

Assignment -4

Wokwi& IBM Cloud

| | |
|---------------------|-----------------|
| Assignment Date | 28 October 2022 |
| Student Name | Dharshini K |
| Student Roll Number | 732219CS020 |
| Maximum Marks | 2 Marks |

Question-1:

Write code and connections in wokwi for ultrasonic sensor. Whenever the distance is less than 100 cms sent "alert" to ibm cloud and display in device recent events.

Solution:

Code:

```
1  #include <WiFi.h>
2  #include <PubSubClient.h>
3  void callback(char* subscribetopic, byte* payload, unsigned int
4  payloadLength);
5  //-----credentials of IBM Accounts-----
6  #define ORG "e97wdi"//IBM ORGANITION ID
7  #define DEVICE_TYPE "IOT"//Device type mentioned in ibm watson IOT Platform
8  #define DEVICE_ID "device100"//Device ID mentioned in ibm watson IOT Platform
9  #define TOKEN "87654321" //Token
10 String data3;
11 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
12 char publishTopic[] = "iot-2/evt/Data/fmt/json";
13 char subscribetopic[] = "iot-2/cmd/test/fmt/String";
14 char authMethod[] = "use-token-auth";
15 char token[] = TOKEN;
16 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
17 WiFiClient wifiClient;
18 PubSubClient client(server, 1883, callback ,wifiClient);
19 const int trigPin = 5;
20 const int echoPin = 18;
21 #define SOUND_SPEED 0.034
22 long duration;
23 float distance;
24 void setup() {
25   Serial.begin(115200);
26   pinMode(trigPin, OUTPUT);
27   pinMode(echoPin, INPUT);
28   wificonnect();
29   mqttconnect();
30 }
```

```

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

```

```

61   Serial.println(payload);
62
63   if (client.publish(publishTopic, (char*) payload.c_str())) {
64     Serial.println("Publish ok");
65   } else {
66     Serial.println("Publish failed");
67   }
68 }
69 void mqttconnect() {
70   if (!client.connected()) {
71     Serial.print("Reconnecting client to ");
72     Serial.println(server);
73     while (!!!client.connect(clientId, authMethod, token)) {
74       Serial.print(".");
75       delay(500);
76     }
77     initManagedDevice();
78     Serial.println();
79   }
80 }
81 void wificonnect()
82 {
83   Serial.println();
84   Serial.print("Connecting to ");
85   WiFi.begin("Wokwi-GUEST", "", 6);
86   while (WiFi.status() != WL_CONNECTED) {
87     delay(500);
88     Serial.print(".");
89   }
90   Serial.println("");

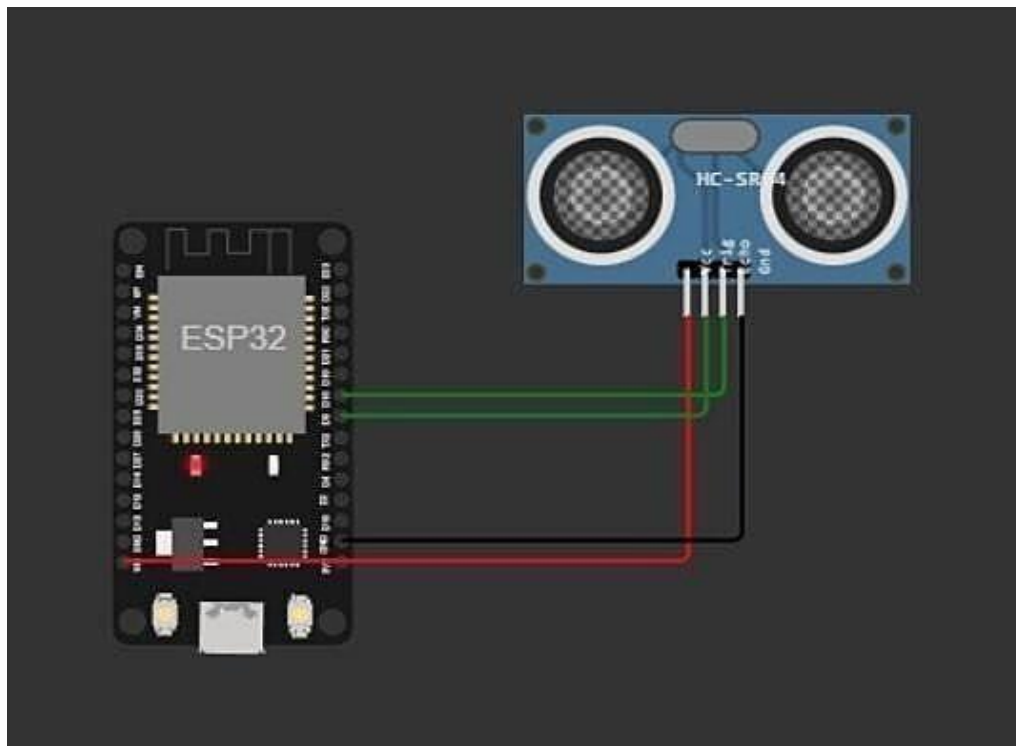
```

```

91 Serial.println("WiFi connected");
92 Serial.println("IP address: ");
93 Serial.println(WiFi.localIP());
94 }
95 void initManagedDevice() {
96   if (client.subscribe(subscribetopic)) {
97     Serial.println(subscribetopic);
98     Serial.println("subscribe to cmd OK");
99   } else {
100     Serial.println("subscribe to cmd FAILED");
101   }
102 }
103 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
104 {
105   Serial.print("callback invoked for topic: ");
106   Serial.println(subscribetopic);
107   for (int i = 0; i < payloadLength; i++) {
108     //Serial.print((char)payload[i]);
109     data3 += (char)payload[i];
110   }
111   Serial.println("data: "+ data3);
112   data3="";
113 }

```

Connections:



Output (wokwi):

The screenshot shows the Wokwi IDE interface. On the left, the 'sketch.ino' file contains C++ code for an Arduino Uno. The code includes libraries for WiFi and PubSubClient, defines credentials for an IBM Watson IoT device, and sets up a distance sensor (trigPin and echoPin) with a buzzer (pin 18). The main loop reads the distance and sends JSON payloads to the IoT cloud, including an 'ALERT!!' message when the distance is less than 100cm.

On the right, the 'Simulation' window shows the output of the code. It displays the connection status (WiFi connected, IP address: 10.10.0.2), the device ID (e97wdi), and the MQTT client ID (e97wdi). The output shows the distance being read (97.02 cm, 96.99 cm) and the corresponding JSON payloads being sent to the IoT cloud.

Link: <https://wokwi.com/projects/347015477471478354>

Output (IBM Cloud):

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. The main content area displays a table of recent events for a device named 'device100'. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. The events show a live stream of data from the device, including distance readings and alert messages.

| Event | Value | Format | Last Received |
|-------|---|--------|-------------------|
| Data | {"Distance":96.99,"ALERT!!":"Distance less than ... | json | a few seconds ago |
| Data | {"Distance":96.99,"ALERT!!":"Distance less than ... | json | a few seconds ago |
| Data | {"Distance":96.99,"ALERT!!":"Distance less than ... | json | a few seconds ago |
| Data | {"Distance":96.99,"ALERT!!":"Distance less than ... | json | a few seconds ago |
| Data | {"Distance":96.99,"ALERT!!":"Distance less than ... | json | a few seconds ago |