

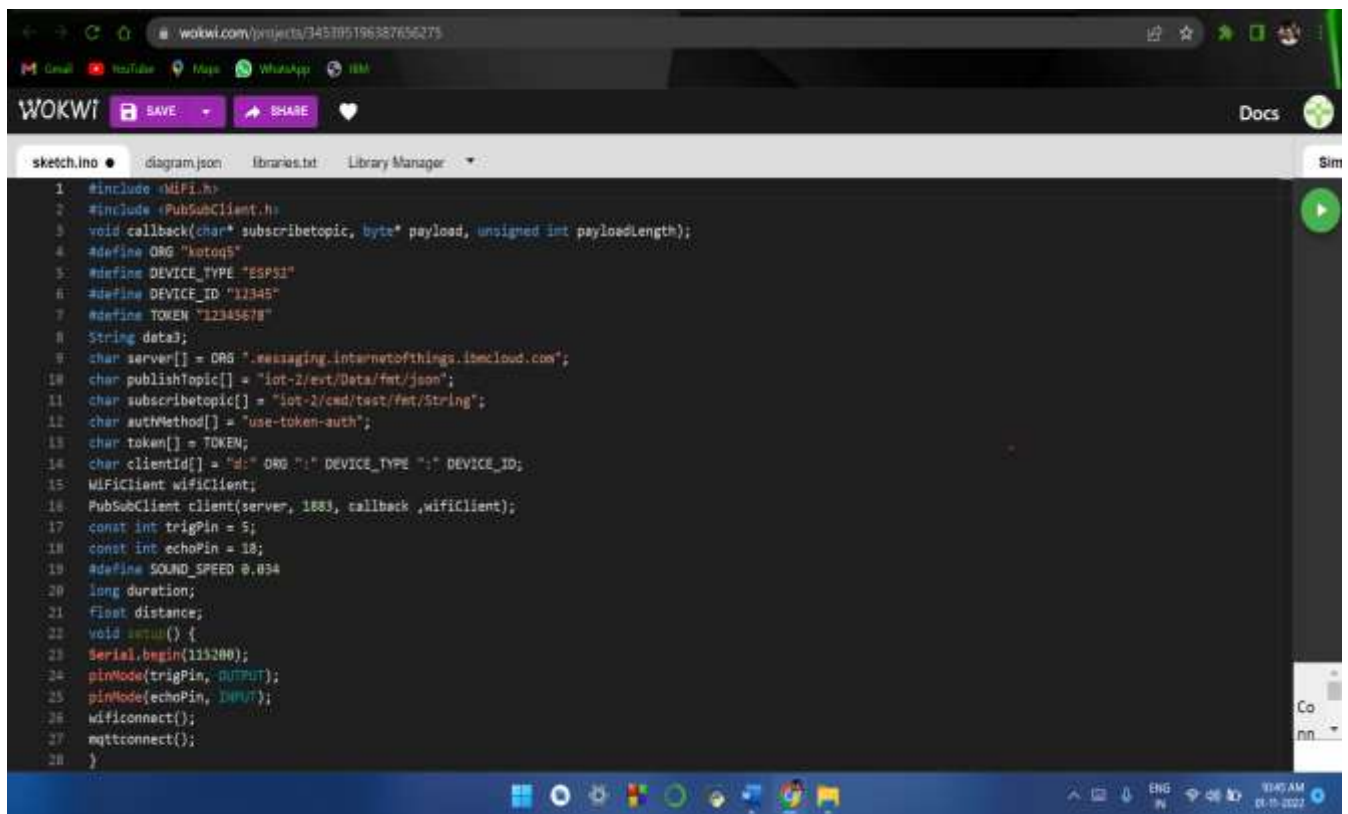
**Assignment -4**  
**Distance Detector using Ultrasonic**  
**Sensor**

Assignment Date	01 November 2022
Student Name	Arthi V
Student Roll Number	611219106003
Maximum Marks	2 Marks

**Question-1:**

Write code and Connection in Wokwi for ultrasonic sensor. Whenever distance is less than 100CMS send “alert” to IBM cloud and display in device recent events.

