

IBM ASSIGNMENT-3 IOT DOMAIN

LINK:

<https://wokwi.com/projects/364540012436747265>

CODE:

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT

const int trigPin = 12; // trig pin of the ultrasonic sensor
const int echoPin = 13; // echo pin of the ultrasonic sensor

void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);

//-----credentials of IBM Accounts-----

#define ORG "dc8x19"//IBM ORGANITION ID
#define DEVICE_TYPE "abcde"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "123"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "87654321" //Token
String data3;
long duration, distance_cm;

//----- Customise the above values -----

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of
event perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT
command type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id

//-----

WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient);
void setup() {
  Serial.begin(9600); // initialize serial communication
  pinMode(trigPin, OUTPUT); // set trig pin as output
  pinMode(echoPin, INPUT); // set echo pin as input
  wificonnect();
  mqttconnect();
}
```

```

void loop() {
    digitalWrite(trigPin, LOW); // send a low pulse to trig pin
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH); // send a high pulse to trig pin for 10
microseconds
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH); // read the duration of the pulse from
echo pin

    distance_cm = duration / 1000; // calculate distance in cm

    if(distance_cm <=100){
        Serial.print("Distance: ");
        Serial.print(distance_cm);
        Serial.println(" cm");
        PublishData(distance_cm);
        delay(1000);
        if (!client.loop()) {
            mqttconnect();
        }
    }
    else{
        Serial.println("Distance is greater than 100 ,we cannot print and sent to
cloud");
    }
    delay(1000); // wait for 1 second before taking the next measurement
}

void PublishData(int distance) {
    mqttconnect();//function call for connecting to ibm
/*
creating the String in in form JSon to update the data to ibm cloud
*/

    String payload = "{\"distance\":";
    payload += distance;
    payload += "}";

    Serial.print("Sending payload: ");
    Serial.println(payload);

    if (client.publish(publishTopic, (char*) payload.c_str())) {
        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
    } else {
        Serial.println("Publish failed");
    }
}

```

```

    }
}
void mqttconnect() {
    if (!client.connected()) {
        Serial.print("Reconnecting client to ");
        Serial.println(server);
        while (!client.connect(clientId, authMethod, token)) {
            Serial.print(".");
            delay(500);
        }
        initManagedDevice();
        Serial.println();
    }
}

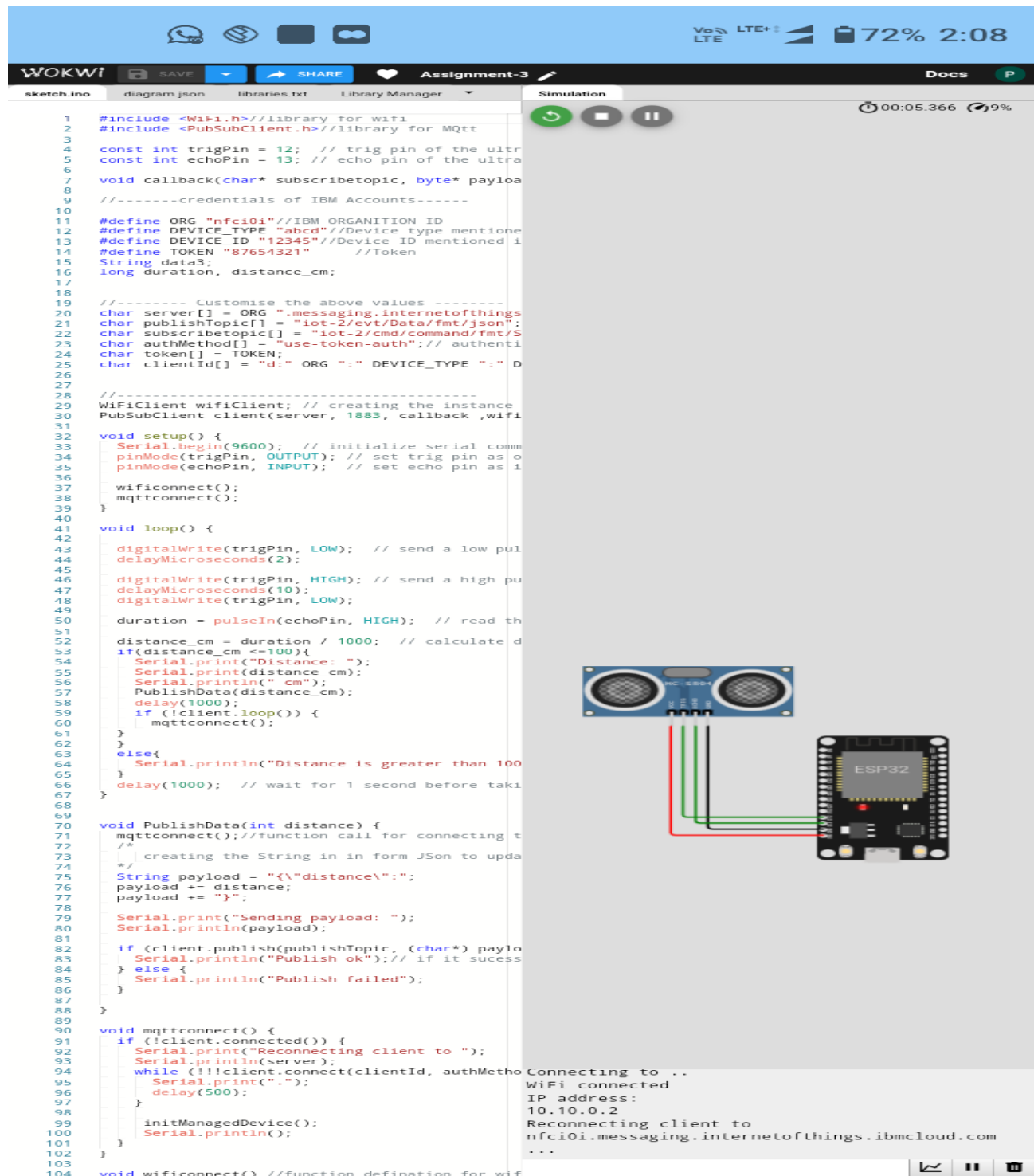
void wificonnect() //function definition for wificonnect
{
    Serial.println();
    Serial.print("Connecting to ");
    WiFi.begin("Wokwi-GUEST", "", 6); //passing the wifi credentials to establish
the connection

