

ASSIGNMENT - 4

DATE	21 OCTOBER 2021
TEAM ID	PNT2022TMID07000
NAME	INDHIRANI S
STUDENT ROLL NUMBER	GCTC1918112
MAXIMUM MARKS	2 MARKS

QUESTION:

Write code and connections in wokwi for the ultrasonic sensor.

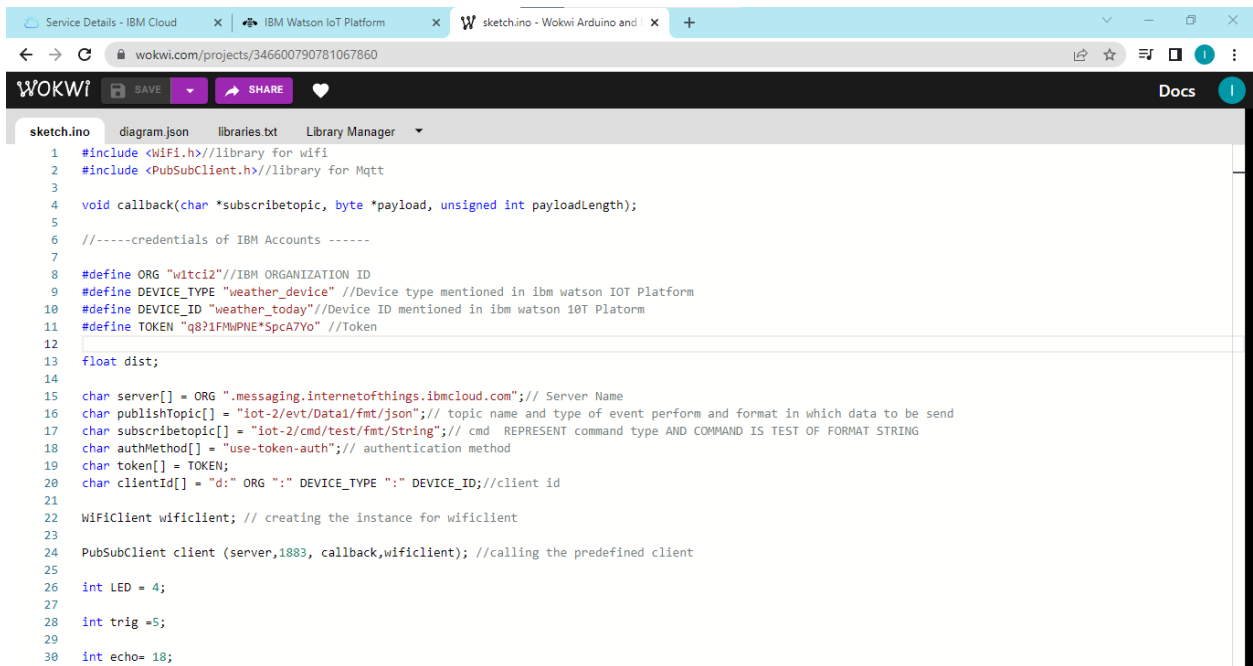
Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display in the device recent events.

Upload document with wokwi share link and images of IBM cloud

WOKWI CODE AND IMPLEMENTATION LINK:

