

**NAME:ABHINANDU THILAK**

**DEPARTMENT : MEDICAL ELECTRONICS**

**YEAR:3<sup>RD</sup>YEAR**

**COLLEGE:SENGUNTHAR COLLEGE OF ENGINEERING**

**GROUP:GROUP-3**

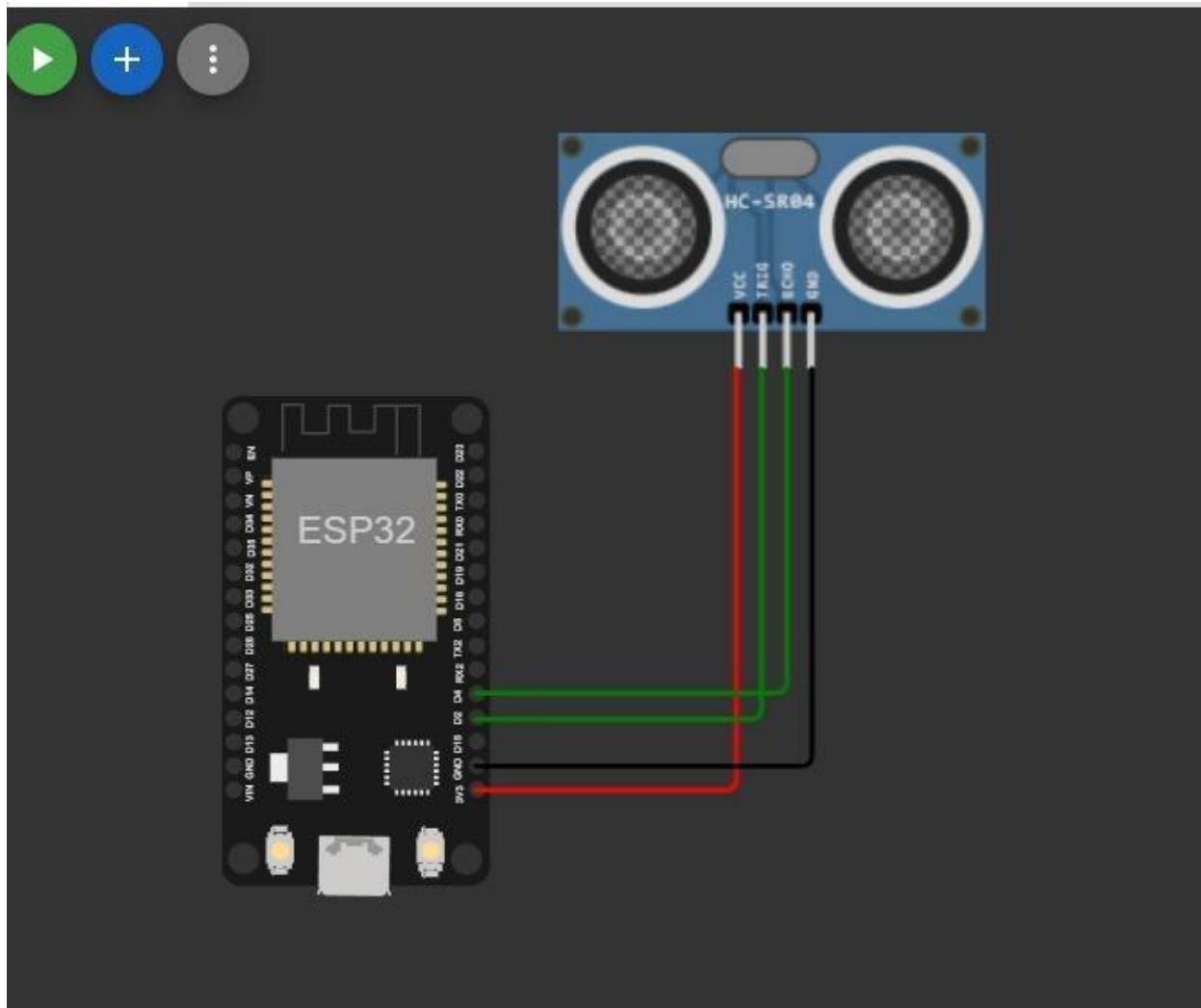
**NM TEAM ID:NM2023TMID13573**

**PROFESSIONAL READINESS FOR  
INNOVATION,EMPLOYABILITY AND  
ENTREPRENEURSHIP**

**INTERNET OF THINGS**

**ASSIGNMENT-3**

### Wokwi simulation:



### Code:

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT
#include "Ultrasonic.h"
Ultrasonic ultrasonic(2, 4);
float distance;

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

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

#define ORG "sptrws" //IBM ORGANITION ID
#define DEVICE_TYPE "abcd" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678" //Token
String data3;
//float h, t;
```

```
//----- 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/test/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); //calling the predefined client id by passing
parameter like server id,portand wificredential
void setup()// configureing the ESP32
{
    Serial.begin(115200);

    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

void loop()// Recursive Function
{

    distance = ultrasonic.read(CM);

    Serial.print("Distance in CM: ");
    Serial.println(distance);
    delay(1000);

    PublishData(distance);
    delay(1000);
    if (!client.loop()) {
        mqttconnect();
    }
}
}
```

/.....retrieving to Cloud ...../

```
void PublishData(float 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 defination 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);

    data3="";

}

```

### Output in wokwi:

The screenshot displays the Wokwi IDE interface. On the left, the 'sketch.ino' file contains the following code:

```

1 #include <Wifi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #include "Ultrasonic.h"
4 Ultrasonic ultrasonic(2, 4);
5 float distance;
6
7
8 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
9
10 //-----credentials of IBM Accounts-----
11
12 #define ORG "sptrws" //IBM ORGANIZATION ID
13 #define DEVICE_TYPE "abcd" //Device type mentioned in ibm watson IOT Platform
14 #define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
15 #define TOKEN "12345678" //Token
16 String data3;
17 //float h, t;
18
19
20 //----- Customise the above values -----
21 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
22 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event
23 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command
24 char authMethod[] = "use-token-auth"; // authentication method
25 char token[] = TOKEN;
26 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
27
28

```

On the right, the 'Simulation' window shows a visual representation of the hardware: an ESP32 microcontroller board connected to an HC-SR04 ultrasonic sensor via four colored jumper wires (red, green, blue, and black). The simulation output at the bottom shows the following sequence of events:

```

Distance in CM: 357.00
Sending payload: {"distance":357.00}
Publish ok
Distance in CM: 357.00
Sending payload: {"distance":357.00}
Publish ok

```

## Output in ibm cloud:

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes the platform name, a user profile icon, and the email address 'mohanrajs.md20@scew.org' with the ID 'sptrws'. Below the navigation bar, there are tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A blue 'Add Device +' button is located on the right. The main content area shows a message: 'The recent events listed show the live stream of data that is coming and going from this device.' Below this message is a table with the following data:

Event	Value	Format	Last Received
Data	{"distance":357}	json	a few seconds ago
Data	{"distance":357}	json	a few seconds ago
Data	{"distance":357}	json	a few seconds ago
Data	{"distance":357}	json	a few seconds ago
Data	{"distance":357}	json	a few seconds ago