Assignment -4

TEAM ID	PNT2022TMID38457
PROJECT TITLE	Industry-Specific Intelligent Fire Management System

QUESTION :
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

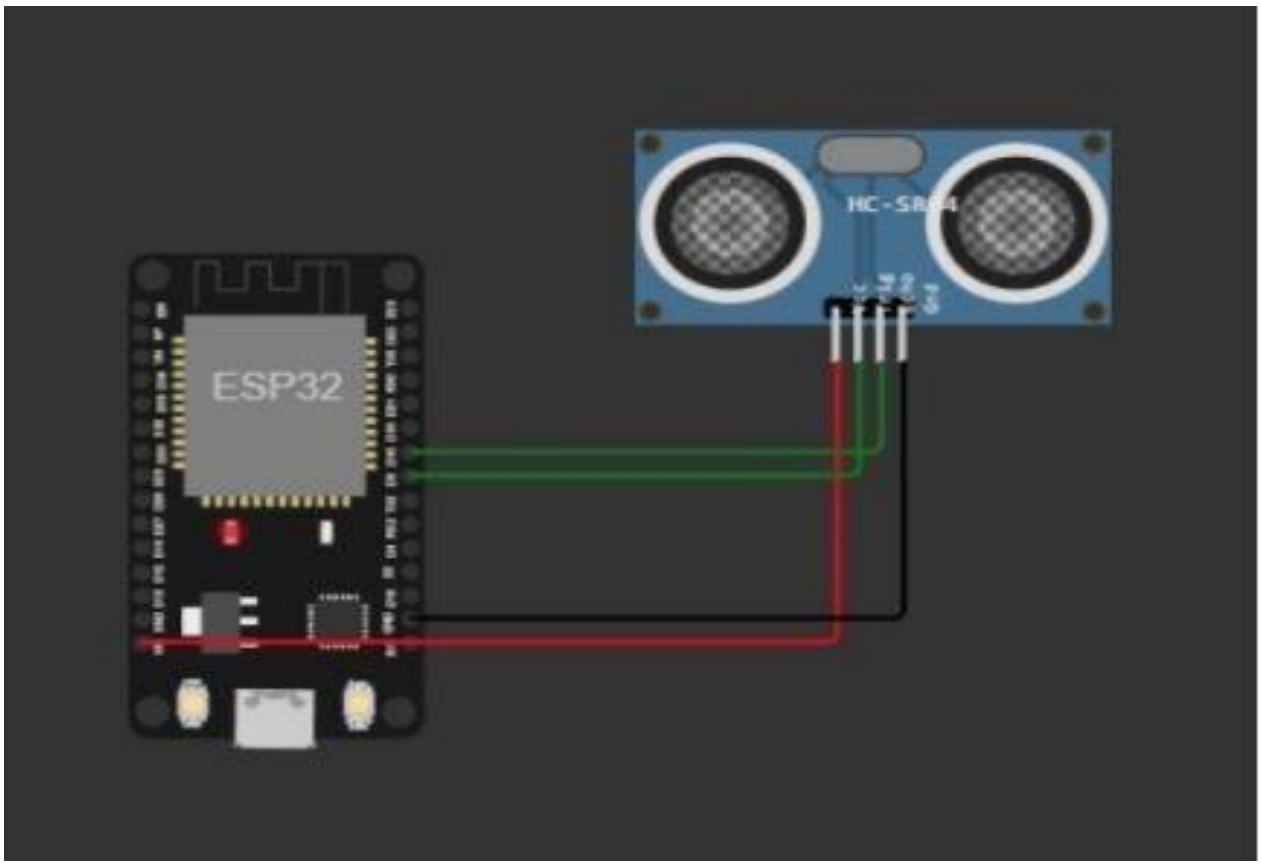
Solution :

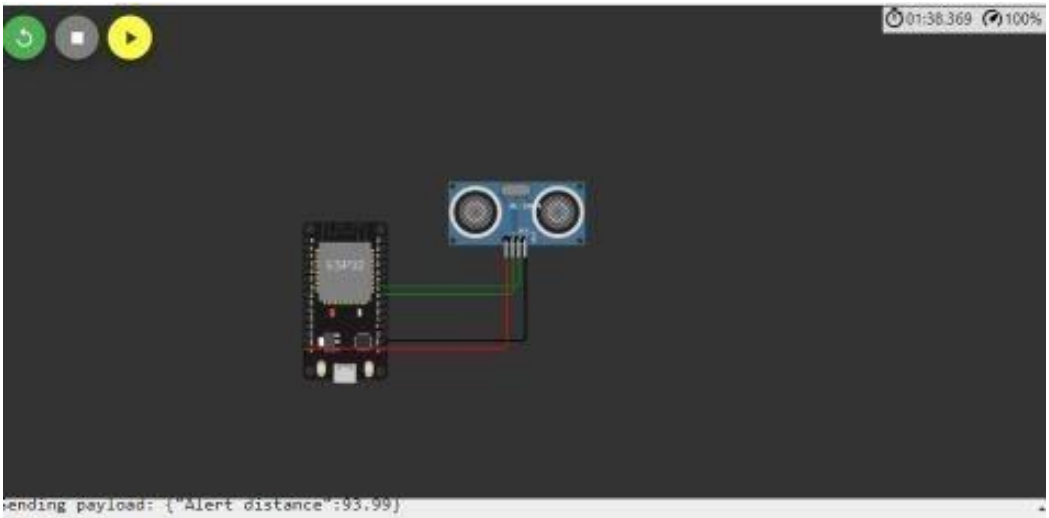
```
#include<WiFi.h>
#include<PubSubClient.h>
#include<ArduinoJson.h>

WiFiClientwifiClient;
#define ORG "nhpwjc"
#define DEVICE_TYPE "raspberypi"
#define DEVICE_ID "12345"
#define TOKEN "123456789"
#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();

constinttrigpin=5;
constintechopin=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){
            DynamicJsonDocumentdoc(1024);
            String payload;
            doc["AlertDistance:"]=dist;
            serializeJson(doc, payload);
            delay(3000);
            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");
            }
      }
}
```

OUTPUT:






```
ending payload: {"Alert distance":93.99}
ublish OK

ending payload: {"Alert distance":93.96}
ublish OK

ending payload: {"Alert distance":93.96}
ublish OK

ending payload: {"Alert distance":93.96}
ublish OK

ending payload: {"Alert distance":93.96}
ublish OK

ending payload: {"Alert distance":93.96}
ublish OK
```