

Assignment 4:

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

```
#include <WiFi.h> #include
<PubSubClient.h> #include
<ArduinoJson.h> WiFiClient
wifiClient; #define ORG
"qr9sdm"
#define DEVICE_TYPE "device1" #define
DEVICE_ID "1234" #define TOKEN
"IUo-B9EP0h?_Fwp4C4" #define speed
0.034
char server[] = ORG "messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char topic[] = "iot-2/cmd/home/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
PubSubClient client(server, 1883, wifiClient);
void publishData();
const int trigpin=5;
const int echopin=18;
String command;
String data="";
long duration;
int dist;
void setup()
{
  Serial.begin(115200);
  pinMode(trigpin, OUTPUT);
  pinMode(echopin, INPUT);
  wifiConnect();
  mqttConnect();
}

void loop() {
  publishData();
  delay(500);
  if (!client.loop()) {
    mqttConnect();
  }
}

void wifiConnect() {
  Serial.print("Connecting to "); Serial.print("Wifi");
```

```

WiFi.begin("Wokwi-GUEST", "", 6);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.print("WiFi connected, IP address: ");
Serial.println(WiFi.localIP());
}
void mqttConnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting MQTT client to "); Serial.println(server);
    while (!client.connect(clientId, authMethod, token)) { Serial.print(".");
      delay(1000);
    }
  }
  initManagedDevice();
  Serial.println();
}
}
void initManagedDevice() {
  if (client.subscribe(topic)) {
    Serial.println(client.subscribe(topic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}
void publishData()
{
  digitalWrite(trigpin, LOW);
  digitalWrite(trigpin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin, LOW);
  duration=pulseIn(echopin, HIGH);
  dist=duration*speed/2; if(dist<100){
  DynamicJsonDocument doc(1024);
  String payload;
  doc["Distance Alert:"]=dist;
  serializeJson(doc, payload);
  delay(300);
  Serial.print("\n");
  Serial.print("Sending payload: ");
  Serial.println(payload);
  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish OK");
  } else {
    Serial.println("Publish FAILED");
  }
}

```

WOKWI LINK:

<https://wokwi.com/projects/347594971923087956>

OUTPUT:

The screenshot shows the Wokwi IDE interface. On the left, the sketch code is displayed, which includes a subscription to a topic and a publish function that sends a JSON payload. The simulation window on the right shows an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor. The sensor's distance is shown as 83cm. The console output shows 'Publish OK' and 'Sending payload: {"Distance Alert":82}'.

```
60 void initManagedDevice() {
61   if (client.subscribe(topic)) {
62     Serial.println(client.subscribe(topic));
63     Serial.println("subscribe to cmd OK");
64   } else {
65     Serial.println("subscribe to cmd FAILED");
66   }
67 }
68 void publishData()
69 {
70   digitalWrite(trigpin, LOW);
71   digitalWrite(trigpin, HIGH);
72   delayMicroseconds(10);
73   digitalWrite(trigpin, LOW);
74   duration=pulseIn(echopin, HIGH);
75   dist=duration*speed/2;
76   if(dist<100){
77     DynamicJsonDocument doc(1024);
78     String payload;
79     doc["Distance Alert:"]=dist;
80     serializeJson(doc, payload);
81     delay(300);
82     Serial.print("\n");
83     Serial.print("Sending payload: ");
84     Serial.println(payload);
85     if (client.publish(publishTopic, (char*) payload.c_str())) {
86       Serial.println("Publish OK");
87     } else {
88       Serial.println("Publish FAILED");
89     }
90 }
```

CLOUD OUTPUT:

The screenshot shows the IBM Watson IoT Platform dashboard. The 'Recent Events' tab is selected, displaying a table of events. The table has columns for Event, Value, Format, and Last Received. The events listed are 'Data' and 'event_1', with values like '{"Distance Alert":82}' and '{}'. The last received time for all events is 'a few seconds ago'.

Event	Value	Format	Last Received
Data	{"Distance Alert":82}	json	a few seconds ago
Data	{"Distance Alert":82}	json	a few seconds ago
Data	{"Distance Alert":82}	json	a few seconds ago
event_1	{}	json	a few seconds ago
event_1	{}	json	a few seconds ago