

## ASSIGNMENT-4

### DISTANCE DETECTION USING ULTRASONIC SENSOR

Date	31 October 2022
Team ID	PNT2022TMID508060
Name	DHANSA R
Student Roll Number	953419104018
Maximum Marks	2 Marks

#### Question1 :

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.

#### WOKWI LINK :

<https://wokwi.com/projects/305566932847821378>

#### CODE :

```
1 #include <SPI.h> //library for spi
2 #include <PubSubClient.h> //library for mqtt
3
4
5 void callback(char* topic, byte* payload, unsigned int payloadLength);
6
7 //-----credentials of IBM Accounts-----
8
9 #define ORG "ibm" //IBM ORGANIZATION ID
10 #define DEVICE_TYPE "ULTRASONIC" //Device type mentioned in the Watson IoT Platform
11 #define DEVICE_ID "DISTANCEDETECT" //Device ID mentioned in the Watson IoT Platform
12 #define TOKEN "wskc398j25agv4k8tc" //token
13 String data;
14 float dist;
15
16
17 //----- Customise the above values -----
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
19 char publishTopic[] = "iot-2/evt/data/fmt/json"; // Topic name and type of event perform and format in which data to be send
20 char subscribeTopic[] = "iot-2/cmd/test/fmt/string"; // cmd REPRESENT COMAND TYPE AND COMMAND IS TEST OF FORMAT STRING
21 char authMethod[] = "use-token-auth"; // authentication method
22 char token[] = TOKEN;
23 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
24
25
26 //-----
27 WiFiClient wifiClient; // creating the instance for wifiClient
28 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client id by passing parameter like server id, port and wifi credentials
29
30 int led = 4;
31 int trig = 5;
32 int echo = 18;
33 void setup()
34 {
35   Serial.begin(115200);
```

```
36  pinMode(trig,OUTPUT);
37  pinMode(echo,INPUT);
38  pinMode(LED, OUTPUT);
39  delay(10);
40  wificonnect();
41  mqttconnect();
42  }
43  void loop()// Recursive Function
44  {
45
46      digitalWrite(trig,LOW);
47      digitalWrite(trig,HIGH);
48      delayMicroseconds(10);
49      digitalWrite(trig,LOW);
50      float dur = pulseIn(echo,HIGH);
51      float dist = (dur * 0.0343)/2;
52      Serial.print ("Distance in cm");
53      Serial.println(dist);
54
55
56      PublishData(dist);
57      delay(1000);
58      if (!client.loop()) {
59          mqttconnect();
60      }
61  }
62
63
64
65  /*.....retrieving to Cloud.....*/
66
67  void PublishData(float dist) {
68      mqttconnect();//function call for connecting to ibm
69      /*
70      |   creating the String in in form JSon to update the data to ibm cloud
```

```

70 | | creating the String in in form JSON to update the data to ibm cloud
71 | |
72 String object;
73 if (dist <100)
74 {
75   digitalWrite(LED,HIGH);
76   Serial.println("object is near");
77   object = "Near";
78 }
79 else
80 {
81   digitalWrite(LED,LOW);
82   Serial.println("no object found");
83   object = "No";
84 }
85
86 String payload = "{\"distance\":";
87 payload += dist;
88 payload += "," " \"object\":";
89 payload += object;
90 payload += "\"}";
91
92
93 Serial.print("Sending payload: ");
94 Serial.println(payload);
95
96
97
98

```

esp32-blink.ino • diagram.json • libraries.txt • Library Manager

```

99
100 if (client.publish(publishTopic, (char*) payload.c_str())) {
101   Serial.println("Publish ok");// if it successfully upload data on the cloud then it will print publish ok in serial monitor or else it will print publish failed
102 } else {
103   Serial.println("Publish failed");
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   }
118 }
119 void wificonnect() //function definition for wificonnect
120 {
121   Serial.println();
122   Serial.print("Connecting to ");
123
124   WiFi.begin("Wesley-DESI", "", 0);//passing the wifi credentials to establish the connection
125   while (WiFi.status() != WL_CONNECTED) {
126     delay(500);
127     Serial.print(".");
128   }
129   Serial.println("");
130   Serial.println("WiFi connected");
131   Serial.println("IP address: ");
132   Serial.println(WiFi.localIP());

