

Assignment -4

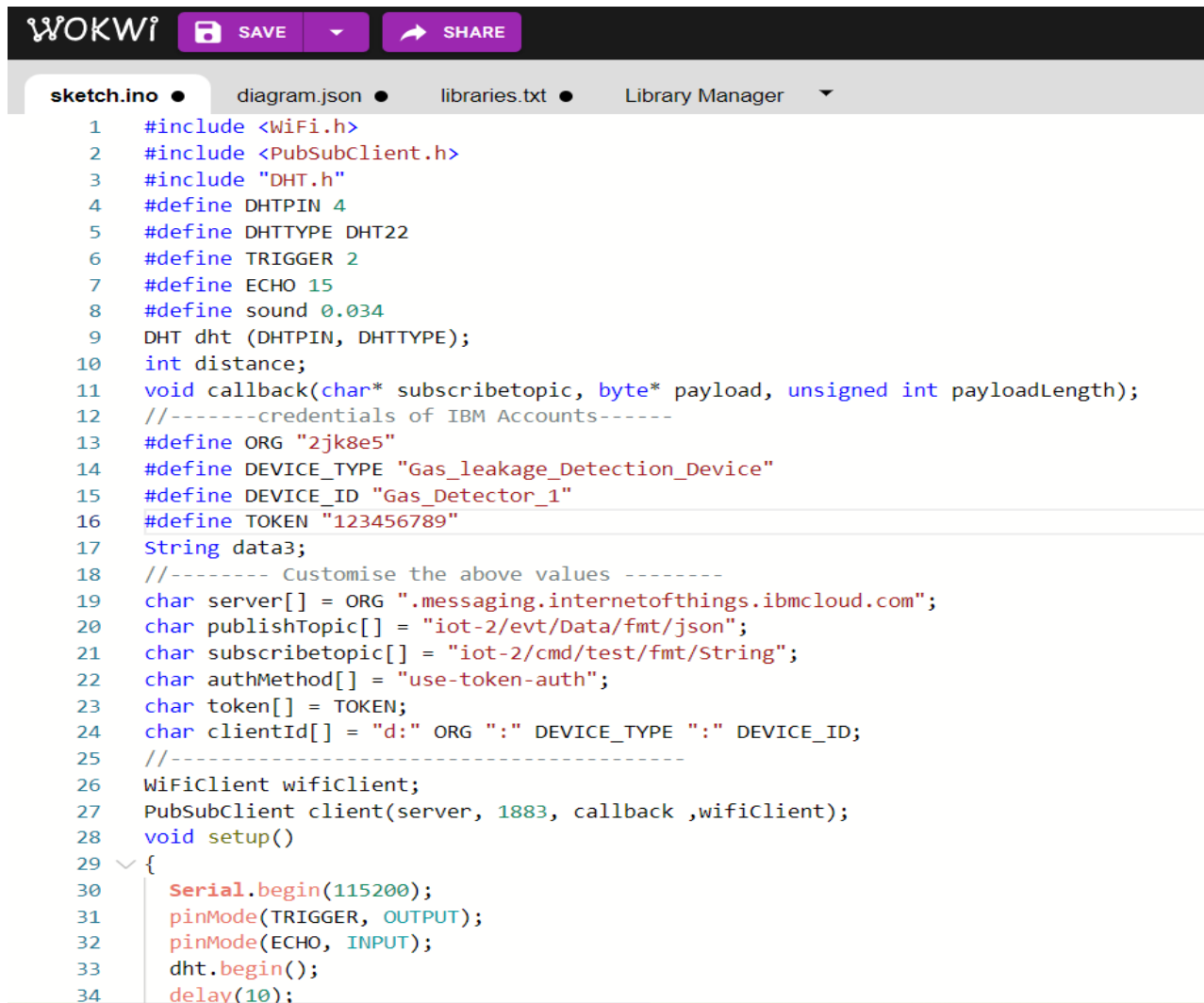
Assignment Date	20 OCTOBER 2022
Student Name	HARITHAA G
Student Roll Number	722819104045
Maximum Marks	2 Marks



Question-1:

Write code and connections in wowki for ultrasonic sensor.