Wokwi Simulation Link: <https://wokwi.com/projects/345395196387656275>



```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength);
4 #define ORG "ketoq5"
5 #define DEVICE_TYPE "ESP8266"
6 #define DEVICE_ID "12345"
7 #define TOKEN "12345678"
8 String data;
9 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
10 char publishTopic[] = "iot-2/ext/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   wifiConnect();
27   mqttConnect();
28 }
```

WOKWI

sketch.ino

```
29 void loop()
30 {
31   digitalWrite(trigPin, LOW);
32   delayMicroseconds(2);
33   digitalWrite(trigPin, HIGH);
34   delayMicroseconds(10);
35   digitalWrite(trigPin, LOW);
36   duration = pulseIn(echoPin, HIGH);
37   distance = duration * SOUND_SPEED/2;
38   Serial.print("Distance (cm): ");
39   Serial.println(distance);
40   if(distance<100)
41   {
42     Serial.println("ALERT!!!");
43     delay(1000);
44     PublishData(distance);
45     delay(1000);
46     if (!client.connected()) {
47       mqttconnect();
48     }
49   }
50   delay(1000);
51 }
52 void PublishData(float dist) {
53   mqttconnect();
54   String payload = "{\"Distance\": ";
55   payload += dist;
56   payload += "\"\", \"ALERT\": \"\" \"Distance less than 100cm\"\"}";
```

WOKWI

sketch.ino

```
56   payload += "\"\", \"ALERT\": \"\" \"Distance less than 100cm\"\"}";
57   payload += "\"}";
58   Serial.print("Sending payload: ");
59   Serial.println(payload);
60   if (client.publish(publishTopic, (char*) payload.c_str())) {
61     Serial.println("Publish ok");
62   } else {
63     Serial.println("Publish failed");
64   }
65 }
66 void mqttconnect() {
67   if (!client.connected()) {
68     Serial.print("Reconnecting client to ");
69     Serial.println(server);
70     while (!client.connect(clientId, authMethod, token)) {
71       Serial.print(".");
72       delay(500);
73     }
74     initManagedDevice();
75     Serial.println();
76   }
77 }
78 void wificonnect()
79 {
80   Serial.println(); Serial.print("Connecting to ");
81   WiFi.begin("Mikol-GUEST", "", 0); while (WiFi.status() !=
82   WL_CONNECTED) { delay(500);
83   Serial.print(".");
```

