

Assignment -4

Python Programming

Assignment Date	25 October 2022
Student Name	Aparna J
Student Roll Number	721719106008
Team ID	PNT2022TMID07524
Maximum Marks	2 Marks

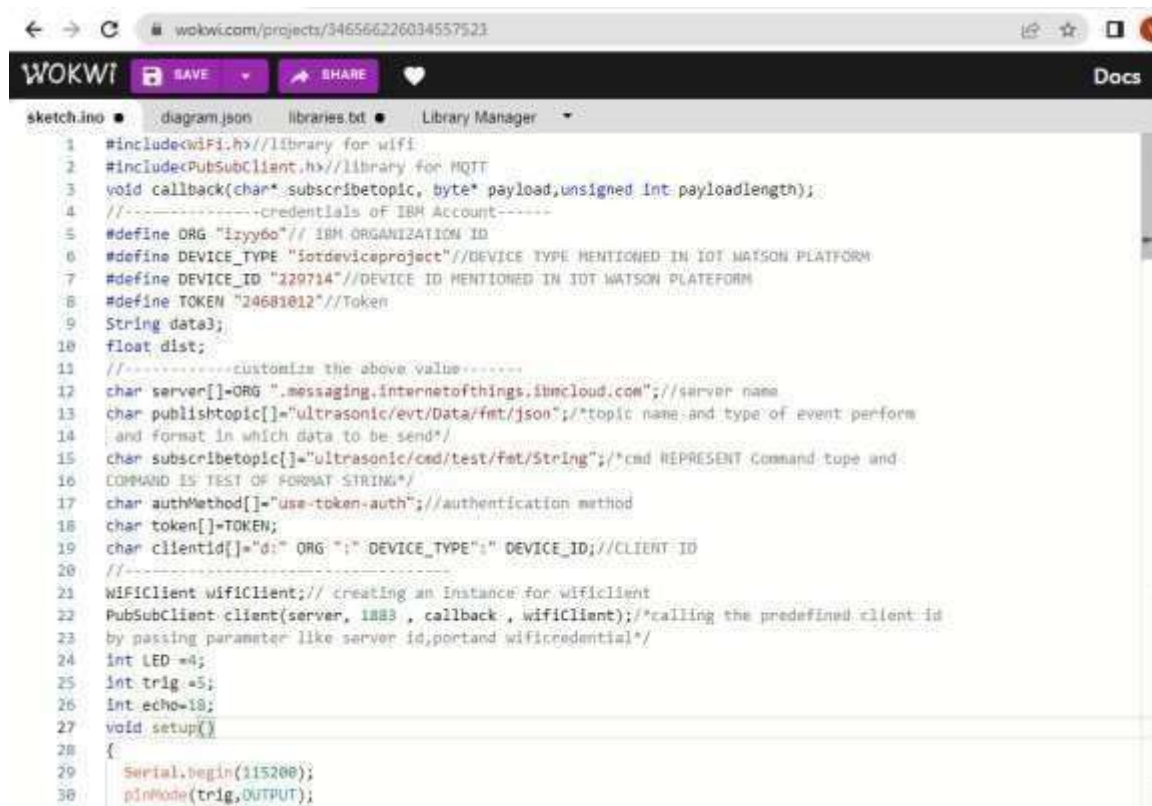
Question-1:

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.

