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.

Code:

```
#include <WiFi.h>

#include <PubSubClient.h>

WiFiClient wifiClient;

String data3;

#define ORG "4yi0vc"

#define DEVICE_TYPE "nodeMcu"

#define DEVICE_ID "Assignment4"

#define TOKEN "123456789"

#define speed 0.034

#define led 14

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;

float dist;


void setup()

{

  Serial.begin(115200);

  pinMode(led, OUTPUT);

  pinMode(trigpin, OUTPUT);

  pinMode(echopin, INPUT);

  wifiConnect();

  mqttConnect();

}


void loop() {

  bool isNearby = dist < 100;

  digitalWrite(led, isNearby);


  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(500);
    }
    initManagedDevice();
    Serial.println();
  }
```

```
}


void initManagedDevice() {
  if (client.subscribe(topic)) {
    // Serial.println(client.subscribe(topic));
    Serial.println("IBM 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){
    String payload = "{\"Normal Distance\":";
    payload += dist;
    payload += "}";


    Serial.print("\n");
    Serial.print("Sending payload: ");
    Serial.println(payload);

    if (client.publish(publishTopic, (char*) payload.c_str())) {

      Serial.println("Publish OK");

    }


  }

    if(dist>101 && dist<111){

    String payload = "{\"Alert distance\":";

    payload += dist;

    payload += "}";


    Serial.print("\n");

    Serial.print("Sending payload: ");

    Serial.println(payload);

     if(client.publish(publishTopic, (char*) payload.c_str())) {

      Serial.println("Warning crosses 110cm -- it automaticaly of the loop");

      digitalWrite(led,HIGH);

    }else {

      Serial.println("Publish 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++){

   dist += (char)payload[i];

 }

 Serial.println("data:"+ data3);

 if(data3=="lighton"){

   Serial.println(data3);

   digitalWrite(led,HIGH);

 }

 data3="";

}
```