

ASSIGNMENT – 4

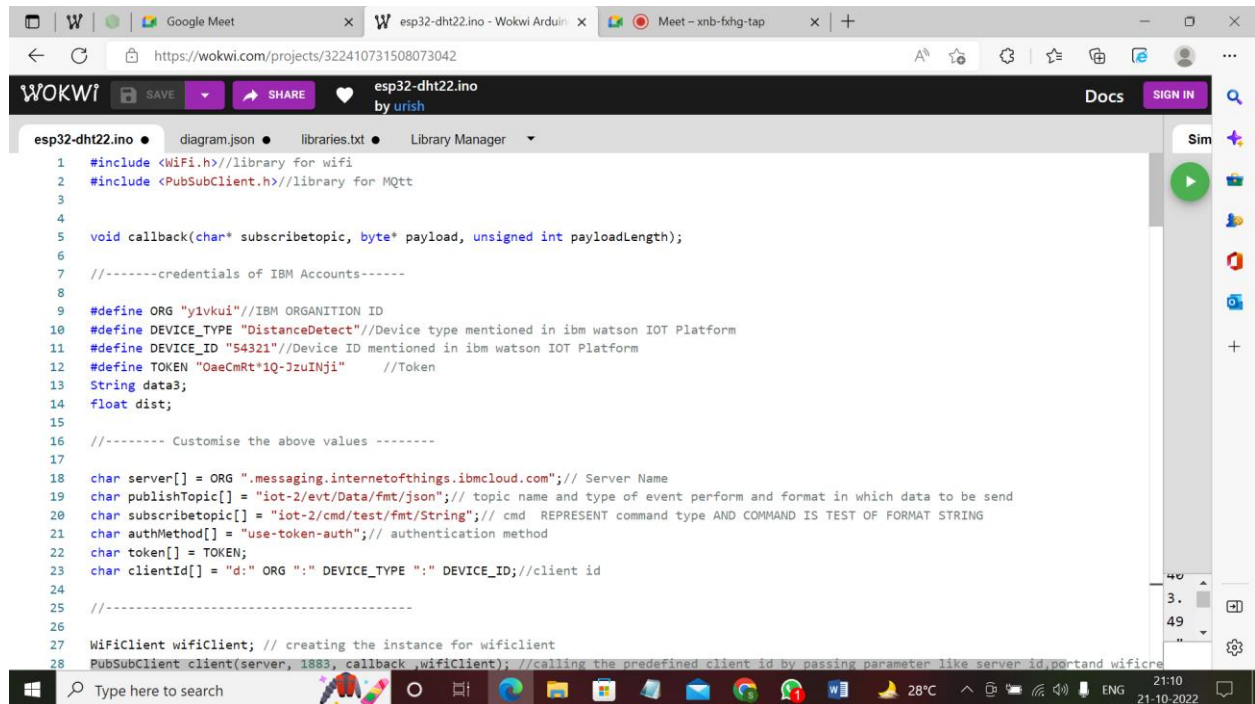
Date	26.10.2022
Team ID	PNT2022TMID27689
Name	B.Saroja
Student Roll Number	311419104069
Maximum Marks	2 Marks

DISTANCE DETECTION USING ULTRASONIC

Write code and connections in wokwi for the ultrasonic sensor.