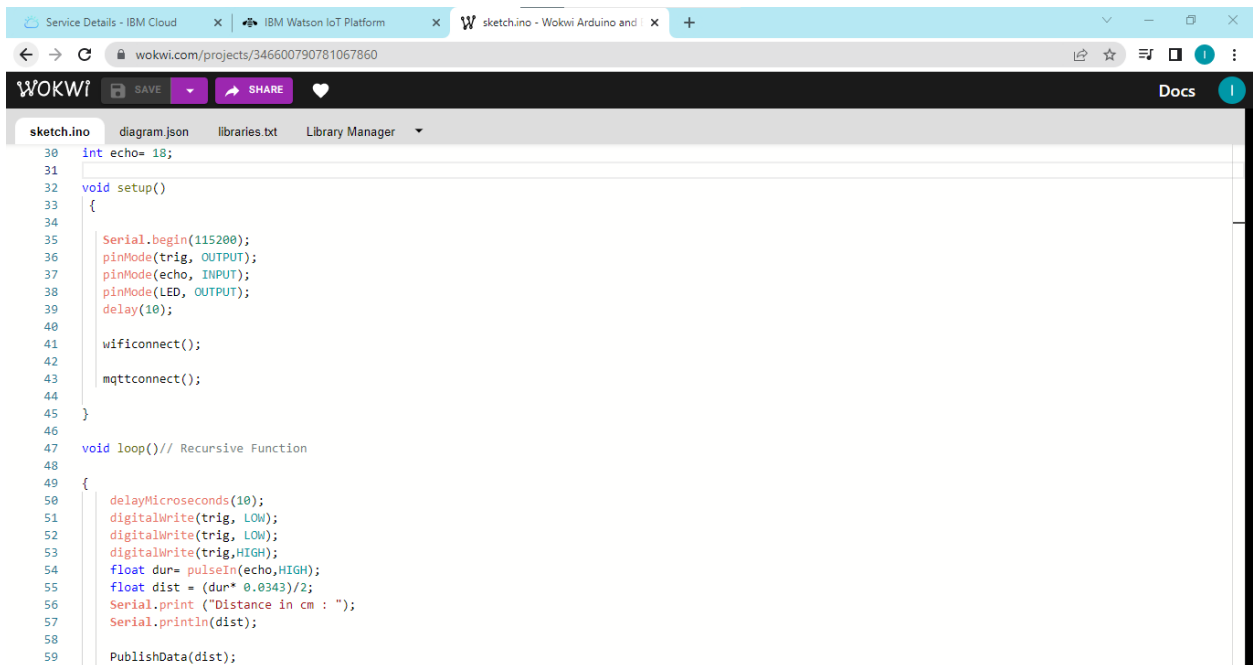
<https://wokwi.com/projects/346600790781067860>

CODE:



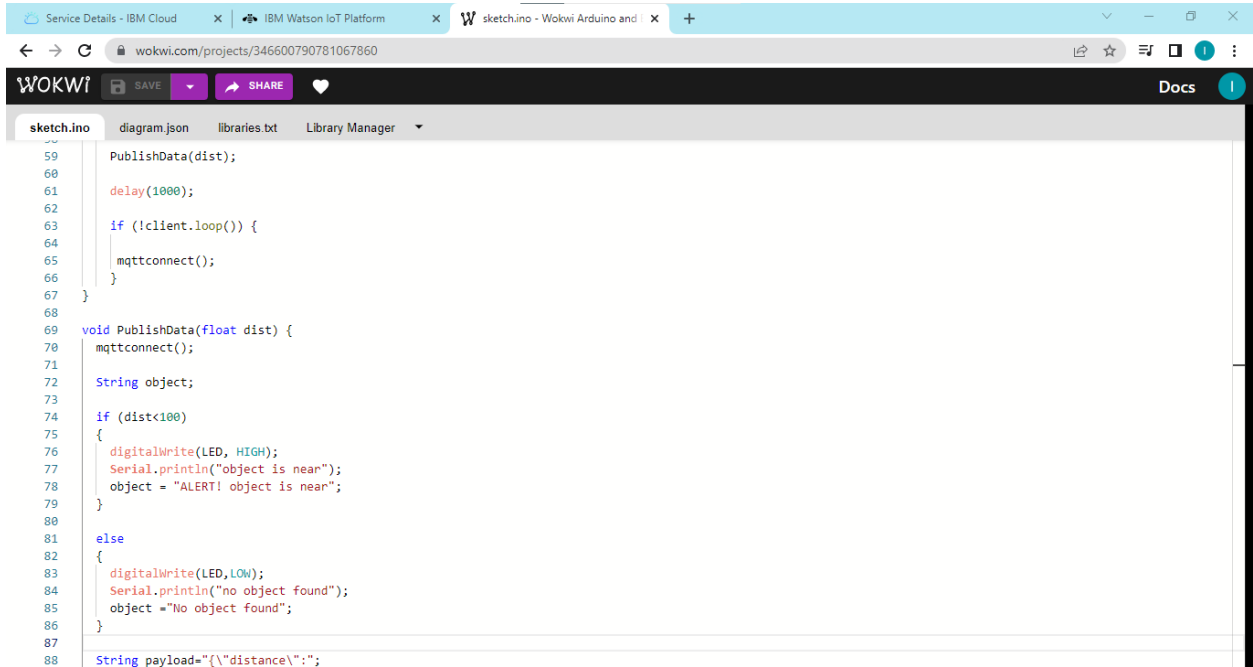
The screenshot shows the Wokwi IDE interface with the following code in sketch.ino:

```
1 #include <WiFi.h>//library for wifi
2 #include <PubSubClient.h>//library for Mqtt
3
4 void callback(char *subscriberTopic, byte *payload, unsigned int payloadLength);
5
6 //-----credentials of IBM Accounts -----
7
8 #define ORG "w1tc12"//IBM ORGANIZATION ID
9 #define DEVICE_TYPE "weather_device" //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 "q8?1FMwPNE*SpcA7Yo" //Token
12
13 float dist;
14
15 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
16 char publishTopic[] = "iot-2/evt/Data1/fmt/json";// topic name and type of event perform and format in which data to be send
17 char subscriberTopic[] = "iot-2/cmd/test/fmt/String";// cmd REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING
18 char authMethod[] = "use-token-auth";// authentication method
19 char token[] = TOKEN;
20 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
21
22 WiFiClient wificlient; // creating the instance for wificlient
23
24 PubSubClient client (server,1883, callback,wificlient); //calling the predefined client
25
26 int LED = 4;
27
28 int trig =5;
29
30 int echo= 18;
```

