

ASSIGNMENT-4

DISTANCE DETECTION USING ULTRASONIC SENSOR

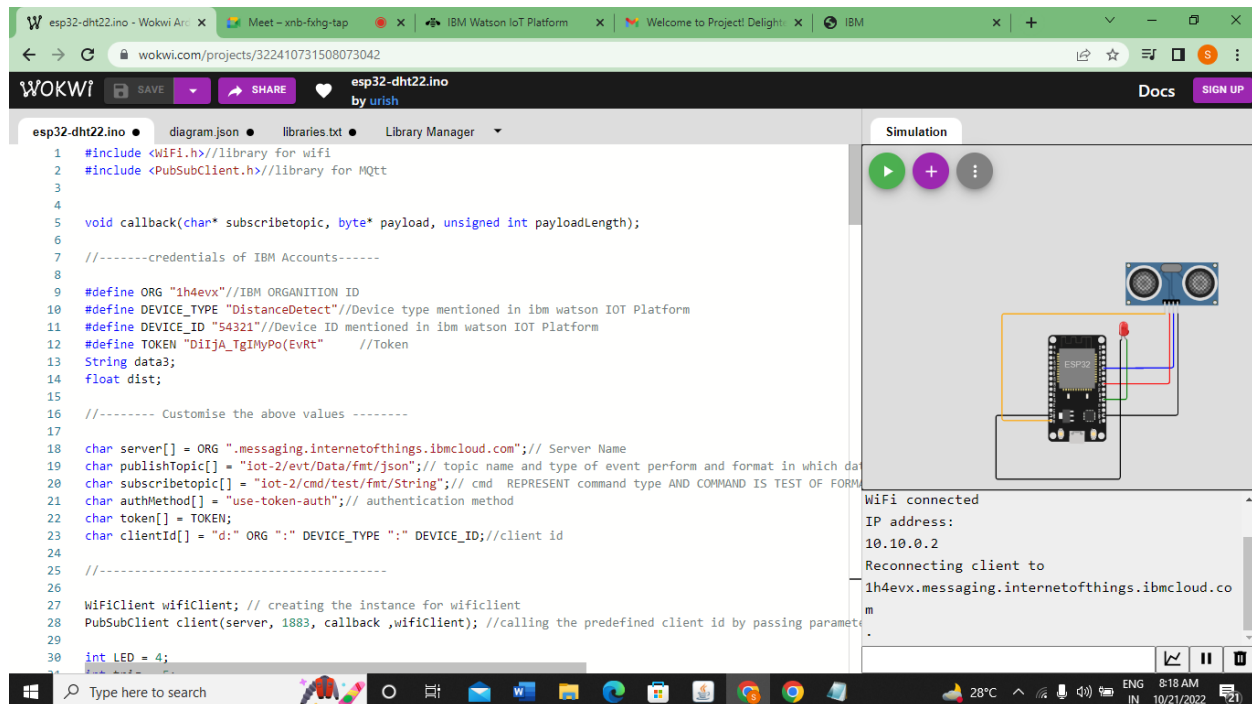
Team ID	PNT2022TMID27689
Name	A.Shobika
Student Roll Number	311419104073
Maximum Marks	2 Marks

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.

CODE:

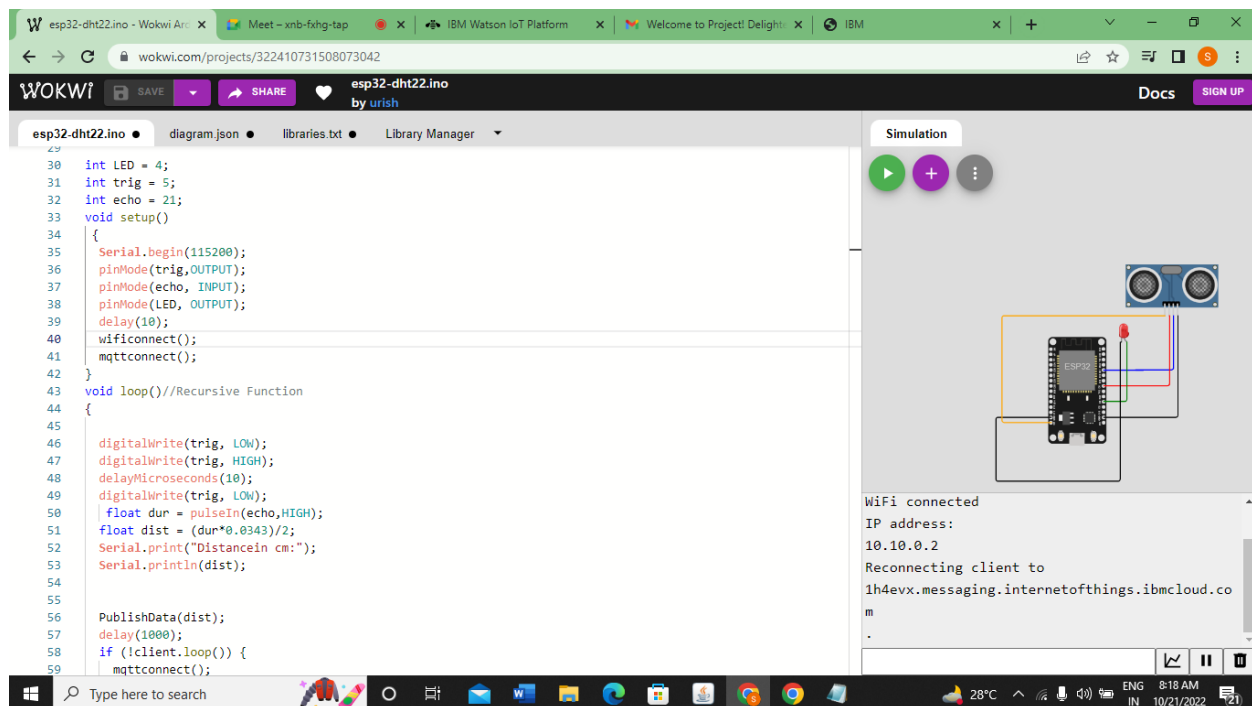


Wokwi IDE interface showing the code for `esp32-dht22.ino`. The code includes headers for WiFi and MQTT, defines credentials and device information, and sets up a client for IBM Watson IoT Platform.

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3
4 void callback(char* subscribtopic, byte* payload, unsigned int payloadLength);
5
6 //-----credentials of IBM Accounts-----
7
8 #define ORG "1h4evx" //IBM ORGANITION ID
9 #define DEVICE_TYPE "DistanceDetect" //Device type mentioned in ibm watson IOT Platform
10 #define DEVICE_ID "54321" //Device ID mentioned in ibm watson IOT Platform
11 #define TOKEN "DlIjA_TgIMyPo(EvRt" //Token
12 String data3;
13 float dist;
14
15 //----- Customise the above values -----
16
17 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
18 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform and format in which data
19 char subscribtopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type AND COMMAND IS TEST OF FORM
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 //-----
25
26 WiFiClient wifiClient; // creating the instance for wifi client
27 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client id by passing paramete
28
29 int LED = 4;
```

Simulation window shows the circuit diagram and the following output:

```
WiFi connected
IP address:
10.10.0.2
Reconnecting client to
1h4evx.messaging.internetofthings.ibmcloud.co
m
```



Wokwi IDE interface showing the code for `esp32-dht22.ino`. The code includes headers for WiFi and MQTT, defines credentials and device information, and sets up a client for IBM Watson IoT Platform. The code also includes a setup function for the LED and trig pin, and a loop function that reads the DHT22 sensor data and publishes it to the MQTT broker.