Whenever distance is less than 100 cms send “alert” to IBM cloud and display in device recent events.

Solution:



```
WOKWI  SAVE  SHARE

sketch.ino • diagram.json • libraries.txt • Library Manager

1  #include <WiFi.h>
2  #include <PubSubClient.h>
3  #include "DHT.h"
4  #define DHTPIN 4
5  #define DHTTYPE DHT22
6  #define TRIGGER 2
7  #define ECHO 15
8  #define sound 0.034
9  DHT dht (DHTPIN, DHTTYPE);
10 int distance;
11 void callback(char* subscribtopic, byte* payload, unsigned int payloadLength);
12 //-----credentials of IBM Accounts-----
13 #define ORG "2jk8e5"
14 #define DEVICE_TYPE "Gas_leakage_Detection_Device"
15 #define DEVICE_ID "Gas_Detector_1"
16 #define TOKEN "123456789"
17 String data3;
18 //----- Customise the above values -----
19 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
20 char publishTopic[] = "iot-2/evt/Data/fmt/json";
21 char subscribtopic[] = "iot-2/cmd/test/fmt/String";
22 char authMethod[] = "use-token-auth";
23 char token[] = TOKEN;
24 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
25 //-----
26 WiFiClient wifiClient;
27 PubSubClient client(server, 1883, callback ,wifiClient);
28 void setup()
29 {
30   Serial.begin(115200);
31   pinMode(TRIGGER, OUTPUT);
32   pinMode(ECHO, INPUT);
33   dht.begin();
34   delay(10);
```

```
34     delay(10);
35     Serial.println();
36     wificonnect();
37     mqttconnect();
38 }
39 void loop()
40 {
41     digitalWrite(TRIGGER, HIGH);
42     delayMicroseconds(10);
43     digitalWrite(TRIGGER, LOW);
44     int time=pulseIn(ECHO,HIGH);
45     distance=(time*sound)/2;
46     Serial.print("Distance:");
47     Serial.print(distance);
48     Serial.println("cms");
49     if(distance<100){
50         //PublishData(distance);
51     }
52     delay(1000);
53     if (!client.loop()) {
54         mqttconnect();
55     }
56 }
57 /*.....retrieving to Cloud.....
58 void PublishData(int d) {
59     mqttconnect();
60     String payload = "{\"message\":\"alert\"}";
61     Serial.print("Sending payload: ");
62     Serial.println(payload);
63
64
65     if (client.publish(publishTopic, (char*) payload.c_str())) {
66         Serial.println("Publish ok");
67     } else {
```

```

66     Serial.println("Publish OK");
67   } else {
68     Serial.println("Publish failed");
69   }
70 }
71 void mqttconnect() {
72   if (!client.connected()) {
73     Serial.print("Reconnecting client to ");
74     Serial.println(server);
75     while (!client.connect(clientId, authMethod, token)) {
76       Serial.print(".");
77       delay(500);
78     }
79     initManagedDevice();
80     Serial.println();
81   }
82 }
83 void wificonnect()
84 {
85   Serial.println();
86   Serial.print("Connecting to ");
87   WiFi.begin("Wokwi-GUEST", "", 6);
88   while (WiFi.status() != WL_CONNECTED) {
89     delay(500);
90     Serial.print(".");
91   }
92   Serial.println("");
93   Serial.println("WiFi connected");
94   Serial.println("IP address: ");
95   Serial.println(WiFi.localIP());
96 }
97 void initManagedDevice() {
98   if (client.subscribe(subscribetopic)) {
99     Serial.println(subscribetopic);

```

```

sketch.ino • diagram.json • libraries.txt • Library Manager
97 void initManagedDevice() {
98     if (client.subscribe(subscribetopic)) {
99         Serial.println(subscribetopic);
100        Serial.println("subscribe to cmd OK");
101    } else {
102        Serial.println("subscribe to cmd FAILED");
103    }
104 }
105 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
106 {
107     Serial.print("callback invoked for topic: ");
108     Serial.println(subscribetopic);
109     for (int i = 0; i < payloadLength; i++) {
110         data3 += (char)payload[i];
111     }
112     Serial.println("data: " + data3);
113     data3="";
114 }

