

ASSIGNMENT-4

QUESTION

Write code and connections in wowki for the ultrasonic sensor. Whenever the distance is less than 100 cms send a **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 "92zbfc"
#define DEVICE_TYPE "esp32"
#define DEVICE_ID "12345"
#define TOKEN "12345678"
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/348224065422492242>

OUTPUT & SIMULATION:

The screenshot displays the Wokwi web-based IDE. On the left, the 'sketch.ino' file is open, showing the following code:

```

1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
4 #define ORG "922bfc"
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 subscribetopic[] = "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   wifiClient.setTimeout(10);
27   mqttconnect();
28 }
29 void loop()
30 {
31   digitalWrite(trigPin, LOW);
32   delayMicroseconds(2);

```

On the right, the 'Simulation' window shows a virtual circuit. An ESP32 microcontroller is connected to an HC-SR04 ultrasonic sensor. The sensor's VCC pin is connected to the ESP32's 5V pin, and its GND pin is connected to the ESP32's GND pin. The sensor's TRIG pin is connected to the ESP32's pin 5, and its ECHO pin is connected to the ESP32's pin 18. A status bar at the top of the simulation window indicates 'Distance: 98cm'.

Below the circuit diagram, the serial monitor output is visible, showing the following messages:

```

ALERT!!
Sending payload: {"Distance":97.97,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 97.97
ALERT!!
Sending payload: {"Distance":97.97,"ALERT!!":"Distance less than 100cms"}
Publish ok

```

Alert to IBM cloud:

The screenshot displays the IBM Watson IoT Platform interface. The browser address bar shows the URL `pybd31.internetofthings.ibmcloud.com/dashboard/devices/browse`. The page header includes the user's email `arnikashree3101@gmail.com` and ID `pybd31`. The main navigation bar has tabs for `Browse`, `Action`, `Device Types`, and `Interfaces`, along with an `Add Device` button. The `Recent Events` tab is selected, showing a table of live data streams. The table has columns for `Event`, `Value`, `Format`, and `Last Received`. Below the table, a status box indicates `2 Simulations running`. The Windows taskbar at the bottom shows the system clock at 17:56 on 14-11-2022.

Event	Value	Format	Last Received
event_1	{"ultrasonic":3}	json	a few seconds ago
event_1	{"ultrasonic":91}	json	a few seconds ago
event_1	{"ultrasonic":93}	json	a few seconds ago
event_1	{"ultrasonic":74}	json	a few seconds ago
event_1	{"ultrasonic":16}	json	a few seconds ago