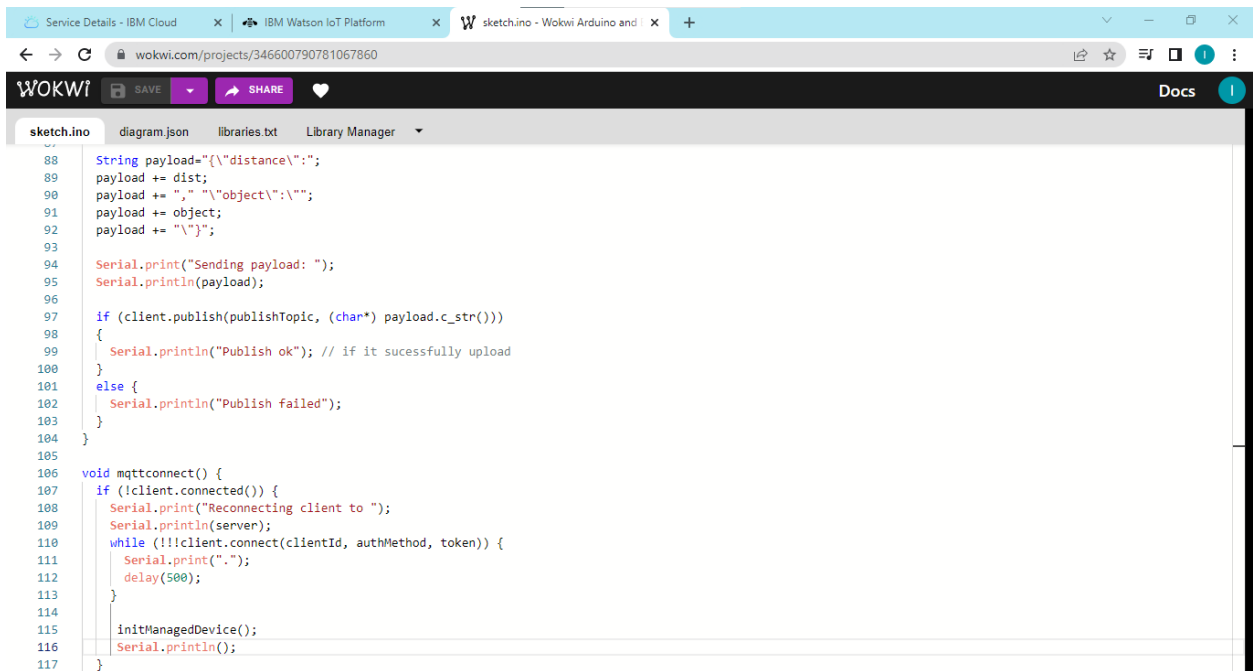


The screenshot shows the continuation of the Wokwi IDE interface with the following code in sketch.ino:

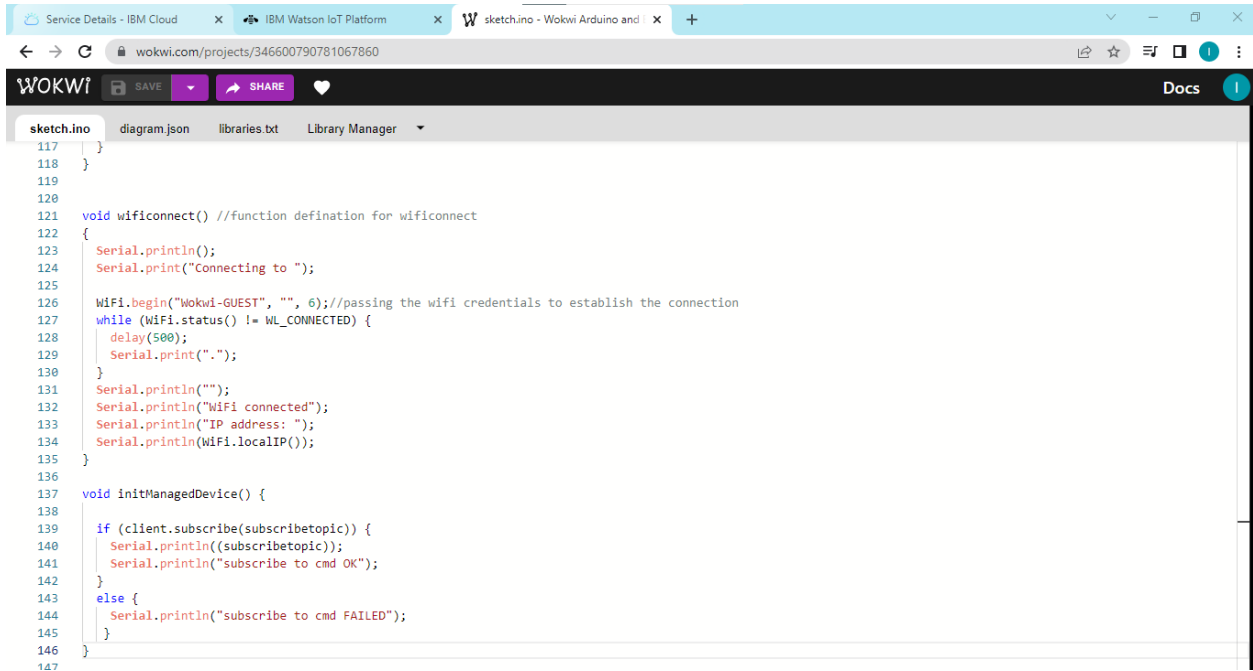
```
30 int echo= 18;
31
32 void setup()
33 {
34
35     Serial.begin(115200);
36     pinMode(trig, OUTPUT);
37     pinMode(echo, INPUT);
38     pinMode(LED, OUTPUT);
39     delay(10);
40
41     wificlient();
42
43     mqttconnect();
44
45 }
46
47 void loop()// Recursive Function
48 {
49
50     delayMicroseconds(10);
51     digitalWrite(trig, LOW);
52     digitalWrite(trig, LOW);
53     digitalWrite(trig,HIGH);
54     float dur= pulseIn(echo,HIGH);
55     float dist = (dur* 0.0343)/2;
56     Serial.print ("Distance in cm : ");
57     Serial.println(dist);
58
59     PublishData(dist);
```



