

ASSIGNMENT-4

DISTANCE DETECTION USING ULTRASONIC SENSOR

Date	16 November 2022
Team ID	PNT2022TMID34961
Name	Pakkiya S
Register No	962819205023
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/347194582937633363>

```
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 ORG "4hn0jp"//IBM ORGANIZATION ID
10 #define DEVICE_TYPE "ULTRASON"//Device type mentioned in ibm watson IOT Platform
11 #define DEVICE_ID "DISTANCEDETECT"//Device ID mentioned in ibm watson IOT Platform
12 #define TOKEN "wuo5s7PR)ZSegV&Rx"//Token
13 String data3;
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 command 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,portand wificredential
29
30 int LED = 4;
31 int trig = 5;
32 int echo = 18;
33 void setup()
34 {
35   Serial.begin(115200);
```

CODE :

```
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 ("Distancein 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
```

```

69  |
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 ▼
98
99  if (client.publish(publishTopic, (char*) payload.c_str())) {
100      Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print publish ok in Serial monitor or else it will print publish failed
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      }
118  }
119  void wificonnect() //function defination for wificonnect
120  {
121      Serial.println();
122      Serial.print("Connecting to ");
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());

```

```
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     // if(data3=="Near") {
159     //     Serial.println("Near");
160     // }
```

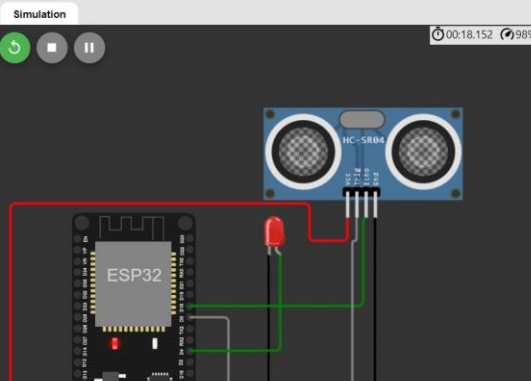
```
esp32-blink.ino • diagram.json • libraries.txt • Library Manager
142 }
143
144 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
145 {
146
147     Serial.print("callback invoked for topic: ");
148     Serial.println(subscribetopic);
149     for (int i = 0; i < payloadLength; 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     // else
162     // {
163     // Serial.println(data3);
164     // digitalWrite(LED,LOW);
165     // }
166
167     data3="";
168 }
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 ORG "4hnejp" //IBM ORGANIZATION ID
10 #define DEVICE_TYPE "ULTRASON" //Device type mentioned in ibm watson IOT Platform
11 #define DEVICE_ID "DISTANCEDETECT" //Device ID mentioned in ibm watson IOT Platform
12 #define TOKEN "wuo5s7PR)ZSegV&R&x" //Token
13 String data3;
14 float dist;
15
16 //----- Customise the above values -----
17
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
19 char publishTopic[] = "iot-2/dev/ota/fmt/json"; // topic name and type of event perform and form
20 char subscribetopic[] = "iot-2/cmd/text/fmt/string"; // cmd - REPRESENT command type AND COMMAND ID
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 WiFiClient wificlient; // creating the instance for wificlient
27 PubSubClient client(server, 1883, callback, wificlient); //calling the predefined client id by p
28
29 int LED = 4;
30 int trig = 5;
31 int echo = 18;
32 void setup()
33 {
34     Serial.begin(115200);
35 }
```

Simulation

00:18.152 98%



no object found
Sending payload: {"distance":141.21,"object":"No"}
Publish ok
Distance in cm 141.21
no object found
Sending payload: {"distance":141.21,"object":"No"}
Publish ok

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

The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons. The main content area displays details for a device named 'DISTANCEDETECT', which is 'Disconnected' and of type 'ULTRASON'. The 'Recent Events' tab is selected, showing a table of data events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. The events are JSON objects containing distance and object status. The status is 'No' for all events shown.

Event	Value	Format	Last Received
Data	{"distance":141.21,"object":"No"}	json	a few seconds ago
Data	{"distance":141.21,"object":"No"}	json	a few seconds ago
Data	{"distance":141.21,"object":"No"}	json	a few seconds ago
Data	{"distance":141.18,"object":"No"}	json	a few seconds ago
Data	{"distance":141.2,"object":"No"}	json	a few seconds ago

when object is near to the ultrasonic sensor

The screenshot shows a Wokwi simulation of an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor. The sensor's red LED is lit, indicating it is active. The simulation output at the bottom shows the following sequence of events:

```
object is near
Sending payload: {"distance":97.82,"object":"Near"}
Publish ok
Distancein cm97.82
object is near
Sending payload: {"distance":97.82,"object":"Near"}
Publish ok
```

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

