

## ASSIGNMENT - 4

DATE	21 October 2021
TEAM ID	PNT2022TMID07000
NAME	JAYAPRIYA G
STUDENT ROLL NUMBER	1918116
MAXIMUM MARKS	2 Marks

### QUESTION:

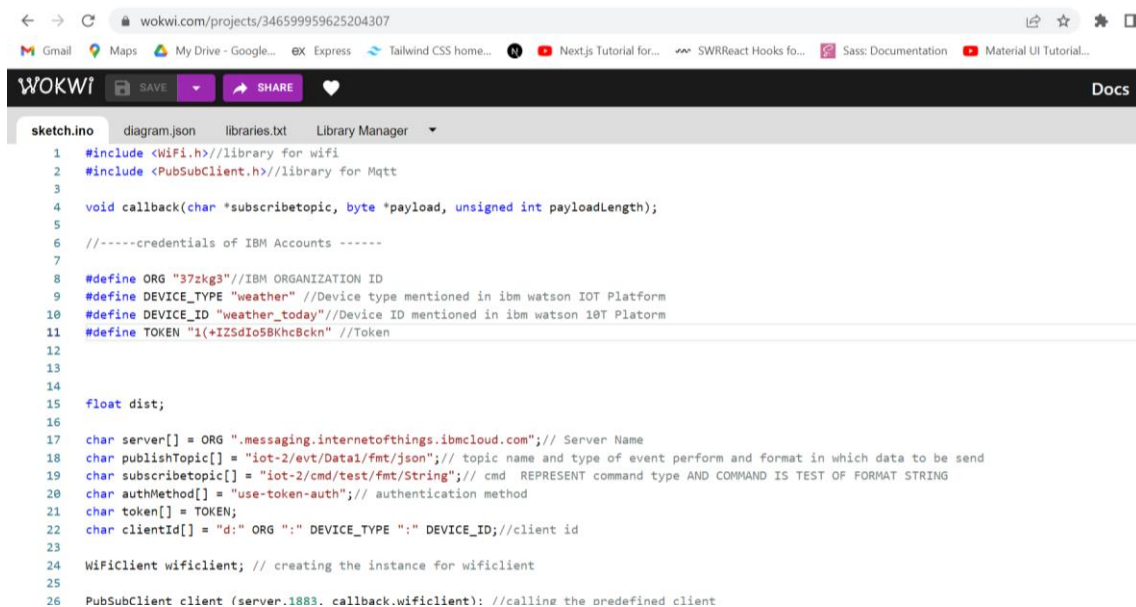
Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 cms send "alert" to ibm cloud and display in device recent events.

Upload document with wokwi share link and images of ibm cloud

### WOKWI CODE AND IMPLEMENTATION LINK:

<https://wokwi.com/projects/346599959625204307>

### CODE:



```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for Mqtt
3
4 void callback(char *topic, byte *payload, unsigned int payloadLength);
5
6 //-----credentials of IBM Accounts -----
7
8 #define ORG "37zkg3" //IBM ORGANIZATION ID
9 #define DEVICE_TYPE "weather" //Device type mentioned in ibm watson IOT Platform
10 #define DEVICE_ID "weather_today" //Device ID mentioned in ibm watson IOT Platform
11 #define TOKEN "1(+IZ5dIo5BKhc8ckn" //Token
12
13
14
15 float dist;
16
17 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
18 char publishTopic[] = "iot-2/evt/Data1/fmt/json"; // topic name and type of event perform and format in which data to be send
19 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING
20 char authMethod[] = "use-token-auth"; // authentication method
21 char token[] = TOKEN;
22 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
23
24 WiFiClient wificlient; // creating the instance for wificlient
25
26 PubSubClient client (server, 1883, callback, wificlient); //calling the predefined client
```

← → ↻ wokwi.com/projects/346599959625204307

Gmail Maps My Drive - Google... Express Tailwind CSS home... Next.js Tutorial for... SWRReact Hooks fo... Sass: Documentation Material UI Tutorial...