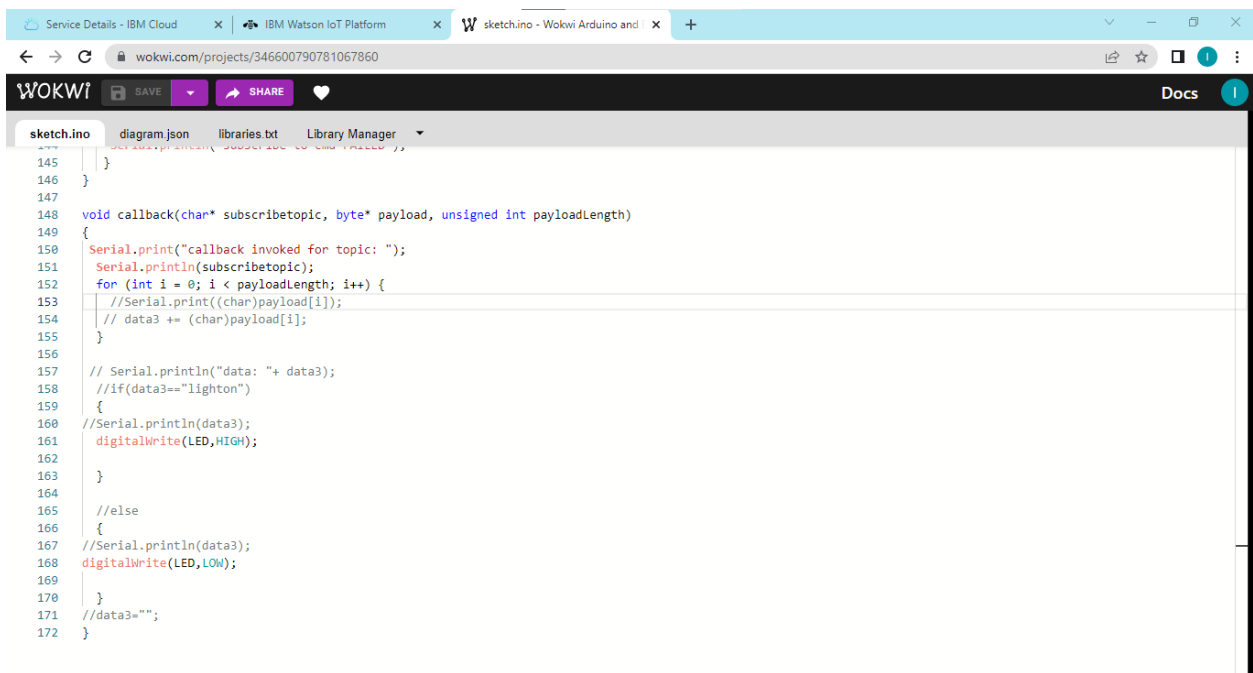
```
59 PublishData(dist);
60
61 delay(1000);
62
63 if (!client.loop()) {
64   mqttconnect();
65 }
66
67 }
68
69 void PublishData(float dist) {
70   mqttconnect();
71
72   String object;
73
74   if (dist<100)
75   {
76     digitalWrite(LED, HIGH);
77     Serial.println("object is near");
78     object = "ALERT! object is near";
79   }
80
81   else
82   {
83     digitalWrite(LED, LOW);
84     Serial.println("no object found");
85     object = "No object found";
86   }
87
88   String payload="{\"distance\":";
```



```
88   String payload="{\"distance\":";
89   payload += dist;
90   payload += ",";
91   payload += object;
92   payload += "\"}";
93
94   Serial.print("Sending payload: ");
95   Serial.println(payload);
96
97   if (client.publish(publishTopic, (char*) payload.c_str()))
98   {
99     Serial.println("Publish ok"); // if it successfully upload
100   }
101   else {
102     Serial.println("Publish failed");
103   }
104 }
105
106 void mqttconnect() {
107   if (!client.connected()) {
108     Serial.print("Reconnecting client to ");
109     Serial.println(server);
110     while (!client.connect(clientId, authMethod, token)) {
111       Serial.print(".");
112       delay(500);
113     }
114   }
115   initManagedDevice();
116   Serial.println();
117 }
```



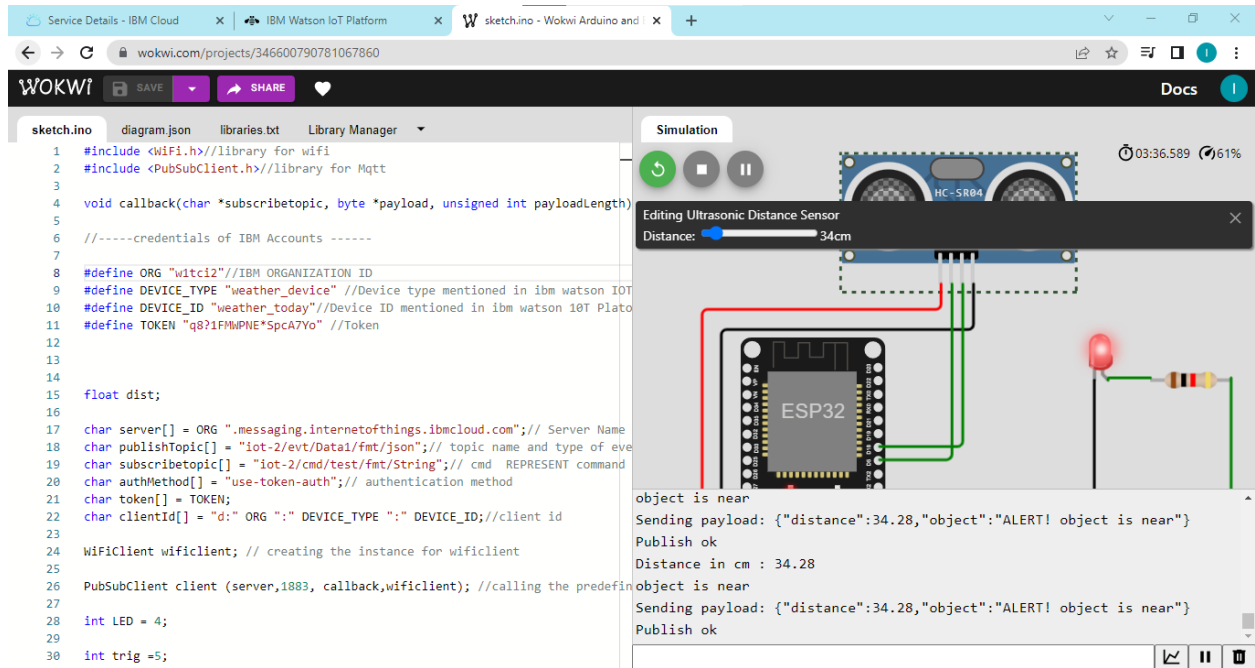
```
117 }
118 }
119
120
121 void wificonnect() //function defination for wificonnect
122 {
123     Serial.println();
124     Serial.print("Connecting to ");
125
126     WiFi.begin("Wokwi-GUEST", "", 6); //passing the wifi credentials to establish the connection
127     while (WiFi.status() != WL_CONNECTED) {
128         delay(500);
129         Serial.print(".");
130     }
131     Serial.println("");
132     Serial.println("WiFi connected");
133     Serial.println("IP address: ");
134     Serial.println(WiFi.localIP());
135 }
136
137 void initManagedDevice() {
138
139     if (client.subscribe(subscribetopic)) {
140         Serial.println(subscribetopic);
141         Serial.println("subscribe to cmd OK");
142     }
143     else {
144         Serial.println("subscribe to cmd FAILED");
145     }
146 }
147
```



```
145     Serial.println("subscribe to cmd FAILED");
146 }
147
148 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
149 {
150     Serial.print("callback invoked for topic: ");
151     Serial.println(subscribetopic);
152     for (int i = 0; i < payloadLength; i++) {
153         //Serial.print((char)payload[i]);
154         // data3 += (char)payload[i];
155     }
156
157     // Serial.println("data: "+ data3);
158     //if(data3=="lighton")
159     {
160         //Serial.println(data3);
161         digitalWrite(LED,HIGH);
162     }
163
164     //else
165     {
166         //Serial.println(data3);
167         digitalWrite(LED,LOW);
168     }
169
170     //data3="";
171 }
172
```