```

CIRCUIT DIAGRAM:

WOKWI SAVE SHARE Docs

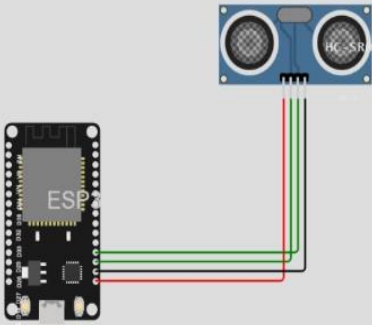
sketch.ino • diagram.json • libraries.txt • Library Manager

```

1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 #include "DHT.h"
4 #define DHTPIN 4
5 #define DHTTYPE DHT22
6 #define TRIGGER 2
7 #define ECHO 15
8 #define sound 0.034
9 DHT dht (DHTPIN, DHTTYPE);
10 int distance;
11 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
12 //-----credentials of IBM Accounts-----
13 #define ORG "2jk8e5"
14 #define DEVICE_TYPE "Gas_leakage_Detection_Device"
15 #define DEVICE_ID "Gas_Detector_1"
16 #define TOKEN "123456789"
17 String data3;
18 //----- Customise the above values -----
19 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
20 char publishTopic[] = "iot-2/evt/Data/fmt/json";
21 char subscribetopic[] = "iot-2/cmd/test/fmt/String";
22 char authMethod[] = "use-token-auth";
23 char token[] = TOKEN;
24 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
25 //-----
26 WiFiClient wifiClient;
27 PubSubClient client(server, 1883, callback, wifiClient);
28 void setup()
29 {
30     Serial.begin(115200);
31     pinMode(TRIGGER, OUTPUT);
32     pinMode(ECHO, INPUT);
33     dht.begin();
34     delay(10);

```

Simulation



Connecting to ...

IBM Cloud Recent Events:

IBM Watson IoT Platform

?

harithaa.g@sece.ac.in

ID: 2jk8e5

Browse

Action

Device Types

Interfaces

Add Device

	Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location																									
▼	Gas_Detector_1	Disconnected	Gas_Leakage_Detection_Device	Device	Oct 8, 2022 10:10 AM		→ ...																								
<div><div>Identity</div><div>Device Information</div><div>Recent Events</div><div>State</div><div>Logs</div><div>×</div></div> <div><div>The recent events listed show the live stream of data that is coming and going from this device.</div><table><thead><tr><th>Event</th><th>Value</th><th>Format</th><th>Last Received</th></tr></thead><tbody><tr><td>Data</td><td>{"message": "alert"}</td><td>json</td><td>a few seconds ago</td></tr><tr><td>Data</td><td>{"message": "alert"}</td><td>json</td><td>a few seconds ago</td></tr><tr><td>Data</td><td>{"message": "alert"}</td><td>json</td><td>a few seconds ago</td></tr><tr><td>Data</td><td>{"message": "alert"}</td><td>json</td><td>a few seconds ago</td></tr><tr><td>Data</td><td>{"message": "alert"}</td><td>json</td><td>a few seconds ago</td></tr></tbody></table></div>								Event	Value	Format	Last Received	Data	{"message": "alert"}	json	a few seconds ago	Data	{"message": "alert"}	json	a few seconds ago	Data	{"message": "alert"}	json	a few seconds ago	Data	{"message": "alert"}	json	a few seconds ago	Data	{"message": "alert"}	json	a few seconds ago
Event	Value	Format	Last Received																												
Data	{"message": "alert"}	json	a few seconds ago																												
Data	{"message": "alert"}	json	a few seconds ago																												
Data	{"message": "alert"}	json	a few seconds ago																												
Data	{"message": "alert"}	json	a few seconds ago																												
Data	{"message": "alert"}	json	a few seconds ago																												