Upload document with wokwi share link and images of ibm cloud

Solution:



```
1 #include<WiFi.h>//library for wifi
2 #include<PubSubClient.h>//library for MQTT
3 void callback(char* subscribetopic, byte* payload,unsigned int payloadlength);
4 //-----credentials of IBM Account-----
5 #define ORG "i3yy6o"// IBM ORGANIZATION ID
6 #define DEVICE_TYPE "iotdeviceproject"//DEVICE TYPE MENTIONED IN IOT WATSON PLATFORM
7 #define DEVICE_ID "229714"//DEVICE ID MENTIONED IN IOT WATSON PLATFORM
8 #define TOKEN "24681012"//Token
9 String data3;
10 float dist;
11 //-----customize the above value-----
12 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";//server name
13 char publishtopic[] = "ultrasonic/evt/Data/fmt/json";//topic name and type of event perform
14 and format in which data to be send*/
15 char subscribetopic[] = "ultrasonic/cmd/test/fmt/String";//cmd REPRESENT Command tope and
16 COMMAND IS TEST OF FORMAT STRING*/
17 char authMethod[] = "use-token-auth";//authentication method
18 char token[] = TOKEN;
19 char clientid[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//CLIENT ID
20 //-----
21 WiFiClient wificlient;// creating an Instance for wificlient
22 PubSubClient client(server, 1883, callback, wificlient);/*calling the predefined client id
23 by passing parameter like server id,portand wificredential*/
24 int LED = 4;
25 int trig = 5;
26 int echo = 18;
27 void setup()
28 {
29   Serial.begin(115200);
30   pinMode(trig,OUTPUT);
```

```

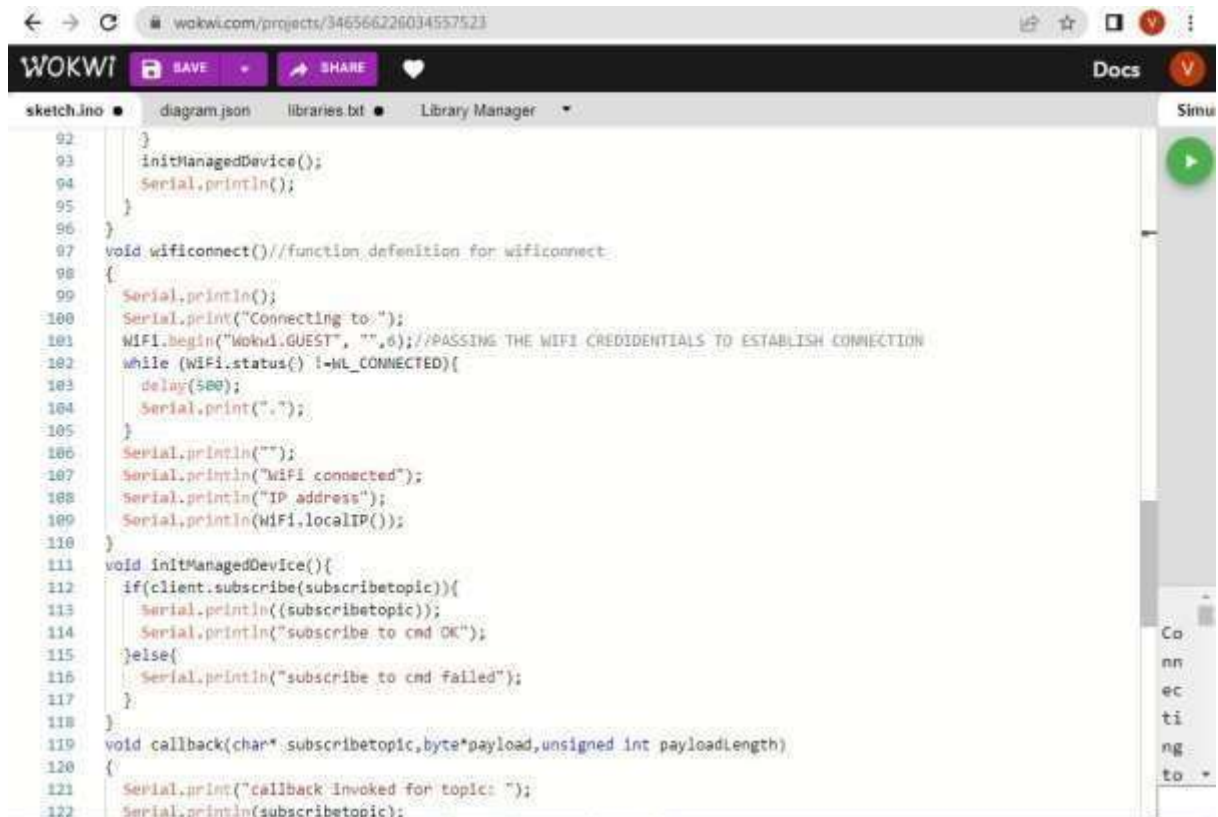
35 | mqttconnect();
36 |
37 | ..... loop()//recursive function
38 |
39 | {
40 |     digitalWrite(trig,HIGH);
41 |     delay(1000);
42 |     digitalWrite(trig,LOW);
43 |     float dur=pulseIn(echo,HIGH);
44 |     float dist=(dur * 0.0343)/2;
45 |     Serial.print("Distance: ");
46 |     Serial.println(dist);
47 |     PublishData(dist);
48 |     delay(1000);
49 |     if (!client.isConnected())
50 |         mqttconnect();
51 | }
52 |
53 | /* .....retriving to cloud..... */
54 | void PublishData(float dist){
55 |     mqttconnect();//function call for connecting to ibm
56 |     /*creating the string in form of json to update the data to ibm cloud*/
57 |     String object;
58 |     if(dist<100)
59 |     {
60 |         digitalWrite(LED,HIGH);

