

NAME: ARSHAD PARVEZ.G

REG. NO: 713319EC009

ASSIGNMENT - IV

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.

SOURCE CODE:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic,byte* payload, unsigned int
payloadLength);
#define ORG "g3gnbh"
#define DEVICE_TYPE "esp"
#define DEVICE_ID "942002"
#define TOKEN "20942002"
String data3;
char server[]= ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[]="iot-2/evt/distance/fmt/json";
char subscribeTopic[]="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);
#define ECHO_PIN 14
#define TRIG_PIN 12
#define led 27
void setup() {
// put your setup code here, to run once:
Serial.begin(115200);
pinMode(led, OUTPUT);
pinMode(TRIG_PIN, OUTPUT);
pinMode(ECHO_PIN, INPUT);
wificonnect();
mqttconnect();
}
float readDistanceCM() {
digitalWrite(TRIG_PIN, LOW);// Clear the trigger
delayMicroseconds(2);

digitalWrite(TRIG_PIN, HIGH);// Sets the trigger pin to HIGH state for 10
microseconds
```

```

delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
int duration=pulseIn(ECHO_PIN, HIGH);
//Serial.println(duration);
//duration = pulseIn(ECHO_PIN, HIGH);
return duration*0.017;
//Serial.println(duration);
}
void loop() {
float distance = readDistanceCM();
//Serial.println(distance);
bool isNearby = distance < 100;
digitalWrite(led, isNearby);
Serial.print("Measured distance: ");
Serial.println(distance);
if(distance<100){
PublishData2(distance);
}else{
PublishData1(distance);
}
//PublishData(distance);
delay(1000);
if(!client.loop()){
mqttconnect();
}
//delay(2000);
}
void PublishData1(float dist){
mqttconnect();
String payload= "{\"distance\":\"";
payload += dist;
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 PublishData2(float dist){
mqttconnect();
String payload= "{\"ALERT\":\"";
payload += dist;
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");
}
}