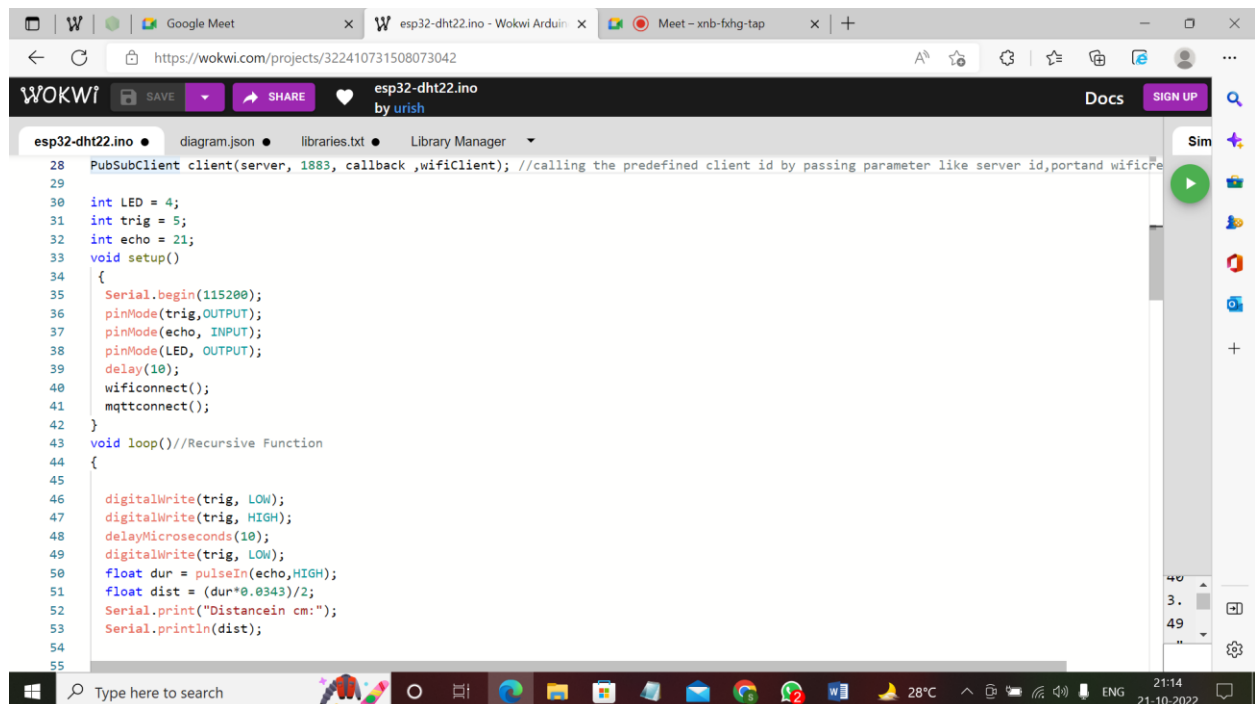
Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display in the device recent events.

CODE:



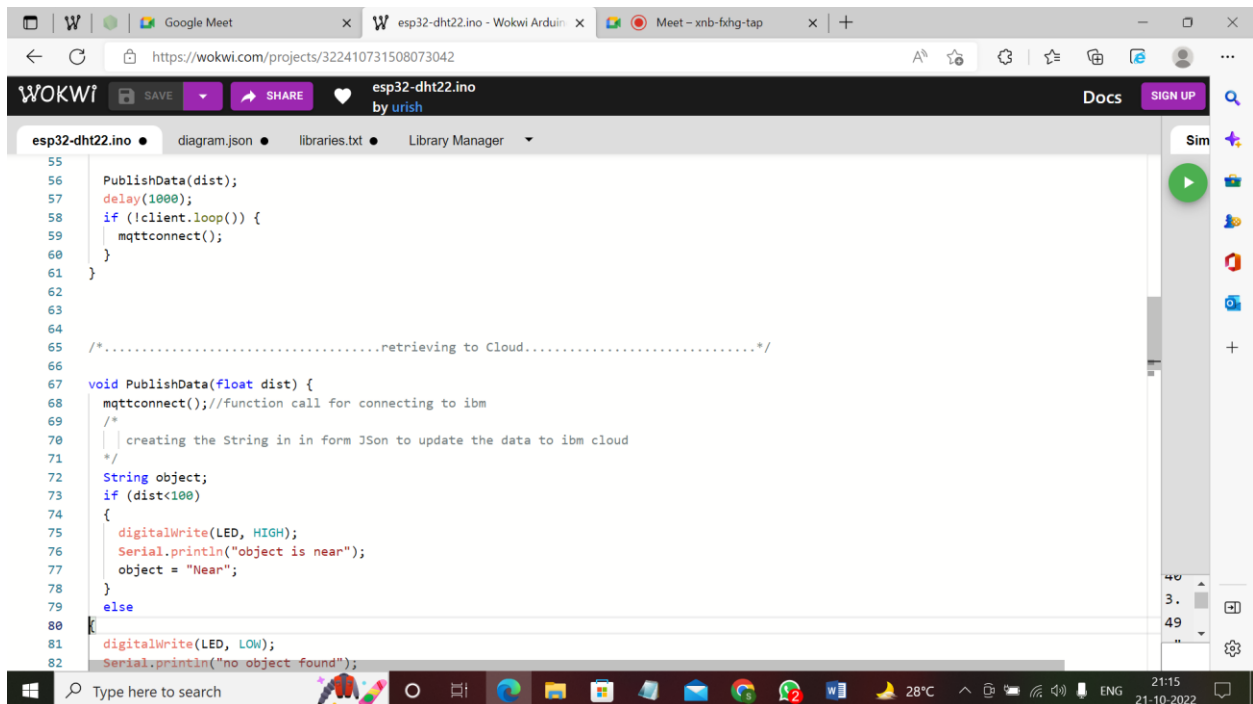
The screenshot shows the Wokwi IDE interface with the file 'esp32-dht22.ino' open. The code includes headers for WiFi and PubSubClient, defines a callback function, and sets up IBM IoT credentials. It also defines the device type and ID, and initializes the WiFi and PubSubClient objects.

```
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 "y1vkui" //IBM ORGANITION ID
10 #define DEVICE_TYPE "DistanceDetect" //Device type mentioned in ibm watson IOT Platform
11 #define DEVICE_ID "54321" //Device ID mentioned in ibm watson IOT Platform
12 #define TOKEN "OaeCmRt*IQ-3zuINji" //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/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 wificlient
```

