopic[] = "iot-2/evt/Data/fmt/json";
23 char subscribetopic[] = "iot-2/cmd/test/fmt/String";
24 char authMethod[] = "use-token-auth";
25 char token[] = TOKEN;
26 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
27
28 //-----
29
30 WiFiClient wifiClient;
31 PubSubClient client(server, 1883, callback, wifiClient);
32 void setup()
33 {
34   Serial.begin(115200);
```

Reconnecting client to 2jk8e5.messaging.internetofthings.ibmcloud.com  
iot-2/cmd/test/fmt/String  
subscribe to cmd OK

399 cms.  
335 cms.  
258 cms.  
169 cms.  
41 cms.  
Sending payload: {"message":"alert"}  
Publish ok  
41 cms.  
Sending payload: {"message":"alert"}  
Publish ok

## IBM CLOUD RECENT EVENTS:

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes the IBM logo, the text "IBM Watson IoT Platform", and a user profile icon. The main content area is divided into a sidebar with navigation options (Browse, Action, Device Types, Interfaces) and a main panel. The main panel displays a table of devices, with the "Gas\_Detector\_1" device selected. Below the device list, a "Recent Events" tab is active, showing a list of events. The events table has columns for Event, Value, Format, and Last Received. The events listed are all "Data" events with a value of {"message":"alert"} in json format, received a few seconds ago.

Event	Value	Format	Last Received
Data	{"message":"alert"}	json	a few seconds ago
Data	{"message":"alert"}	json	a few seconds ago
Data	{"message":"alert"}	json	a few seconds ago
Data	{"message":"alert"}	json	a few seconds ago
Data	{"message":"alert"}	json	a few seconds ago