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}
void initManagedDevice() {
    if (client.subscribe(subscribetopic)) {
        Serial.println((subscribetopic));
        Serial.println("subscribe to cmd OK");
    } else {
        Serial.println("subscribe to cmd FAILED");
    }
}

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{
    Serial.print("callback invoked for topic: ");
    Serial.println(subscribetopic);
    for (int i = 0; i < payloadLength; i++) {
        //Serial.print((char)payload[i]);
        data3 += (char)payload[i];
    }
    Serial.println("data: "+ data3);
}

```

CONNECTION:



The screenshot displays the Wokwi IoT simulator interface. The left pane shows the Arduino sketch code, and the right pane shows the simulation results.

Code (sketch.ino):

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3
4 const int trigPin = 12; // trig pin of the ultrasonic sensor
5 const int echoPin = 13; // echo pin of the ultrasonic sensor
6
7 void callback(char* subscribetopic, byte* payload) {
8   // Do something with the message
9   //-----credentials of IBM Accounts-----
10
11   #define ORG "nfc10i" //IBM ORGANITION ID
12   #define DEVICE_TYPE "abcd" //Device type mentioned in IBM Cloud IoT
13   #define DEVICE_ID "12345" //Device ID mentioned in IBM Cloud IoT
14   #define TOKEN "87654321" //Token
15   String data3;
16   long duration, distance_cm;
17
18   //----- Customise the above values -----
19
20   char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
21   char publishTopic[] = "iot-2/evt/Data/fmt/json";
22   char subscribetopic[] = "iot-2/cmd/command/fmt/JSON";
23   char authMethod[] = "use-token-auth"; // authentication method
24   char token[] = TOKEN;
25   char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
26
27   //-----
28   WiFiClient wifiClient; // creating the instance
29   PubSubClient client(server, 1883, callback, wifiClient);
30
31   void setup() {
32     Serial.begin(9600); // initialize serial communication
33     pinMode(trigPin, OUTPUT); // set trig pin as output
34     pinMode(echoPin, INPUT); // set echo pin as input
35
36     wifiConnect();
37     mqttConnect();
38   }
39
40   void loop() {
41     digitalWrite(trigPin, LOW); // send a low pulse
42     delayMicroseconds(2);
43     digitalWrite(trigPin, HIGH); // send a high pulse
44     delayMicroseconds(10);
45     digitalWrite(trigPin, LOW);
46
47     duration = pulseIn(echoPin, HIGH); // read the duration of the pulse
48
49     distance_cm = duration / 1000; // calculate the distance in cm
50
51     if (distance_cm <= 100) {
52       Serial.print("Distance: ");
53       Serial.print(distance_cm);
54       Serial.println(" cm");
55       PublishData(distance_cm);
56       delay(1000);
57     } else {
58       Serial.println("Distance is greater than 100 cm");
59       delay(1000); // wait for 1 second before taking next measurement
60     }
61   }
62
63   void PublishData(int distance) {
64     mqttConnect(); //function call for connecting to MQTT
65     /* creating the String in form of JSON to update the cloud */
66     String payload = "{\"distance\": ";
67     payload += distance;
68     payload += " }";
69     Serial.print("Sending payload: ");
70     Serial.println(payload);
71
72     if (client.publish(publishTopic, (char*) payload)) {
73       Serial.println("Publish ok"); // if it success
74     } else {
75       Serial.println("Publish failed");
76     }
77   }
78
79   void mqttConnect() {
80     if (!client.connected()) {
81       Serial.print("Reconnecting client to ");
82       Serial.println(server);
83       while (!client.connect(clientId, authMethod, token)) {
84         Serial.print(".");
85         delay(500);
86       }
87       initManagedDevice();
88       Serial.println();
89     }
90   }
91
92   void wifiConnect() //function definition for wifi connection
```

Simulation Results:

The simulation window shows a visual representation of the hardware (ESP32 and ultrasonic sensor) and a log of connection events:

```
Connecting to ..
WiFi connected
IP address: 10.10.0.2
Reconnecting client to nfc10i.messaging.internetofthings.ibmcloud.com
...
```

IBM CLOUD RECENTS EVENTS SCREENSHOT:

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A user profile is visible in the top right corner with the email 'bpavithra962@gmail.com' and ID 'nfc0i'. The main content area shows a device named '12345' with status 'Connected' and name 'abcd'. Below this, the 'Recent Events' tab is selected, displaying a table of events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. The events listed are all 'Data' events with a value of '{\"distance\":23}' in 'json' format, received at various times. A status message at the bottom indicates '1 Simulation running'.

Event	Value	Format	Last Received
Data	{\"distance\":23}	json	a few seconds ago
Data	{\"distance\":23}	json	a few seconds ago
Data	{\"distance\":23}	json	a few seconds ago
Data	{\"distance\":23}	json	a few seconds ago
Data	{\"distance\":23}	json	a minute ago

1 Simulation running