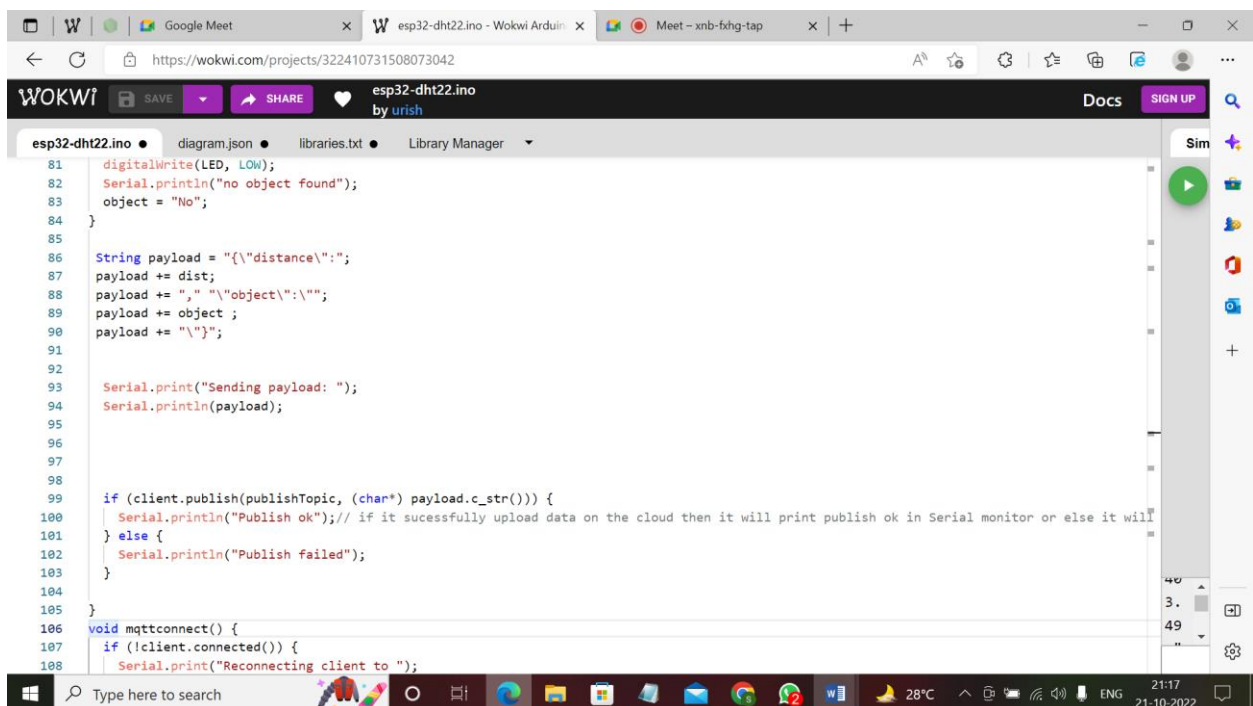


The screenshot shows the continuation of the Arduino code in the Wokwi IDE. It includes the initialization of the PubSubClient, setup of the LED and echo pins, and the main loop function that controls the LED and sends distance data via MQTT.

```
28 PubSubClient client(server, 1883, callback ,wificlient); //calling the predefined client id by passing parameter like server id,portand wificlient
29
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   wificlient.connect();
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
55 }
```



```
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  
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");
```



```
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  
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  
101   } else {  
102     Serial.println("Publish failed");  
103   }  
104  
105 }  
106  
107 void mqttconnect() {  
108   if (!client.connected()) {  
109     Serial.print("Reconnecting client to ");
```

WOKWI

SAVE SHARE esp32-dht22.ino by urish Docs SIGN UP

esp32-dht22.ino diagram.json libraries.txt Library Manager

```
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());
133 }
134
135 void initManagedDevice() {
```