```

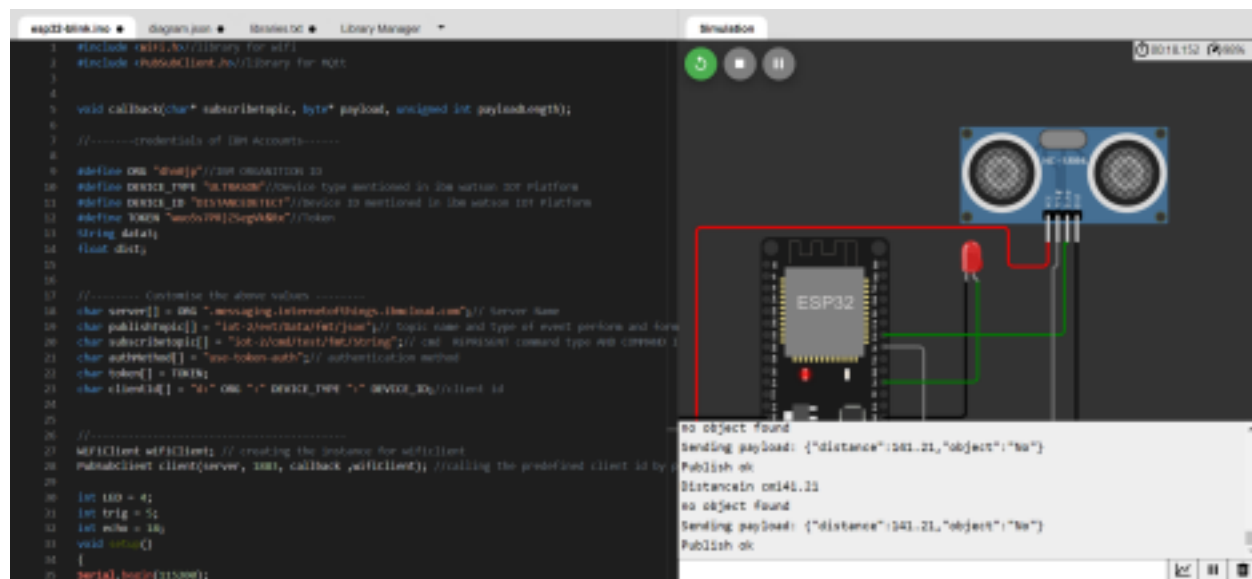
```
123
124   WiFi.begin("Wokwi-GUEST", "", 6); //passing the wifi credentials to establish the connection
125   while (WiFi.status() != WL_CONNECTED) {
126       delay(500);
127       Serial.print(".");
128   }
129   Serial.println("");
130   Serial.println("WiFi connected");
131   Serial.println("IP address: ");
132   Serial.println(WiFi.localIP());
133 }
134
135 void initManagedDevice() {
136     if (client.subscribe(subscribetopic)) {
137         Serial.println((subscribetopic));
138         Serial.println("subscribe to cmd OK");
139     } else {
140         Serial.println("subscribe to cmd FAILED");
141     }
142 }
143
144 void callback(char* subscribetopic, byte* payload, unsigned int payloadlength)
145 {
146
147     Serial.print("callback invoked for topic: ");
148     Serial.println(subscribetopic);
149     for (int i = 0; i < payloadlength; i++) {
150         //Serial.print((char)payload[i]);
151         data3 += (char)payload[i];
152     }
153
154     // Serial.println("data: "+ data3);
155     // if(data3=="Near")
156     // {
157     // Serial.println(data3);
158     // }
159 }
```

```

esp32-blink.ino  ●  diagram.json  ●  libraries.txt  ●  Library Manager  ▼
142  }
143
144  void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
145  {
146
147      Serial.print("callback invoked for topic: ");
148      Serial.println(subscribetopic);
149      for (int i = 0; i < payloadLength; i++) {
150          //Serial.print((char)payload[i]);
151          data3 += (char)payload[i];
152      }
153
154      //  Serial.println("data: "+ data3);
155      //  if(data3=="Near")
156      //  {
157      //  Serial.println(data3);
158      //  digitalWrite(LED,HIGH);
159
160      //  }
161
162      //  else
163      //  {
164      //  Serial.println(data3);
165      //  digitalWrite(LED,LOW);
166
167      //  }
168      data3="";
169
170
171  }