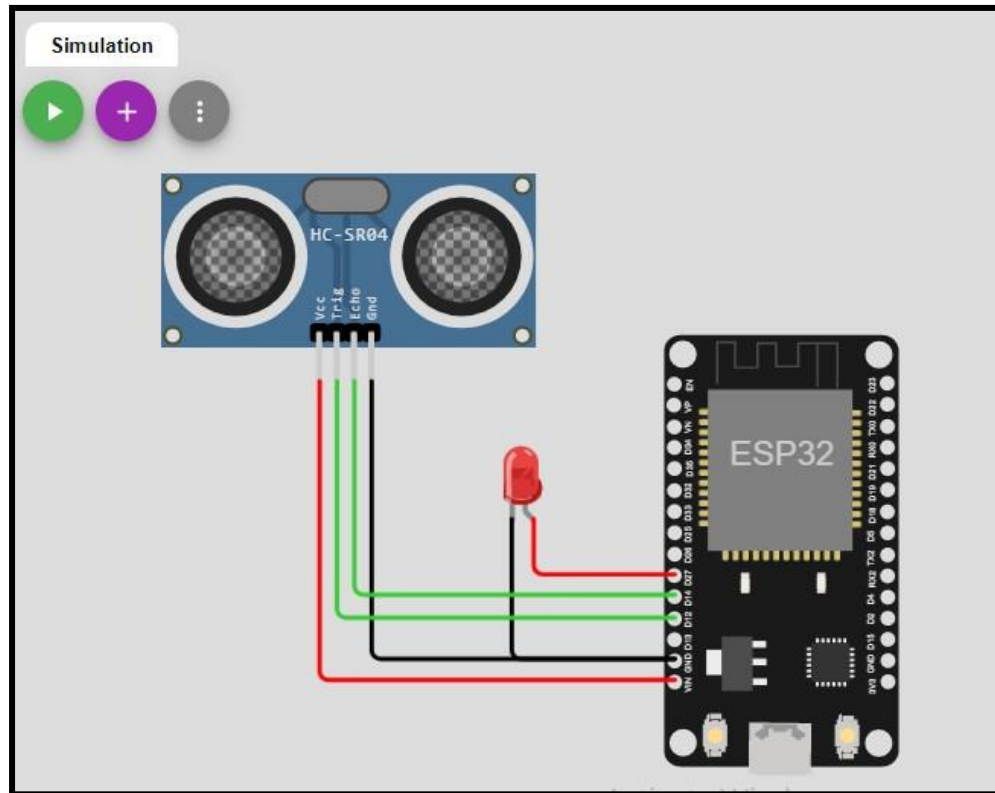
```

```
}  
}
```

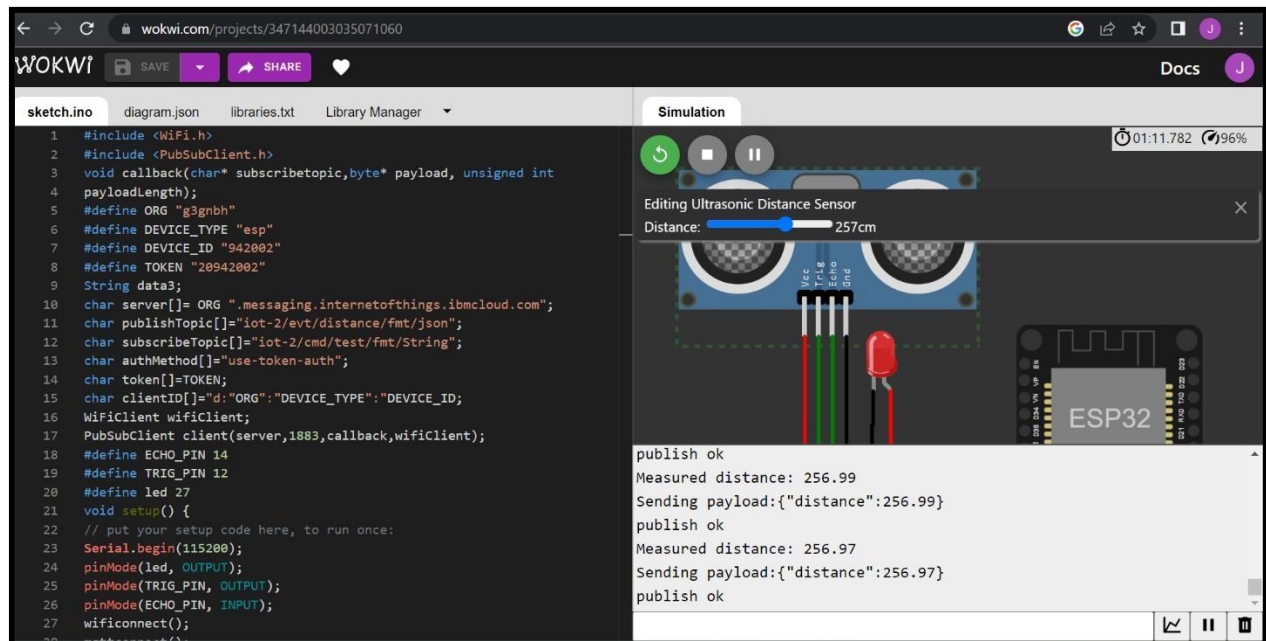
```
void mqttconnect(){  
  
    if(!client.connected()){  
        Serial.print("Reconnecting to ");  
        Serial.println(server);  
        while(!!!client.connect(clientID, authMethod, token)){  
            Serial.print(".");  
            delay(500);  
        }  
        initManagedDevice();  
        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);  
    if(data3=="lighton"){  
        Serial.println(data3);  
        digitalWrite(led, HIGH);  
    }else{  
        Serial.println(data3);  
        digitalWrite(led, LOW);  
    }  
    data3="";  
}
```

OUTPUT:

CONNECTION IN WOWKI FOR ULTRASONIC SENSOR:



NORMAL CASE: (distance > 100cms)



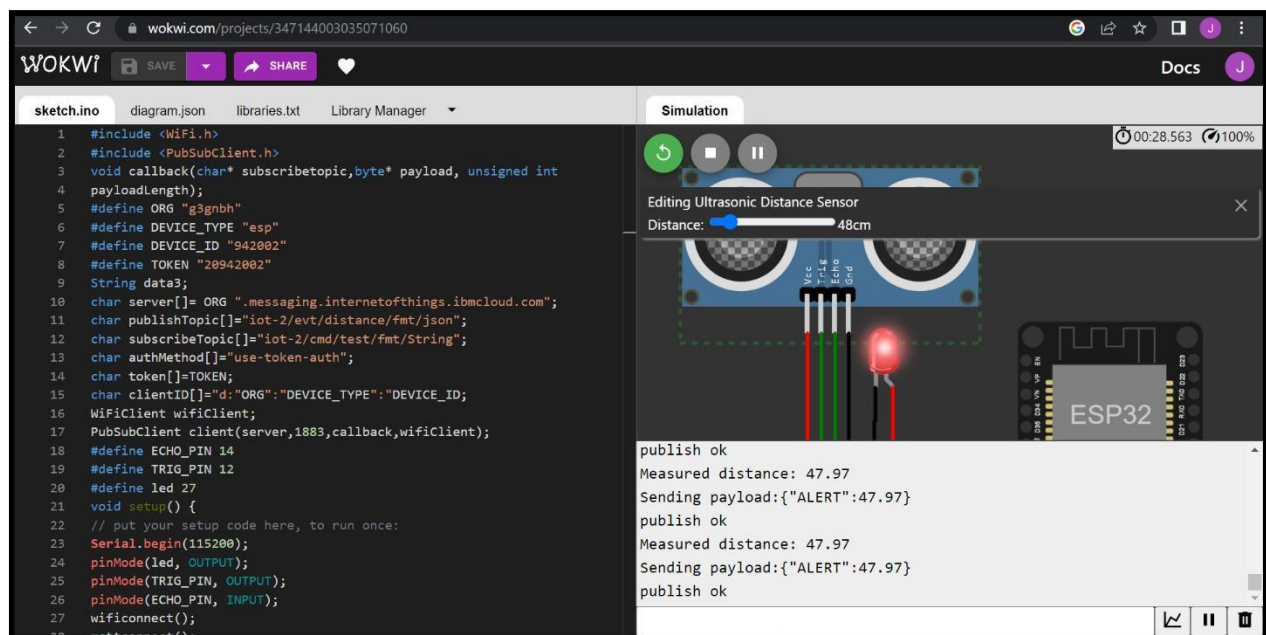
The screenshot shows the Wokwi web IDE interface. The sketch.ino file contains the following code:

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribetopic,byte* payload,unsigned int
4 payloadLength);
5 #define ORG "g3gnbh"
6 #define DEVICE_TYPE "esp"
7 #define DEVICE_ID "942002"
8 #define TOKEN "20942002"
9 String data3;
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/distance/fmt/json";
12 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
13 char authMethod[] = "use-token-auth";
14 char token[] = TOKEN;
15 char clientID[] = "d:"ORG":DEVICE_TYPE":DEVICE_ID;
16 WiFiClient wificlient;
17 PubSubClient client(server,1883,callback,wificlient);
18 #define ECHO_PIN 14
19 #define TRIG_PIN 12
20 #define led 27
21 void setup() {
22 // put your setup code here, to run once:
23 Serial.begin(115200);
24 pinMode(led, OUTPUT);
25 pinMode(TRIG_PIN, OUTPUT);
26 pinMode(ECHO_PIN, INPUT);
27 wificonnect();
28 }
```

The simulation window shows the Ultrasonic Distance Sensor with a distance of 257cm. The console output is as follows:

```
publish ok
Measured distance: 256.99
Sending payload:{"distance":256.99}
publish ok
Measured distance: 256.97
Sending payload:{"distance":256.97}
publish ok
```

ALERT CASE: (distance < 100cms)