OUTPUT:

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



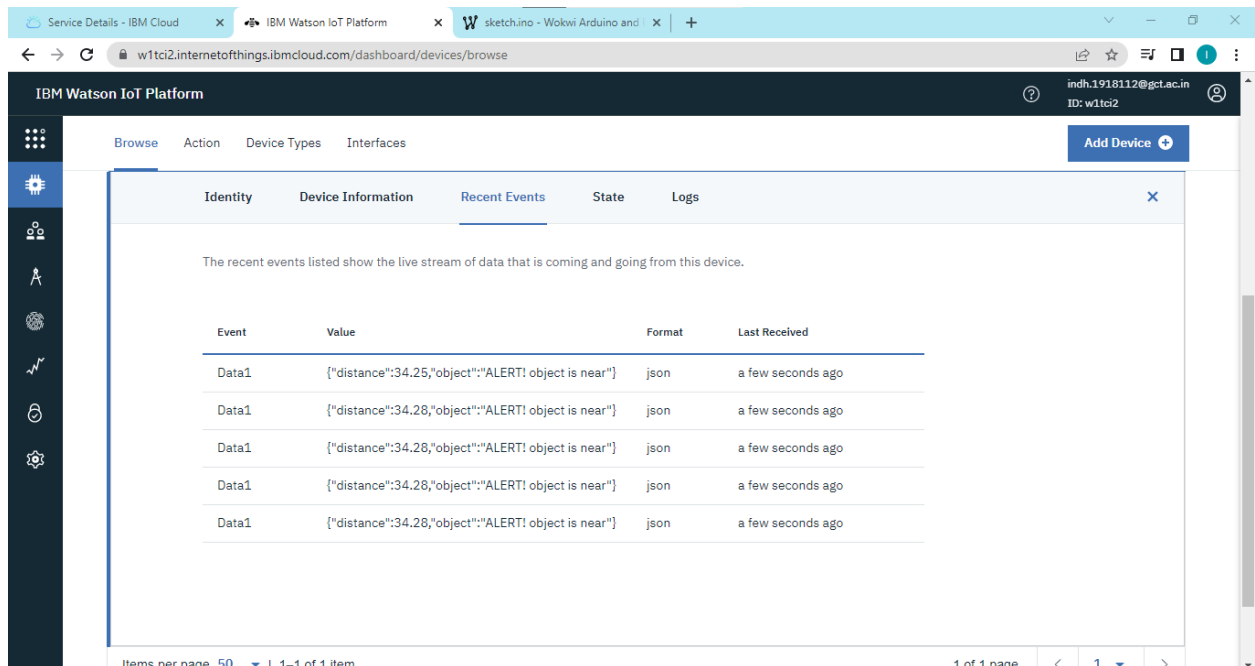
The screenshot shows the Wokwi IDE interface. On the left, the sketch code is displayed, which includes the following key sections:

```
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 "w1tc12" //IBM ORGANIZATION ID
9 #define DEVICE_TYPE "weather_device" //Device type mentioned in ibm watson IOT
10 #define DEVICE_ID "weather_today" //Device ID mentioned in ibm watson IOT
11 #define TOKEN "q8?1FMWPNESpcA7Yo" //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
19 char subscribeTopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command
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
27
28 int LED = 4;
29
30 int trig =5;
```