```

OUTPUT:



```

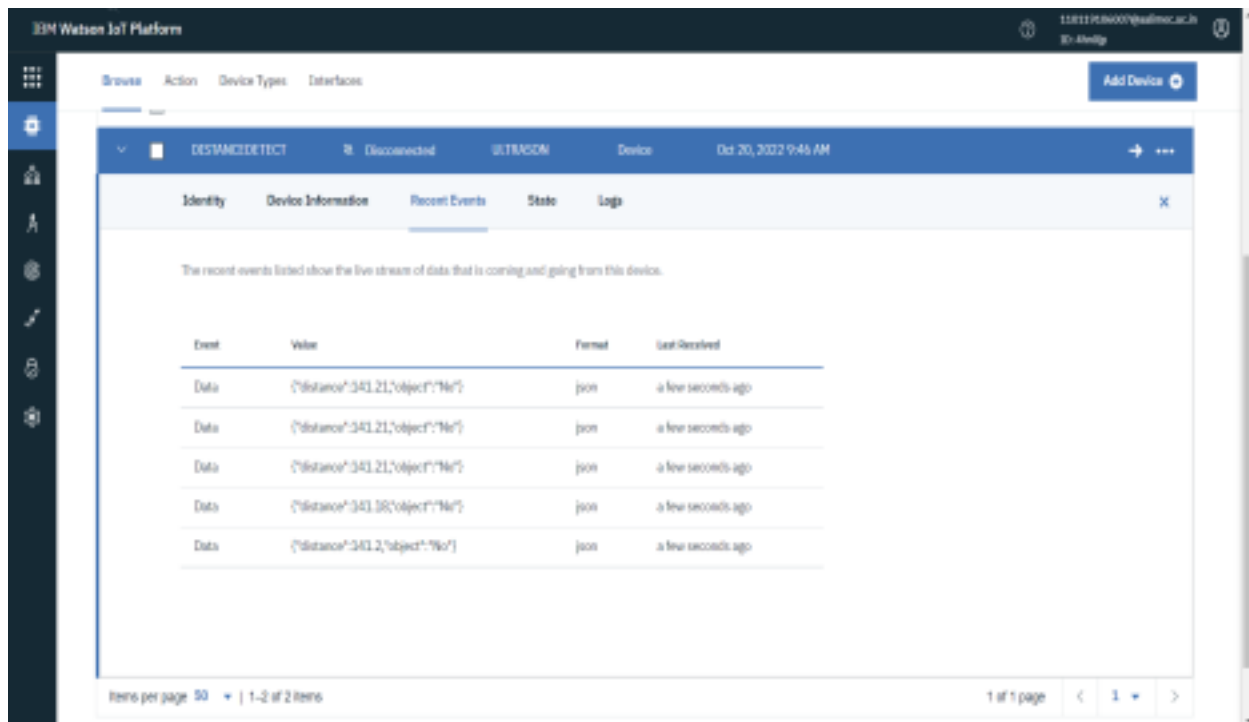
esp32-blink.ino  ●  diagram.json  ●  libraries.txt  ●  Library Manager  ▼
1  #include <WiFi.h> //library for wifi
2  #include <PubSubClient.h> //library for mqtt
3
4
5  void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
6
7  //-----credentials of IBM Accounts-----
8
9  #define DM "dmgp" //IBM ORG/ID/IDEM ID
10 #define DEVICE_TYPE "ULTRASONIC" //device type mentioned in the action bot platform
11 #define DEVICE_ID "DISTANCEBOT100" //device id mentioned in the action bot platform
12 #define XORG "moox79" //orgId
13 #define XORG "moox79" //orgId
14 #define XORG "moox79" //orgId
15 #define XORG "moox79" //orgId
16
17 //----- Customize the above values -----
18 #define SERVER "messaging.internetofthings.ibmcloud.com" // Server Name
19 #define PUBLISH_TOPIC "iot-2/evt/data/evt/loc" // topic name and type of event perform and how
20 #define SUBSCRIBE_TOPIC "iot-2/cmd/test/loc/string" // cmd -> IBM Watson command type web connect
21 #define AUTH_TOKEN "moox79" // authentication token
22 #define CLIENT_ID "IoT-ORG-" DEVICE_TYPE "-" DEVICE_ID //client id
23
24
25 //-----
26 #define MQTT_CLIENT mqttClient // creating the instance for mqttClient
27 #define MQTT_CLIENT mqttClient // calling the predefined client id by
28
29
30 int led = 4;
31 int trig = 5;
32 int echo = 10;
33
34 void setup()
35 {
36     Serial.begin(115200);