21:17 21-10-2022

WOKWI

SAVE SHARE esp32-dht22.ino by urish Docs SIGN IN

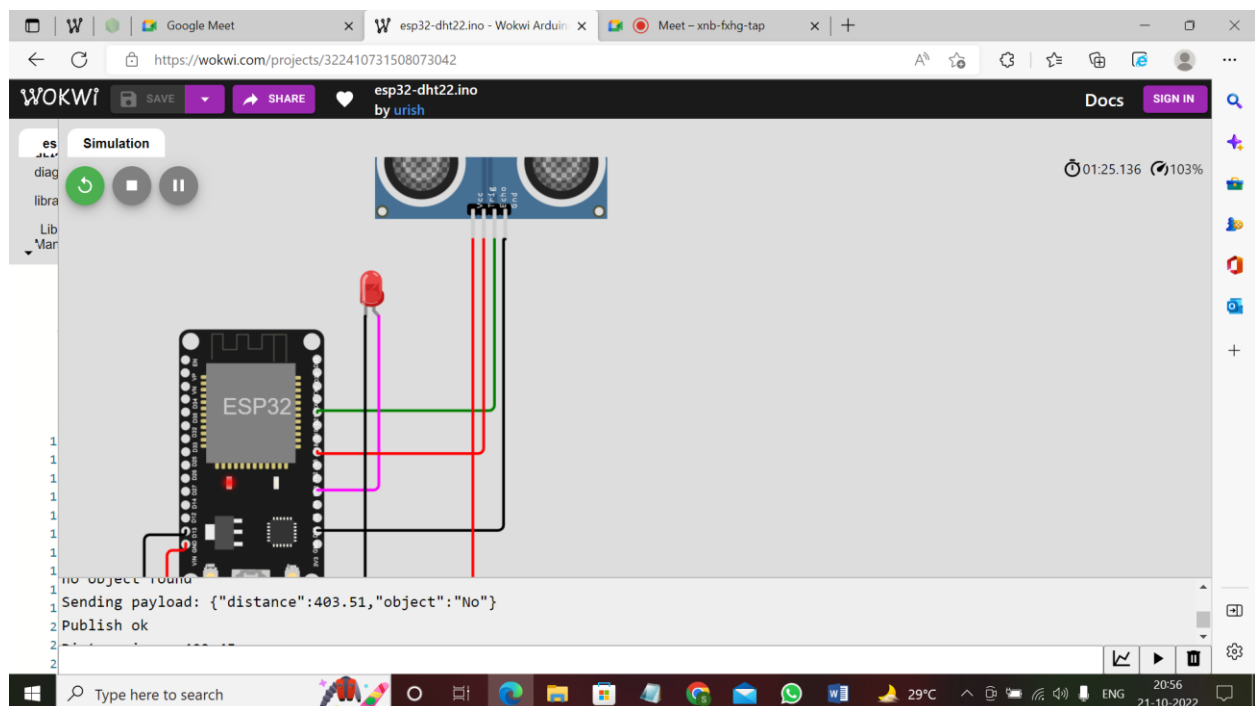
esp32-dht22.ino diagram.json libraries.txt Library Manager


```
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   Serial.print("callback invoked for topic: ");
147   Serial.println(subscribetopic);
148   for (int i = 0; i < payloadLength; i++) {
149     //Serial.print(char payload[i]);
150     data3 += (char)payload[i];
151   }
152
153   // Serial.println("data: "+ data3);
154   // if(data3=="lighton")
155   // {
156   //   Serial.println(data3);
157   //   digitalWrite(LED,HIGH);
158   // }
159
160   // }
161 }
```

21:18 21-10-2022

```
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
167     data3="";
168
169
170 }
```

OUTPUT:





The screenshot shows the IBM Watson IoT Platform interface. At the top, the breadcrumb navigation indicates the path: **IBM Watson IoT Platform** > **Dashboard** > **Devices** > **Browse**. The main header displays the device ID **311419104069@smarinternz.com** and the device name **ID: y1vkui**. Below the header, there are tabs for **Browse**, **Action**, **Device Types**, and **Interfaces**, with **Browse** being the active tab. An **Add Device** button is located on the right. The central area contains a table with the following data:

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

At the bottom of the table, it indicates **Items per page 50** and **1-2 of 2 items**. Below the table, a status bar shows **1 Simulation running**. The bottom of the image shows a Windows taskbar with various application icons and a system clock displaying **20:46 21-10-2022**.