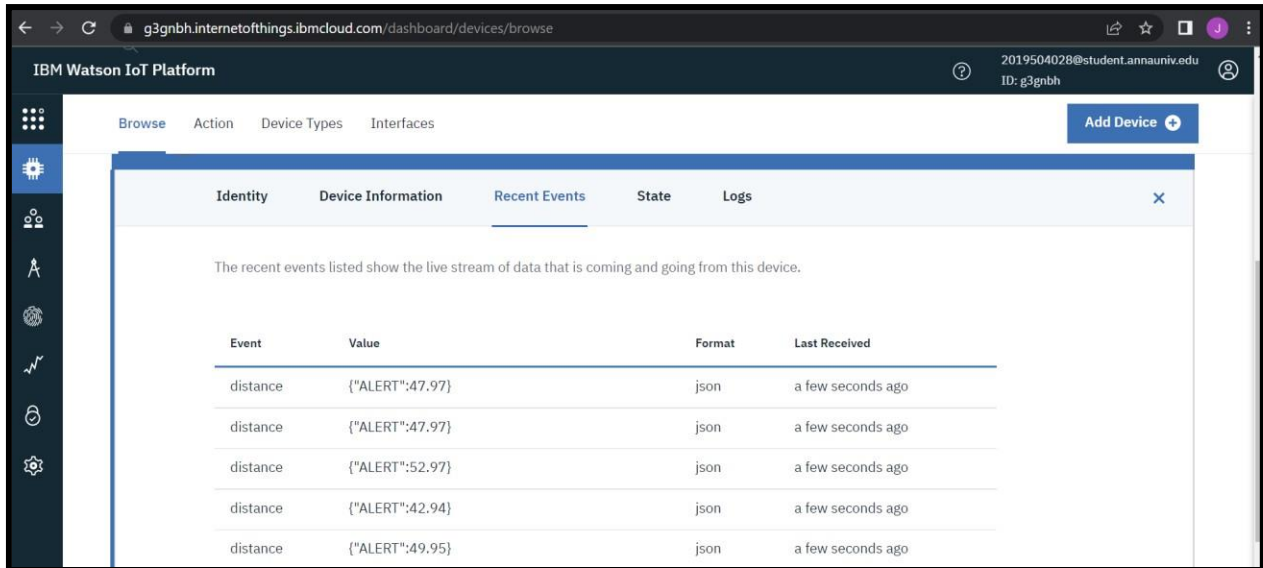
The screenshot shows the Wokwi web IDE interface. The sketch.ino file contains the following code:

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribetopic,byte* payload,unsigned int
4 payloadLength);
5 #define ORG "g3gnbh"
6 #define DEVICE_TYPE "esp"
7 #define DEVICE_ID "942002"
8 #define TOKEN "20942002"
9 String data3;
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/distance/fmt/json";
12 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
13 char authMethod[] = "use-token-auth";
14 char token[] = TOKEN;
15 char clientID[] = "d:"ORG":DEVICE_TYPE":DEVICE_ID;
16 WiFiClient wificlient;
17 PubSubClient client(server,1883,callback,wificlient);
18 #define ECHO_PIN 14
19 #define TRIG_PIN 12
20 #define led 27
21 void setup() {
22 // put your setup code here, to run once:
23 Serial.begin(115200);
24 pinMode(led, OUTPUT);
25 pinMode(TRIG_PIN, OUTPUT);
26 pinMode(ECHO_PIN, INPUT);
27 wificonnect();
28 }
```

The simulation window shows the Ultrasonic Distance Sensor with a distance of 48cm. The LED is on. The console output is as follows:

```
publish ok
Measured distance: 47.97
Sending payload:{"ALERT":47.97}
publish ok
Measured distance: 47.97
Sending payload:{"ALERT":47.97}
publish ok
```

IBM CLOUD DISPLAY IN RECENT EVENTS:



The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for device management. The main content area is titled 'Recent Events' and displays a table of live data streams. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. The data shows five entries for 'distance' events, each with a JSON value containing an alert and a distance value, all received 'a few seconds ago'.

Event	Value	Format	Last Received
distance	{"ALERT":47.97}	json	a few seconds ago
distance	{"ALERT":47.97}	json	a few seconds ago
distance	{"ALERT":52.97}	json	a few seconds ago
distance	{"ALERT":42.94}	json	a few seconds ago
distance	{"ALERT":49.95}	json	a few seconds ago

DISCUSSION OF THE RESULT:

- The connection has been made for ultrasonic sensor using LED and ESP32 using wowki simulator.
- It is observed that when the distance is greater than 100cms, it doesn't send any alert message.
- But, when the distance is less than 100cms, it sends an alert message to the IBM cloud and the corresponding message can be viewed under IBM cloud device recent events.