

## Assignment-4

Student Name	PRIYADHARSHINI
Register Number	810419106038
Project Name	Real Time River Water Quality Monitoring and Control System

### 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:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribtopic, byte* payload, unsigned int
payloadLength);
#define ORG "49x4b9"
#define DEVICE_TYPE "esp32"
#define DEVICE_ID "97043"
#define TOKEN "7993276080"
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++)
  {
    data3 += (char)payload[i];
  }
  Serial.println("data: "+ data3);
  data3="";
}

```

**Wokwi Link:** <https://wokwi.com/projects/347310895770108498>

## Output and Simulation :

The screenshot shows the Wokwi IDE interface. On the left, the sketch.ino file contains the following code:

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int length) {
4   #define ORG "92zbfcc"
5   #define DEVICE_TYPE "esp32"
6   #define DEVICE_ID "12345"
7   #define TOKEN "12345678"
8   String data3;
9   char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
10  char publishTopic[] = "iot-2/evt/Data/fmt/json";
11  char subscribTopic[] = "iot-2/cmd/test/fmt/String";
12  char authMethod[] = "use-token-auth";
13  char token[] = TOKEN;
14  char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
15  WiFiClient wifiClient;
16  PubSubClient client(server, 1883, callback, wifiClient);
17  const int trigPin = 5;
18  const int echoPin = 18;
19  #define SOUND_SPEED 0.034
20  long duration;
21  float distance;
22  void setup() {
23    Serial.begin(115200);
24    pinMode(trigPin, OUTPUT);
25    pinMode(echoPin, INPUT);
26    wificonnect();
27    mqttconnect();
28  }
29  void loop()
30  {
31    digitalWrite(trigPin, LOW);
```

The simulation window on the right shows an Ultrasonic Distance Sensor at 99cm. The console output displays the following messages:

```
100cms"}
Publish ok
Distance (cm): 98.94
ALERT!!
Sending payload: {"Distance":98.94,"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 dashboard. The top section displays a list of devices with the following columns: Device ID, Status, Device Type, Class ID, Date Added, and Descriptive Location. The device 12345 is shown as 'Connected' with type 'esp32'.

The 'Recent Events' tab for device 12345 is selected, showing a table of events:

Time	Value	Format	Last Received
Data	{"Distance":98.94,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":98.94,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":98.94,"ALERT!!":"Distance less than ...	json	a few seconds ago

The bottom of the dashboard shows a status bar with '0 Simulations running' and a 'Type here to search' field.