On the right, the simulation window shows an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor and a red LED. A console window at the bottom displays the following output:

```
object is near
Sending payload: {"distance":34.28,"object":"ALERT! object is near"}
Publish ok
Distance in cm : 34.28
object is near
Sending payload: {"distance":34.28,"object":"ALERT! object is near"}
Publish ok
```

IBM CLOUD IMAGE

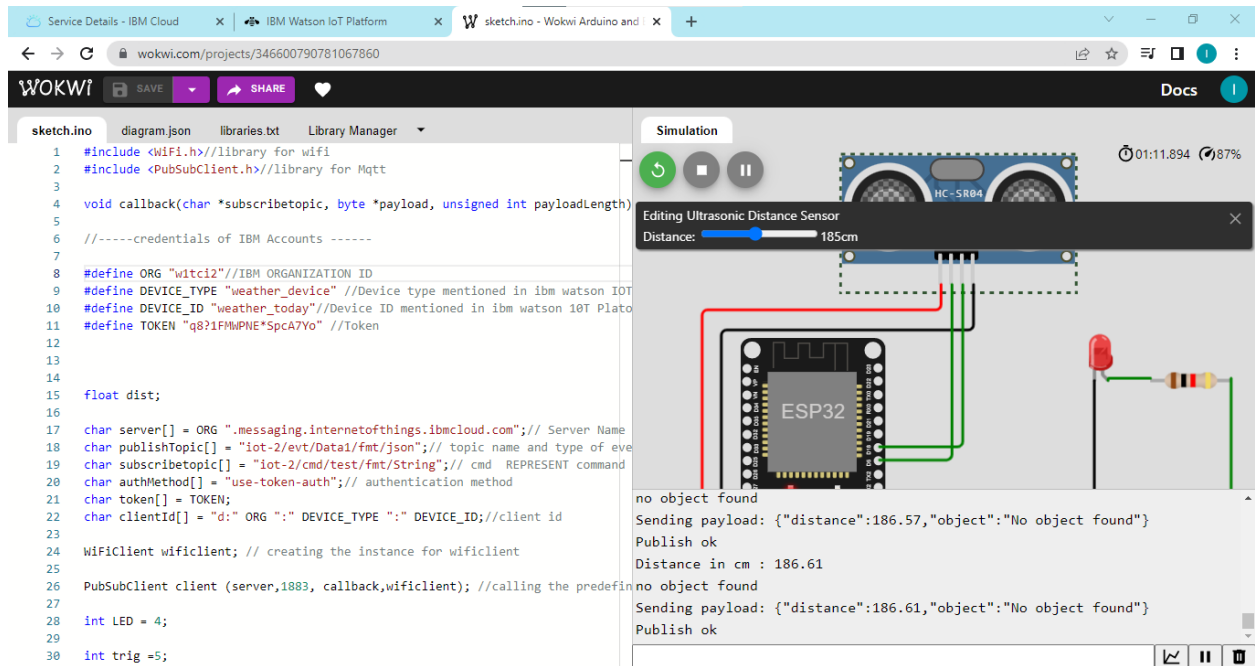


The screenshot shows the IBM Watson IoT Platform dashboard. The 'Recent Events' tab is selected, displaying a table of events for the device 'w1tc12'. The table has the following columns: Event, Value, Format, and Last Received.

Event	Value	Format	Last Received
Data1	{"distance":34.25,"object":"ALERT! object is near"}	json	a few seconds ago
Data1	{"distance":34.28,"object":"ALERT! object is near"}	json	a few seconds ago
Data1	{"distance":34.28,"object":"ALERT! object is near"}	json	a few seconds ago
Data1	{"distance":34.28,"object":"ALERT! object is near"}	json	a few seconds ago
Data1	{"distance":34.28,"object":"ALERT! object is near"}	json	a few seconds ago

At the bottom of the dashboard, it shows 'Items per page 50' and '1 of 1 page'.