WOKWI SAVE SHARE

Docs

sketch.ino diagram.json libraries.txt Library Manager

```
26 PubSubClient client (server,1883, callback,wificlient); //calling the predefined client
27
28 int LED = 4;
29
30 int trig =5;
31
32 int echo= 18;
33
34 void setup()
35 {
36
37   Serial.begin(115200);
38   pinMode(trig, OUTPUT);
39   pinMode(echo, INPUT);
40   pinMode(LED, OUTPUT);
41   delay(10);
42
43   wificonnect();
44
45   mqttconnect();
46
47 }
48
49 void loop()// Recursive Function
50
```

← → ↻ wokwi.com/projects/346599959625204307

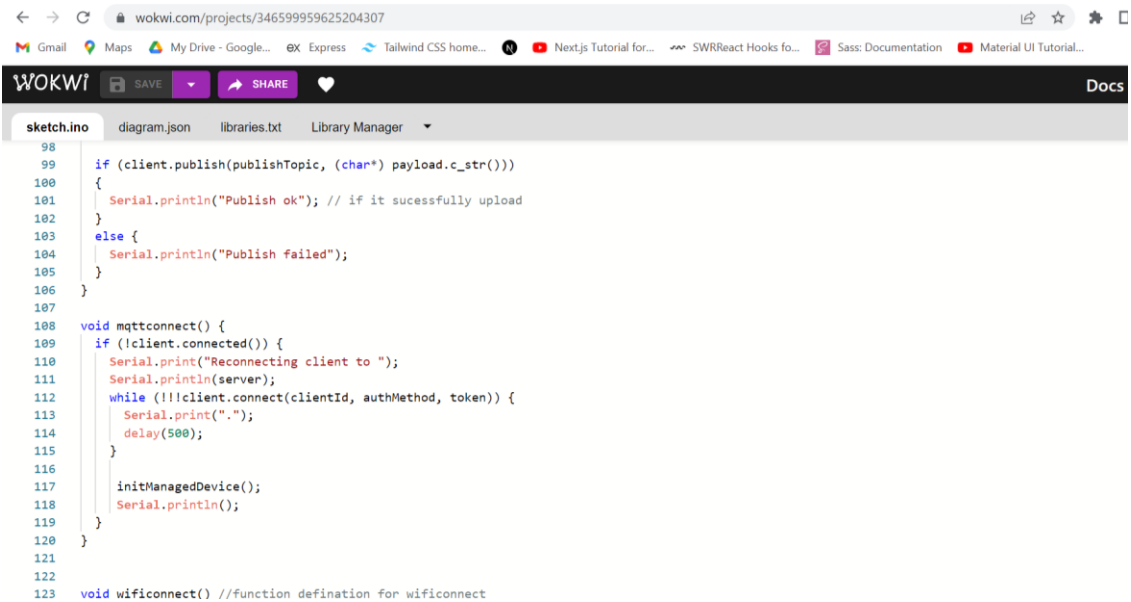
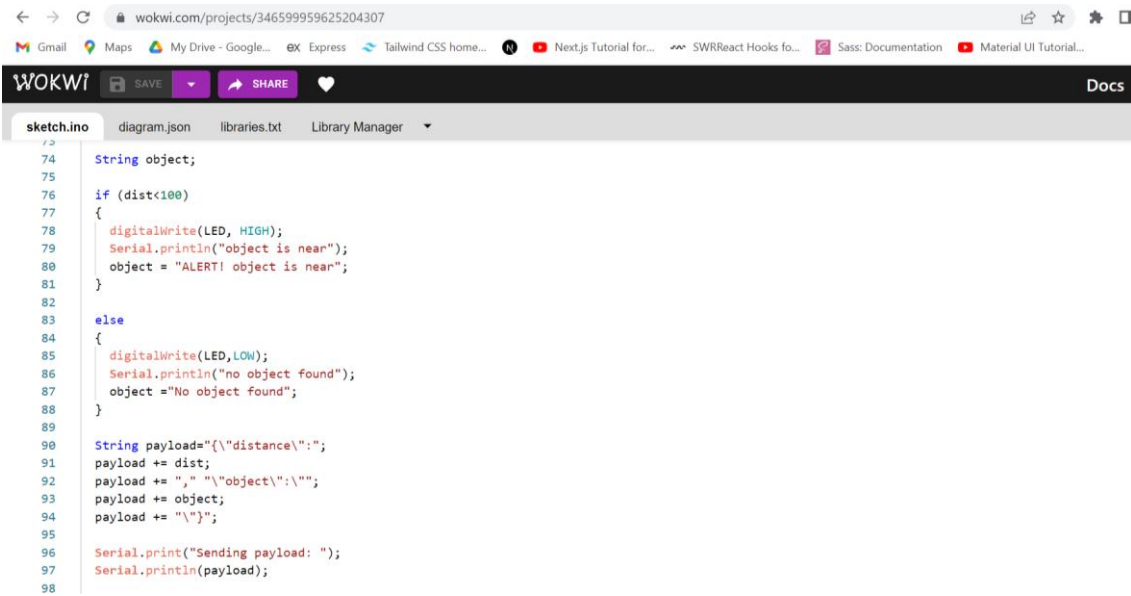
Gmail Maps My Drive - Google... Express Tailwind CSS home... Next.js Tutorial for... SWRReact Hooks fo... Sass: Documentation Material UI Tutorial...