```

```

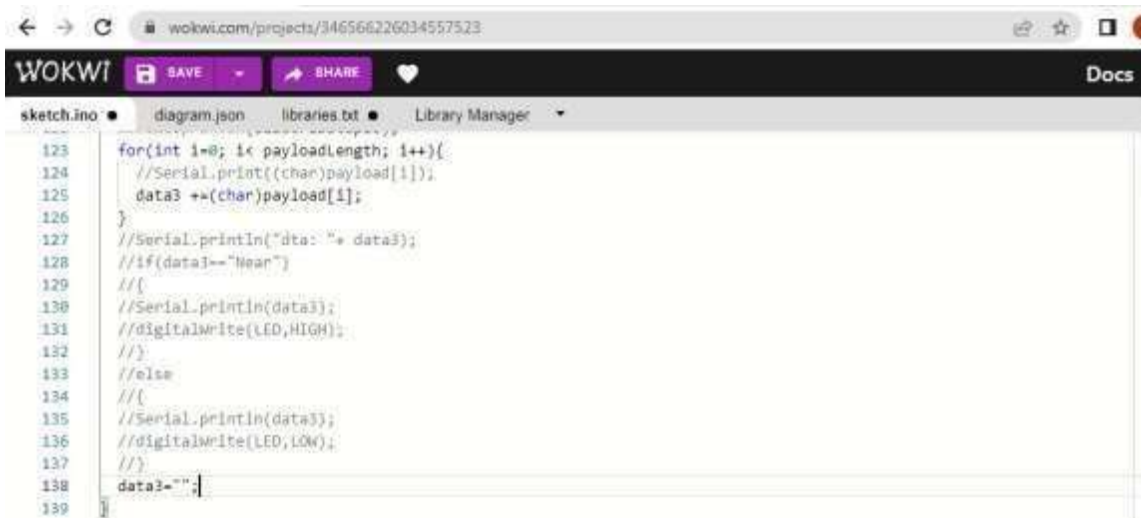
61 |     Serial.println("object=");
62 |     object="1";
63 | }
64 | else
65 | {
66 |     digitalWrite(LED,LOW);
67 |     Serial.println("No");
68 |     object="No";
69 | }
70 | String payload="{\"";
71 | payload +=dist;
72 | payload +=\",\";
73 | payload += object;
74 | payload += \"\";
75 |
76 | Serial.print("Sending payload: ");
77 | Serial.println(payload);
78 | if(client.publish(topic, (char*) payload.c_str())){
79 |     Serial.println("Publish ok");
80 |     /* if it's successful, upload data on the cloud then it will print
81 |     publish ok in serial monitor or else it will print publish failed */
82 | } else{
83 |     Serial.println("Publish failed");
84 | }
85 | }
86 |
87 | void mqttconnect(){

```



The screenshot shows the Wokwi IDE interface with a project titled 'wokwi.com/projects/346566226034557523'. The code in 'sketch.ino' includes functions for initializing a managed device, connecting to a WiFi network, and subscribing to an MQTT topic. The code is as follows:

```
92 }
93 initManagedDevice();
94 Serial.println();
95 }
96 }
97 void wificonnect()//function definition for wificonnect
98 {
99     Serial.println();
100     Serial.print("Connecting to:");
101     WiFi.begin("Wokwi.GUEST", "",6);//PASSING THE WIFI CREDENTIALS TO ESTABLISH CONNECTION
102     while (WiFi.status() !=WL_CONNECTED){
103         delay(500);
104         Serial.print(".");
105     }
106     Serial.println("");
107     Serial.println("WiFi connected");
108     Serial.println("IP address:");
109     Serial.println(WiFi.localIP());
110 }
111 void initManagedDevice(){
112     if(client.subscribe(subscribetopic)){
113         Serial.println((subscribetopic));
114         Serial.println("subscribe to cmd OK");
115     }else{
116         Serial.println("subscribe to cmd failed");
117     }
118 }
119 void callback(char* subscribetopic,byte*payload,unsigned int payloadLength)
120 {
121     Serial.print("callback invoked for topic: ");
122     Serial.println(subscribetopic);
```