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



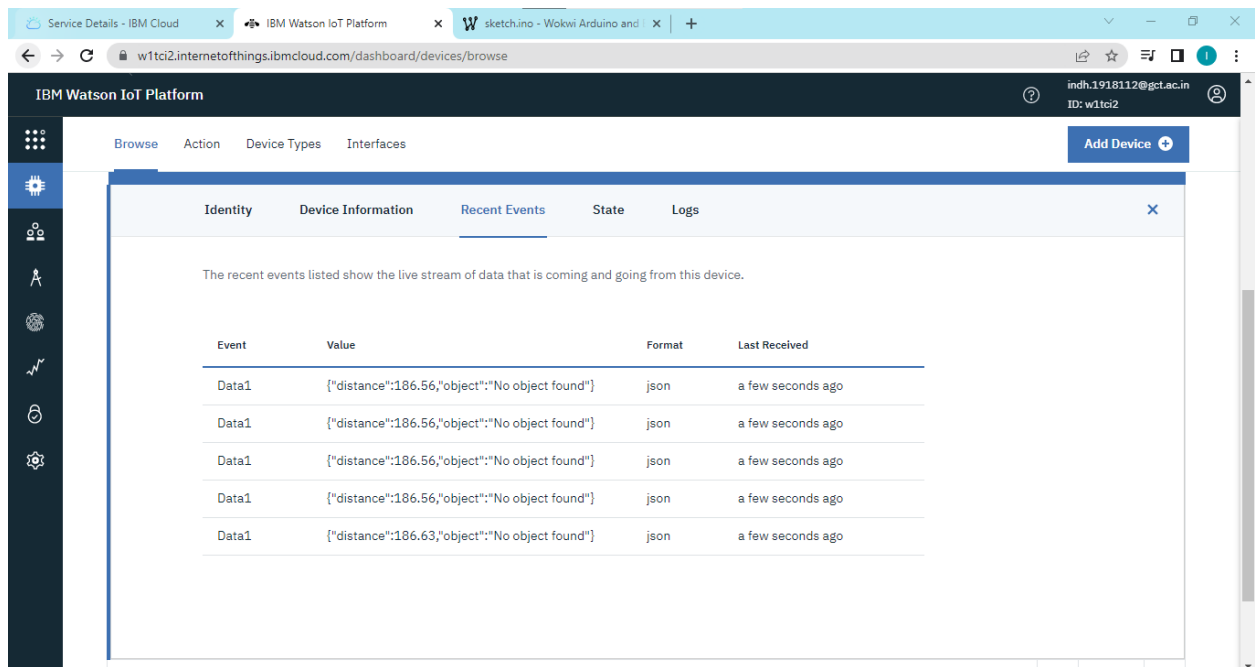
The screenshot shows the Wokwi IDE interface. On the left, the sketch code is displayed, which includes the necessary libraries for WiFi and MQTT, and defines the device type as "weather_device". The code sets up a WiFi client and an MQTT client, and defines a callback function to handle incoming data. The main loop reads the distance from the ultrasonic sensor and publishes the data to the MQTT topic. If the distance is greater than 100 cm, it sends "no object found".

```
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 "w1tc12" //IBM ORGANIZATION ID
9 #define DEVICE_TYPE "weather_device" //Device type mentioned in ibm watson IOT
10 #define DEVICE_ID "weather_today" //Device ID mentioned in ibm watson IOT
11 #define TOKEN "q8?1FHMpNE*SpcA7yo" //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
19 char subscribTopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command
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
27
28 int LED = 4;
29
30 int trig =5;
```

On the right, the simulation window shows an ESP32 microcontroller connected to an ultrasonic sensor (HC-SR04). The sensor's distance is set to 185 cm. The console output shows the following messages:

```
no object found
Sending payload: {"distance":186.57,"object":"No object found"}
Publish ok
Distance in cm : 186.61
no object found
Sending payload: {"distance":186.61,"object":"No object found"}
Publish ok
```

IBM CLOUD IMAGE



The screenshot shows the IBM Watson IoT Platform dashboard. The 'Recent Events' tab is selected, displaying a table of events received from the device. The table has four columns: Event, Value, Format, and Last Received. The events are listed as follows:

Event	Value	Format	Last Received
Data1	{"distance":186.56,"object":"No object found"}	json	a few seconds ago
Data1	{"distance":186.56,"object":"No object found"}	json	a few seconds ago
Data1	{"distance":186.56,"object":"No object found"}	json	a few seconds ago
Data1	{"distance":186.56,"object":"No object found"}	json	a few seconds ago
Data1	{"distance":186.63,"object":"No object found"}	json	a few seconds ago