

## ASSIGNMENT- 4

### DISTANCE DETECTION USING ULTRASONIC SENSOR

Date	20 October 2022
Team ID	PNT2022TMID32971
Name	MEDHA S.P
Student Roll Number	820419106033
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.

#### CODE :

```
1 #include <Arduino.h> //for arduino
2 #include <Wire.h> //for I2C
3
4 // Define pins for the sensor
5 const int TRIG_PIN = 9; // Trigger pin
6 const int ECHO_PIN = 10; // Echo pin
7
8 // Define variables
9 int distance; // Distance in cm
10
11 // Define the server URL
12 const char* serverURL = "https://api.cloud.ibm.com";
13
14 // Define the API key
15 const char* apiKey = "your-api-key";
16
17 // Define the device ID
18 const char* deviceId = "your-device-id";
19
20 // Define the event name
21 const char* eventName = "distance_detection";
22
23 // Define the event data
24 const char* eventData = "{\"distance\": " + String(distance) + "}";
25
26 // Define the headers
27 const char* headers = "Content-Type: application/json";
28
29 // Define the callback function
30 void callback(char* serverURL, byte* payload, unsigned int payloadLength);
31
32 // Define the main function
33 void setup() {
34   pinMode(TRIG_PIN, OUTPUT);
35   pinMode(ECHO_PIN, INPUT);
36   Serial.begin(115200);
37 }
38
39 void loop() {
40   // Send a pulse to the trigger pin
41   digitalWrite(TRIG_PIN, LOW);
42   delayMicroseconds(2);
43   digitalWrite(TRIG_PIN, HIGH);
44   delayMicroseconds(10);
45   digitalWrite(TRIG_PIN, LOW);
46
47   // Measure the time taken for the echo to return
48   long duration = pulseIn(ECHO_PIN, HIGH);
49
50   // Calculate the distance
51   distance = duration * 0.034 / 2;
52
53   // Send the data to the server
54   httpPost(serverURL, apiKey, deviceId, eventName, eventData);
55
56   // Delay for 100ms
57   delay(100);
58 }
59
60 // Function to send data to the server
61 void httpPost(char* serverURL, const char* apiKey, const char* deviceId, const char* eventName, const char* eventData) {
62   // Create an HTTP client
63   HTTPClient http;
64   http.begin(serverURL);
65   http.addHeader("Content-Type", "application/json");
66   http.addHeader("Authorization", "Basic " + String(apiKey));
67   http.POST(deviceId + "/" + eventName + "/" + eventData);
68   http.end();
69 }
```

```
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 the 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

```

```

esp32-01rk-nc • setup.py • browser • library manager
99
100 if (client.publish(topic, (char*) payload.c_str())) {
101     Serial.println("publish ok"); // if it successfully send data to the cloud then it will print publish ok in serial monitor so also it will print publish failed
102 } else {
103     Serial.println("publish failed");
104 }
105 }
106
107 void setup() {
108     if (client.connected()) {
109         Serial.print("Disconnecting client to ");
110         Serial.println(server);
111         while (!client.connect(topic, authmethod, token)) {
112             Serial.print("-");
113             delay(500);
114         }
115
116         initMcuDevice();
117         Serial.println();
118     }
119 }
120
121 void setup() //function definition for setup()
122 {
123     Serial.println();
124     Serial.println("connecting to ");
125
126     WiFi.begin("wifidroid", ""); //assing the wifi credentials to variables (the connection
127     while (WiFi.status() != WL_CONNECTED) {
128         delay(500);
129         Serial.print(".");
130     }
131
132     Serial.println("");
133     Serial.println("WiFi connected");
134     Serial.println("IP address: ");
135     Serial.println(WiFi.localIP());

```

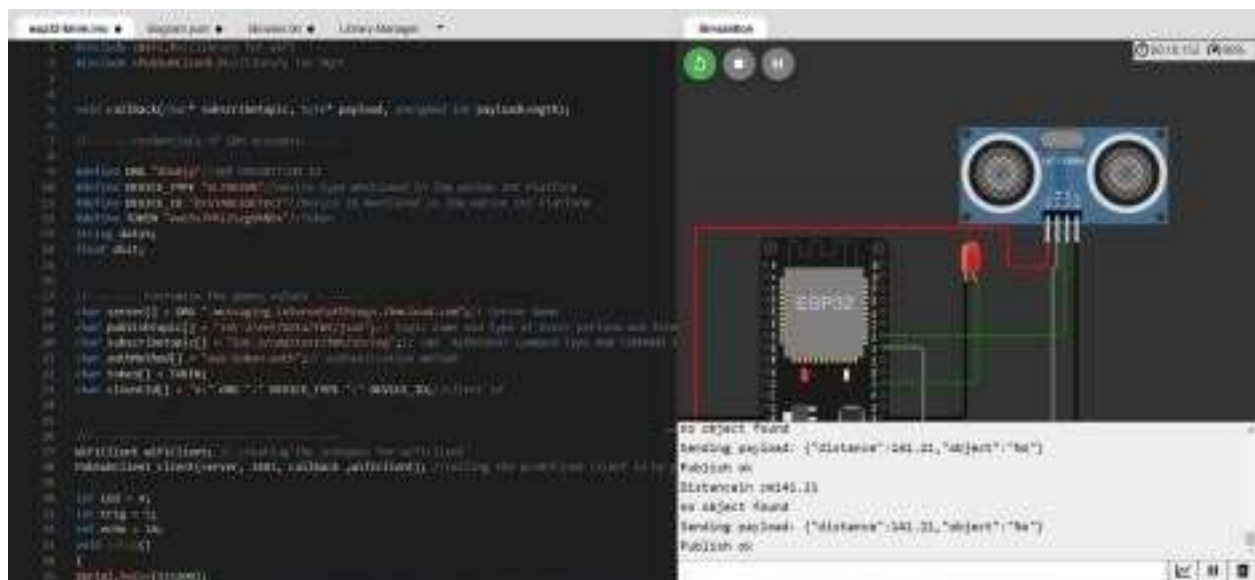
```
123
124   WiFi.begin("Wokwi-GUEST", ""); //passing the wifi credentials to establish the connection
125   while (WiFi.status() != WL_CONNECTED) {
126       delay(500);
127       Serial.print(".");
128   }
129   Serial.println("");
130   Serial.println("WiFi connected");
131   Serial.println("IP address: ");
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=="hear")
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     // else
162     // {
163     //     Serial.println(data3);
164     //     digitalWrite(LED, LOW);
165     // }
166
167     data3="";
168 }
169
170
171