```
30 int LED = 4;
31 int trig = 5;
32 int echo = 21;
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   digitalWrite(trig, LOW);
46   digitalWrite(trig, HIGH);
47   delayMicroseconds(10);
48   digitalWrite(trig, LOW);
49   float dur = pulseIn(echo, HIGH);
50   float dist = (dur*0.0343)/2;
51   Serial.print("Distance in cm:");
52   Serial.println(dist);
53
54   PublishData(dist);
55   delay(1000);
56   if (!client.loop()) {
57     mqttConnect();
58   }
59 }
```

Simulation window shows the circuit diagram and the following output:

```
WiFi connected
IP address:
10.10.0.2
Reconnecting client to
1h4evx.messaging.internetofthings.ibmcloud.co
m
```

Wokwi IoT Platform interface showing a project named "esp32-dht22.ino" by urish. The code is displayed in the editor, and the simulation window shows the device connected to WiFi and the IP address 10.10.0.2. The code includes a loop that checks for an object near the LED and publishes data to IBM Cloud via MQTT.

```
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
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   String payload = "{\"distance\": ";
```

Simulation window output:

```
WiFi connected
IP address:
10.10.0.2
Reconnecting client to
1h4evx.messaging.internetofthings.ibmcloud.co
m
```

Wokwi IoT Platform interface showing a project named "esp32-dht22.ino" by urish. The code editor displays the following C++ code:

```
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
99 if (client.publish(publishTopic, (char*) payload.c_str())) {
100 | Serial.println("Publish ok");// if it successfully upload data on the cloud then it will print publish ok
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 }
```

The right sidebar shows the "Simulation" tab with a visual representation of the ESP32 and a terminal output:

```
WiFi connected
IP address:
10.10.0.2
Reconnecting client to
1h4evx.messaging.internetofthings.ibmcloud.co
m
```

Wokwi IoT Platform interface showing the same project "esp32-dht22.ino" by urish. The code editor displays the following C++ code:

```
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("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 | }
```

The right sidebar shows the "Simulation" tab with a visual representation of the ESP32 and a terminal output:

```
WiFi connected
IP address:
10.10.0.2
Reconnecting client to
1h4evx.messaging.internetofthings.ibmcloud.co
m
```

Wokwi IoT Platform interface showing a project named "esp32-dht22.ino" by urish. The code editor displays the following C++ code:

```
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=="lighton")
156   // {
157   //Serial.println(data3);
158   //digitalWrite(LED,HIGH);
159   // }
160
161   // else
162   // {
163   //   Serial.println(data3);
164   //   digitalWrite(LED,LOW);
165   // }
166   data3="";
167
168
169 }
```

The simulation window shows a WiFi icon and the following status:

Wifi connected
IP address:
10.10.0.2
Reconnecting client to
1h4evx.messaging.internetofthings.ibmcloud.co
m

Wokwi IoT Platform interface showing the same project "esp32-dht22.ino" by urish. The code editor displays the following C++ code:

```
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=="lighton")
156   // {
157   //Serial.println(data3);
158   //digitalWrite(LED,HIGH);
159   // }
160
161   // else
162   // {
163   //   Serial.println(data3);
164   //   digitalWrite(LED,LOW);
165   // }
166   data3="";
167
168
169 }
170 }
```

The simulation window shows a WiFi icon and the following status:

Wifi connected
IP address:
10.10.0.2
Reconnecting client to
1h4evx.messaging.internetofthings.ibmcloud.co
m

OUTPUT:

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various IoT functions. The main content area shows a list of devices, with the second device (ID 54321) selected. This device is in a 'Disconnected' state and has a 'DistanceDetect' type. Below the device list, a 'Recent Events' tab is active, showing a stream of data events. The events are listed in a table with columns for 'Event', 'Value', 'Format', and 'Last Received'. The events show a sequence of distance measurements (403.47 and 403.49) and status updates ('No'). A status message at the bottom indicates '1 Simulation running'.

Event	Value	Format	Last Received
Data	{"distance":403.47,"object":"","No"}	json	a few seconds ago
Data	{"distance":403.49,"object":"","No"}	json	a few seconds ago
Data	{"distance":403.49,"object":"","No"}	json	a few seconds ago
Data	{"distance":403.47,"object":"","No"}	json	a few seconds ago
Data	{"distance":403.49,"object":"","No"}	json	a few seconds ago

1 Simulation running