WOKWI SAVE SHARE

Docs

sketch.ino diagram.json libraries.txt Library Manager

```
48
49 void loop()// Recursive Function
50
51 {
52   delayMicroseconds(10);
53   digitalWrite(trig, LOW);
54   digitalWrite(trig, LOW);
55   digitalWrite(trig,HIGH);
56   float dur= pulseIn(echo,HIGH);
57   float dist = (dur* 0.0343)/2;
58   Serial.print ("Distance in cm : ");
59   Serial.println(dist);
60
61   PublishData(dist);
62
63   delay(1000);
64
65   if (!client.loop()) {
66     mqttconnect();
67   }
68 }
69
70
71 void PublishData(float dist) {
72   mqttconnect();
73 }
```



← → ↺ wokwi.com/projects/346599959625204307

Gmail Maps My Drive - Google... Express Tailwind CSS home... Next.js Tutorial for... SWRReact Hooks fo... Sass: Documentation Material UI Tutorial...

WOKWI SAVE SHARE

Docs

sketch.ino diagram.json libraries.txt Library Manager

```
124 {
125   Serial.println();
126   Serial.print("Connecting to ");
127
128   WiFi.begin("Wokwi-GUEST", "", 6); //passing the wifi credentials to establish the connection
129   while (WiFi.status() != WL_CONNECTED) {
130     delay(500);
131     Serial.print(".");
132   }
133   Serial.println("");
134   Serial.println("WiFi connected");
135   Serial.println("IP address: ");
136   Serial.println(WiFi.localIP());
137 }
138
139 void initManagedDevice() {
140
141   if (client.subscribe(subscribetopic)) {
142     Serial.println((subscribetopic));
143     Serial.println("subscribe to cmd OK");
144   }
145   else {
146     Serial.println("subscribe to cmd FAILED");
147   }
148 }
```

← → ↺ wokwi.com/projects/346599959625204307

Gmail Maps My Drive - Google... Express Tailwind CSS home... Next.js Tutorial for... SWRReact Hooks fo... Sass: Documentation Material UI Tutorial...

WOKWI SAVE SHARE

Docs

sketch.ino diagram.json libraries.txt Library Manager

```
149
150 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
151 {
152   Serial.print("callback invoked for topic: ");
153   Serial.println(subscribetopic);
154   for (int i = 0; i < payloadLength; i++) {
155     //Serial.print((char)payload[i]);
156     // data3 += (char)payload[i];
157   }
158
159   // Serial.println("data: "+ data3);
160   //if(data3=="lighton")
161   {
162     //Serial.println(data3);
163     digitalWrite(LED,HIGH);
164
165   }
166
167   //else
168   {
169     //Serial.println(data3);
170     digitalWrite(LED,LOW);
171
172   }
173   //data3="";
174 }
```

## OUTPUT:

When the distance is less than 100 cms, send an “alert” message to IBM Watson IoT Platform

The screenshot shows the Wokwi simulation environment. On the left, the sketch.ino file contains the following code:

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for Mqtt
3
4 void callback(char *topic, byte *payload, unsigned int len) {
5
6 //-----credentials of IBM Accounts -----
7
8 #define ORG "37zk3" //IBM ORGANIZATION ID
9 #define DEVICE_TYPE "weather" //Device type mentioned in ibm watson
10 #define DEVICE_ID "weather_today" //Device ID mentioned in ibm watson
11 #define TOKEN "1(+IZ5dIo5BKhc8ckn" //Token
12
13
14
15 float dist;
16
17 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Se
18 char publishTopic[] = "iot-2/evt/Data1/fmt/json"; // topic name and t
19 char subscribTopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT
20 char authMethod[] = "use-token-auth"; // authentication method
21 char token[] = TOKEN;
22 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
23
24 WiFiClient wificlient; // creating the instance for wificlient
25
26 PubSubClient client (server, 1883, callback, wificlient); //calling th
```

On the right, the simulation shows an ESP32 microcontroller connected to an Ultrasonic Distance Sensor. The sensor's distance is 61cm. The console output shows the following messages:

```
object is near
Sending payload: {"distance":61.48,"object":"ALERT! object is near"}
Publish ok
Distance in cm : 61.50
object is near
Sending payload: {"distance":61.50,"object":"ALERT! object is near"}
Publish ok
```

## IBM CLOUD IMAGE

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes the following links: Browse, Action, Device Types, Interfaces. The main content area displays a table of events received from the device.

Event	Value	Format	Last Received
Data1	{"distance":61.5,"object":"ALERT! object is near"}	json	a few seconds ago
Data1	{"distance":61.5,"object":"ALERT! object is near"}	json	a few seconds ago
Data1	{"distance":61.5,"object":"ALERT! object is near"}	json	a few seconds ago
Data1	{"distance":61.5,"object":"ALERT! object is near"}	json	a few seconds ago
Data1	{"distance":61.55,"object":"ALERT! object is near"}	json	a few seconds ago

At the bottom of the table, there is a pagination bar showing "Items per page 50" and "1-1 of 1 item".

When the object is far( greater than 100 cms) , send “ no object found” to the IBM Watson IOT Platform.

The screenshot shows the Wokwi IDE interface. On the left, the `sketch.ino` file is open, displaying Arduino code for an ESP32 connected to an Ultrasonic Distance Sensor. The code includes libraries for WiFi and MQTT, defines IBM Cloud credentials, and sets up a callback function to send sensor data to the IBM Watson IoT Platform. The main loop reads the sensor distance and publishes a JSON payload if it's greater than 100 cm.

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for Mqtt
3
4 void callback(char *topic, byte *payload, unsigned int length) {
5   //-----credentials of IBM Accounts -----
6
7   #define ORG "37zk3" //IBM ORGANIZATION ID
8   #define DEVICE_TYPE "weather" //Device type mentioned in ibm watson
9   #define DEVICE_ID "weather_today" //Device ID mentioned in ibm watson
10  #define TOKEN "1(+IZ5dIoSBKhcBckn" //Token
11
12
13
14  float dist;
15
16
17  char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Se
18  char publishTopic[] = "iot-2/evt/Data1/fmt/json"; // topic name and t
19  char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT
20  char authMethod[] = "use-token-auth"; // authentication method
21  char token[] = TOKEN;
22  char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
23
24  WiFiClient wificlient; // creating the instance for wificlient
25
26  PubSubClient client (server,1883, callback,wificlient); //calling th
```

On the right, the simulation window shows an ESP32 board with an Ultrasonic Distance Sensor connected. The sensor's distance is set to 350 cm. The console output shows the following sequence of events:

```
no object found
Sending payload: {"distance":353.05,"object":"No object found"}
Publish ok
Distance in cm : 353.05
no object found
Sending payload: {"distance":353.05,"object":"No object found"}
Publish ok
```

## IBM CLOUD IMAGE

The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes links for Browse, Action, Device Types, and Interfaces. The main content area displays a table of events for a device named "37zk3". The table has columns for Event, Value, Format, and Last Received. The events are all of type "Data1" and contain a JSON payload indicating "No object found".

Event	Value	Format	Last Received
Data1	{"distance":353.05,"object":"No object found"}	json	a few seconds ago
Data1	{"distance":353.05,"object":"No object found"}	json	a few seconds ago
Data1	{"distance":353.05,"object":"No object found"}	json	a few seconds ago
Data1	{"distance":353.05,"object":"No object found"}	json	a few seconds ago
Data1	{"distance":353.05,"object":"No object found"}	json	a few seconds ago

At the bottom of the table, there is a pagination control showing "Items per page 50" and "1 of 1 page".