```

OUTPUT:

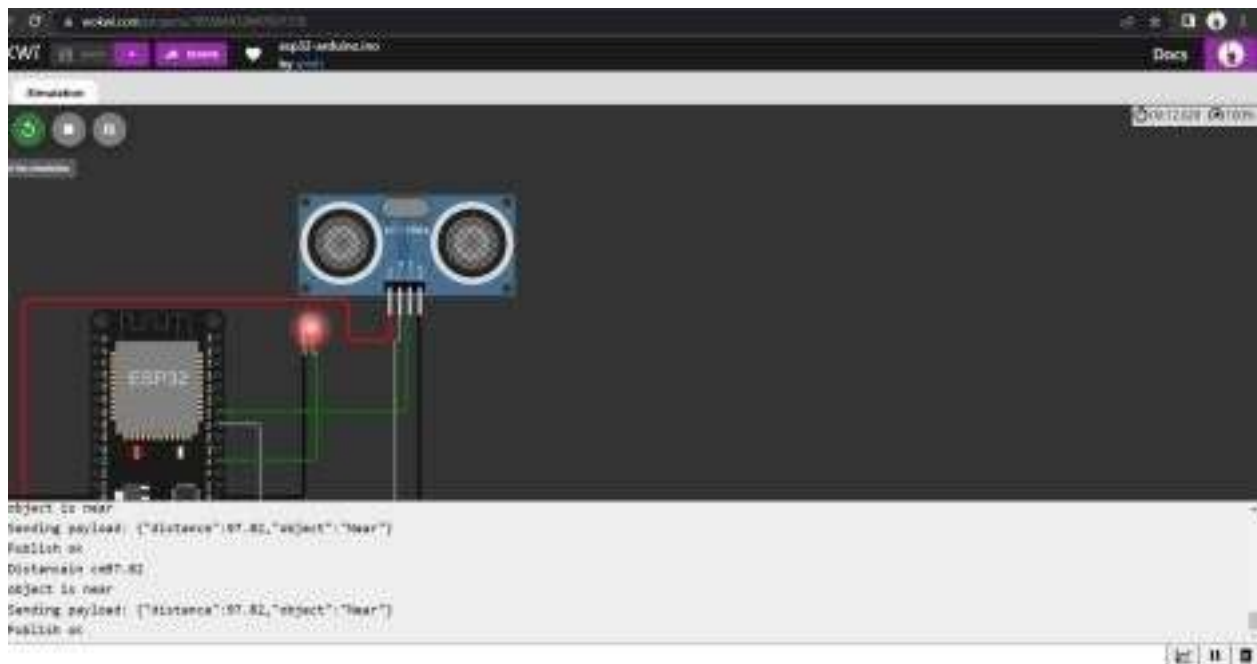


Data send to the IBM cloud device when the object is far

The screenshot shows the IBM Watson IoT Platform interface. The top navigation bar includes 'Devices', 'Alerts', 'Device Types', and 'Integrations'. A sidebar on the left contains icons for various functions. The main content area displays details for a specific device, 'DEVM451E7C1', which is an 'Arduino Uno' with IP address '192.168.1.100'. The 'Recent Events' tab is selected, showing a table of events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. There are five rows of data, all with a 'Data' event type and a 'json' format, received 'a few seconds ago'. The bottom of the interface shows 'Items per page: 50' and '1 of 1 page'.

Event	Value	Format	Last Received
Data	{"distance":341.21,"object":"Near"}	json	a few seconds ago
Data	{"distance":341.21,"object":"Near"}	json	a few seconds ago
Data	{"distance":341.21,"object":"Near"}	json	a few seconds ago
Data	{"distance":341.21,"object":"Near"}	json	a few seconds ago
Data	{"distance":341.21,"object":"Near"}	json	a few seconds ago

When object is near to the ultrasonic sensor



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