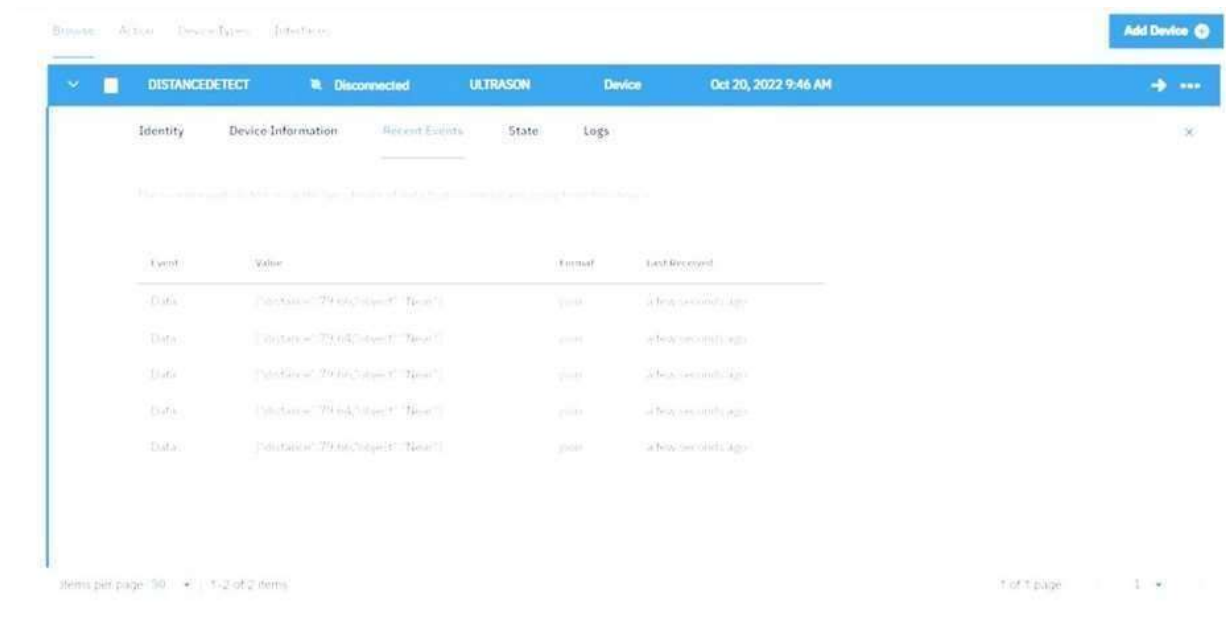
The screenshot shows the Wokwi IDE interface with the same project. The code in 'sketch.ino' continues with a loop to process the MQTT payload. The code is as follows:

```
123 for(int i=0; i< payloadLength; i++){
124     //Serial.print((char)payload[i]);
125     data3 +=(char)payload[i];
126 }
127 //Serial.println("dto: "+ data3);
128 //if(data3=="Hear")
129 //{
130 //Serial.println(data3);
131 //digitalWrite(LED,HIGH);
132 //}
133 //else
134 //{
135 //Serial.println(data3);
136 //digitalWrite(LED,LOW);
137 //}
138 data3="";
139 }
```

OUTPUT:

<https://wokwi.com/projects/346572482591851092>

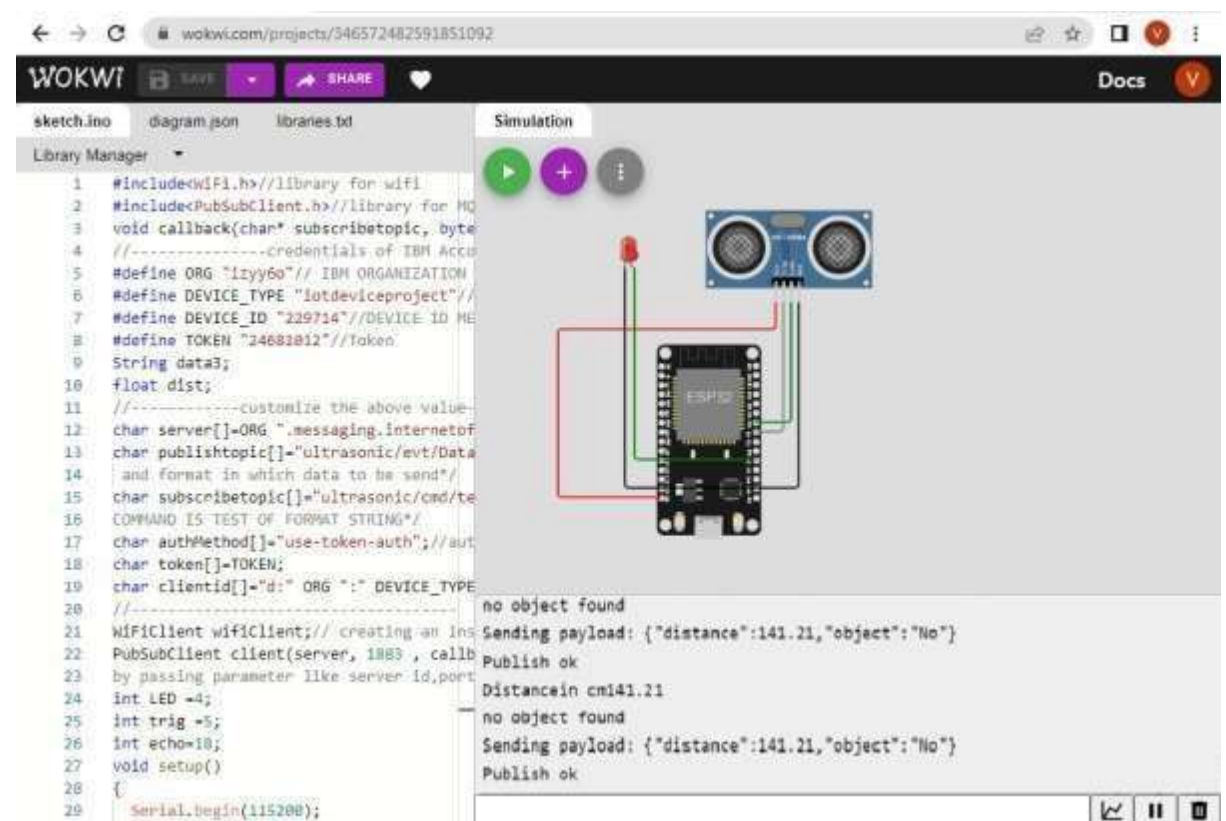
DATA SENT TO IBM CLOUD ON NO OBJECT DETECTED



The screenshot shows the IBM IoT Dashboard interface. At the top, there's a navigation bar with 'DISTANCEDETECT' selected. Below it, a table displays recent events. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. The events are all 'Data' type, with values indicating 'Distance: 79 cm (object: "No")'. The 'Format' column shows 'json', and the 'Last Received' column shows timestamps like '4 hrs, 55 min, 11 s ago'.

Event	Value	Format	Last Received
Data	Distance: 79 cm (object: "No")	json	4 hrs, 55 min, 11 s ago
Data	Distance: 79 cm (object: "No")	json	4 hrs, 55 min, 11 s ago
Data	Distance: 79 cm (object: "No")	json	4 hrs, 55 min, 11 s ago
Data	Distance: 79 cm (object: "No")	json	4 hrs, 55 min, 11 s ago
Data	Distance: 79 cm (object: "No")	json	4 hrs, 55 min, 11 s ago

WHEN NO OBJECT DETECTED BY ULTRASONIC DETECTOR



The screenshot shows the Wokwi IDE interface. On the left, the 'sketch.ino' file is open, displaying code for an ESP8266 connected to an ultrasonic sensor. The code includes comments and defines variables for the sensor's pin configuration and the MQTT server details. The 'Simulation' tab is active, showing a visual representation of the ESP8266 and the ultrasonic sensor. The console output shows the following messages:

```
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 SENT TO IBM CLOUD ON OBJECT BEING DETECTED