```

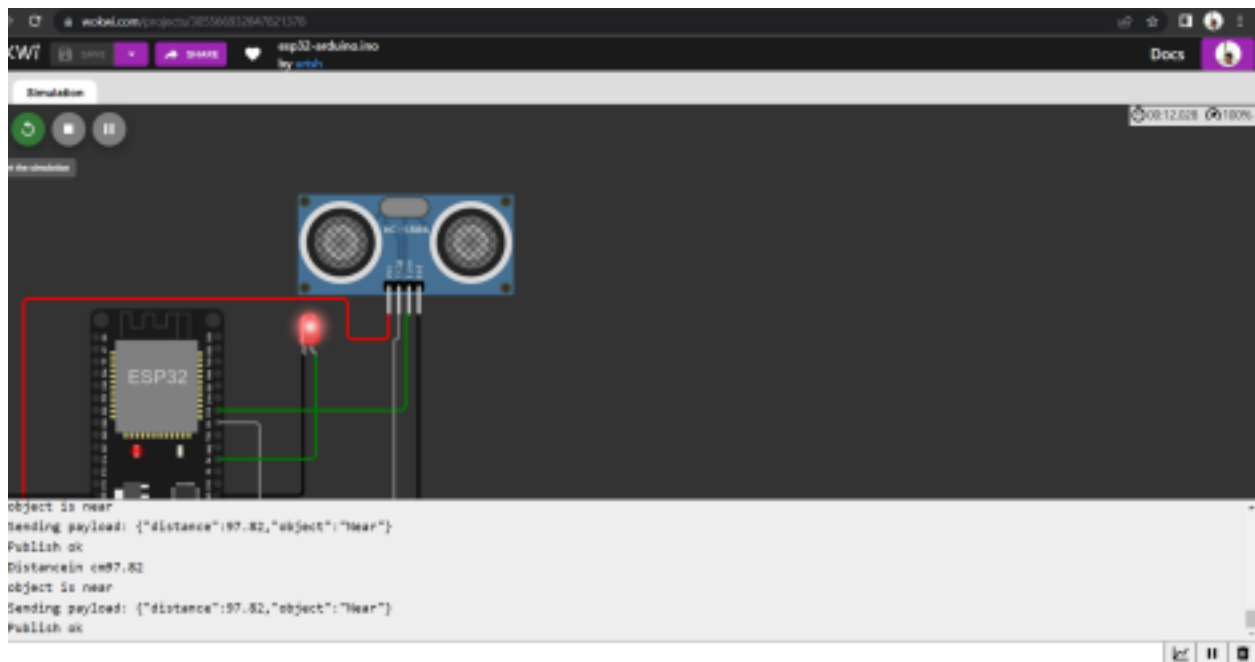
Simulation

no object found  
 Sending payload: {"distance":101.21,"object":"No"}  
 Publish ok  
 Distancein cm101.21  
 no object found  
 Sending payload: {"distance":101.21,"object":"No"}  
 Publish ok

Data send to the IBM cloud device when the object is far



when object is near to the ultrasonic sensor



Data sent to the IBM Cloud Device when the object is near

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various functions. The main content area shows a device named 'DESTINATOR-TEXT' with a status of 'Disconnected' and a type of 'ULTRASONIC'. Below this, a tabbed interface is visible with tabs for 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, displaying a message: 'The recent events listed show the live stream of data that is coming and going from this device.' Below this message is a table with the following data:

Event	Value	Format	Last Received
Data	{"distance":79.66,"object":"Near"}	json	a few seconds ago
Data	{"distance":79.64,"object":"Near"}	json	a few seconds ago
Data	{"distance":79.66,"object":"Near"}	json	a few seconds ago
Data	{"distance":79.64,"object":"Near"}	json	a few seconds ago
Data	{"distance":79.66,"object":"Near"}	json	a few seconds ago

At the bottom of the interface, there is a pagination control showing 'Items per page: 50' and '1 of 1 page'.

<https://wokwi.com/projects/305566932847821378>