wokwi.com/projects/345195196187656275

WOKWI

SAVE SHARE

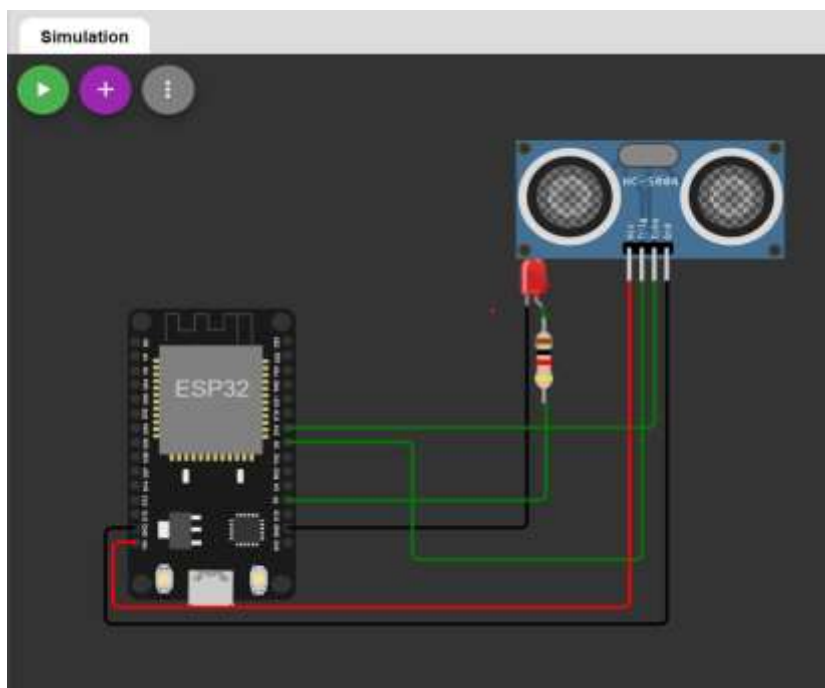
Docs

sketch.ino diagram.json libraries.txt Library Manager

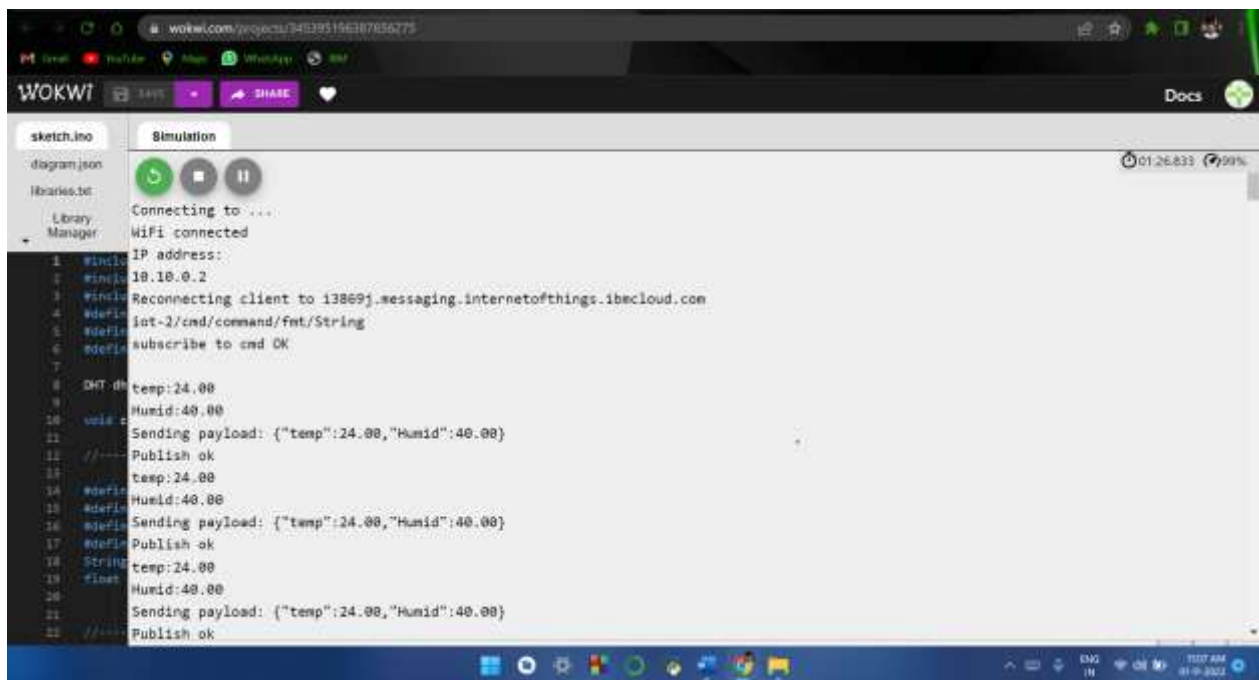
```
82 on_CONNECTED() { delay(500);
83   Serial.print(".");
84 }
85 Serial.println(""); Serial.println("Wifi connected");
86 Serial.println("IP address:");
87 Serial.println(WiFi.localIP());
88 }
89 void initManagedDevice() {
90   if (client.subscribe(subscribetopic)) {
91     Serial.println(subscribetopic); Serial.println("subscribe to cmd OK");
92   } else {
93     Serial.println("subscribe to cmd FAILED");
94   }
95 }
96 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
97 {
98   Serial.print("callback invoked for topic: ");
99   Serial.println(subscribetopic);
100   for (int i = 0; i < payloadLength; i++) {
101     //Serial.print((char)payload[i]);
102     data3 += (char)payload[i];
103   }
104   Serial.println("data: " + data3);
105   data3="";
106 }
```

Ca mn

9:48 AM 01-9-2022



## Wokwi Output



The screenshot displays the Wokwi web IDE interface. The top navigation bar includes the Wokwi logo, a 'SHARE' button, and a 'Docs' link. The left sidebar shows a file explorer with 'sketch.ino', 'diagram.json', 'libraries.txt', and 'Library Manager'. The main workspace is titled 'Simulation' and contains a code editor with an Arduino sketch. The sketch defines a LoRaWAN node that connects to a cloud broker, subscribes to a command topic, and publishes temperature and humidity data. The output window on the right shows the simulation progress, including connection status and data payloads.

```
1 #include <Arduino.h>
2 #include <LoRa.h>
3 #include <ReconnectingClient.h>
4 #define IoT-2/cmd/command/fmt/String
5 #define IoT-2/cmd/command/fmt/String
6 #define IoT-2/cmd/command/fmt/String
7
8 DHT dt temp:24.00
9 Humid:40.00
10 via z
11 Sending payload: {"temp":24.00,"Humid":40.00}
12 //---- Publish ok
13 temp:24.00
14 Humid:40.00
15 Sending payload: {"temp":24.00,"Humid":40.00}
16 Publish ok
17 String temp:24.00
18 Humid:40.00
19 Sending payload: {"temp":24.00,"Humid":40.00}
20 //---- Publish ok
```

