

## IBM ASSIGNMENT – 4

Assignment Date	07 NOVEMBER 2022
Student Name	Guhan G
Student Roll Number	710019106015
Team ID	PNT2022TMID42279

**Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 cm send "alert" to IBM cloud and display in device recent events.**

### **CODE:**

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribtopic, byte* payload, unsigned int
payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "udgvx5"//IBM ORGANITION ID
#define DEVICE_TYPE "esp32"//Device type mentioned in ibm watson
IOT Platform
#define DEVICE_ID "DHT_22"//Device ID mentioned in ibm watson IOT
Platform
#define TOKEN "Vanthiyan22" //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("");
}

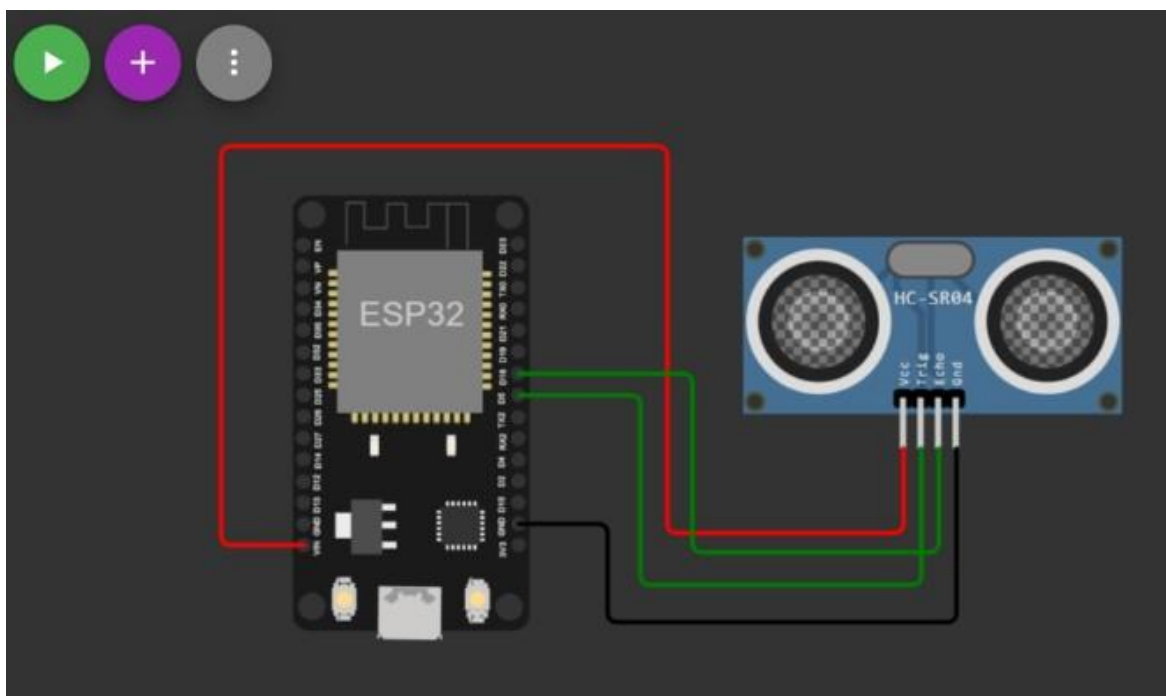
```

```

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="";
}

```

## SCHEMATIC DIAGRAM:



## WOKWI OUTPUT:

The screenshot shows the Wokwi online IDE interface. On the left, the sketch code is displayed, which includes an ESP32 setup and a PubSubClient library for IoT communication. On the right, the simulation window shows a virtual circuit with an ESP32 module and an Ultrasonic Distance Sensor. The sensor's distance is currently 43cm. Below the simulation, the serial output shows the device publishing data and sending an alert when the distance is less than 100cm.

```
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 "udgvx5"//IBM ORGANITION ID
7 #define DEVICE_TYPE "esp32"//Device type mentioned in ibm watson IOT Plat
8 #define DEVICE_ID "DHT_22"//Device ID mentioned in ibm watson IOT Platfor
9 #define TOKEN "Vanthiyan22" //Token
10 String data3;
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();
```

Simulation Output:

```
100cms"}
Publish ok
Distance (cm): 42.98
ALERT!!
Sending payload: {"Distance":42.98,"ALERT!!":"Distance less than
100cms"}
Publish ok
```

## WOKWI LINK:

<https://wokwi.com/projects/347519715656073812>

## IBM CLOUD OUTPUT:

The screenshot shows the IBM Watson IoT Platform dashboard. The 'Recent Events' tab is selected, displaying a table of live data streams from the device. The table includes columns for Event, Value, Format, and Last Received. The events show distance measurements and alerts.

Event	Value	Format	Last Received
Data	{"Distance":42.98,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":82.94,"ALERT!!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":82.99,"ALERT!!":"Distance less than ...	json	a minute ago
Data	{"Distance":61.97,"ALERT!!":"Distance less than ...	json	a minute ago
Data	{"Distance":45.97,"ALERT!!":"Distance less than ...	json	a minute ago

0 Simulations running