

## Assignment -4

### ESP32 Programming

Assignment Date	1 November 2022
Student Name	JAYA SREE C T
Student Roll Number	962819106018
Maximum Marks	2 Marks

#### Assignment 4:

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.

Upload document with wokwi share link and images of IBM cloud

**My Completed Assignment Wokwi share link:**

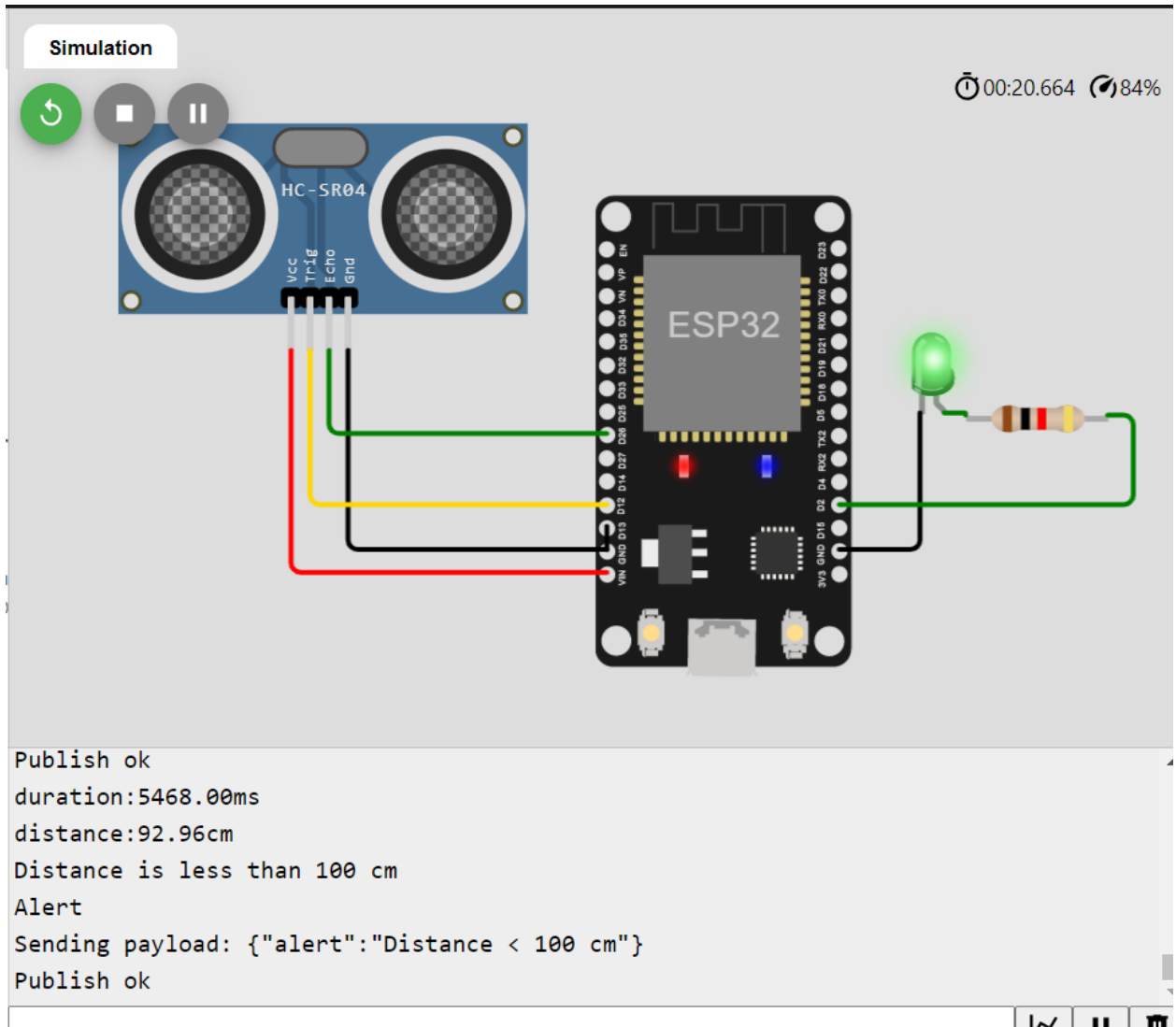
<https://wokwi.com/projects/346605259691393619>

#### Circuit Diagram:

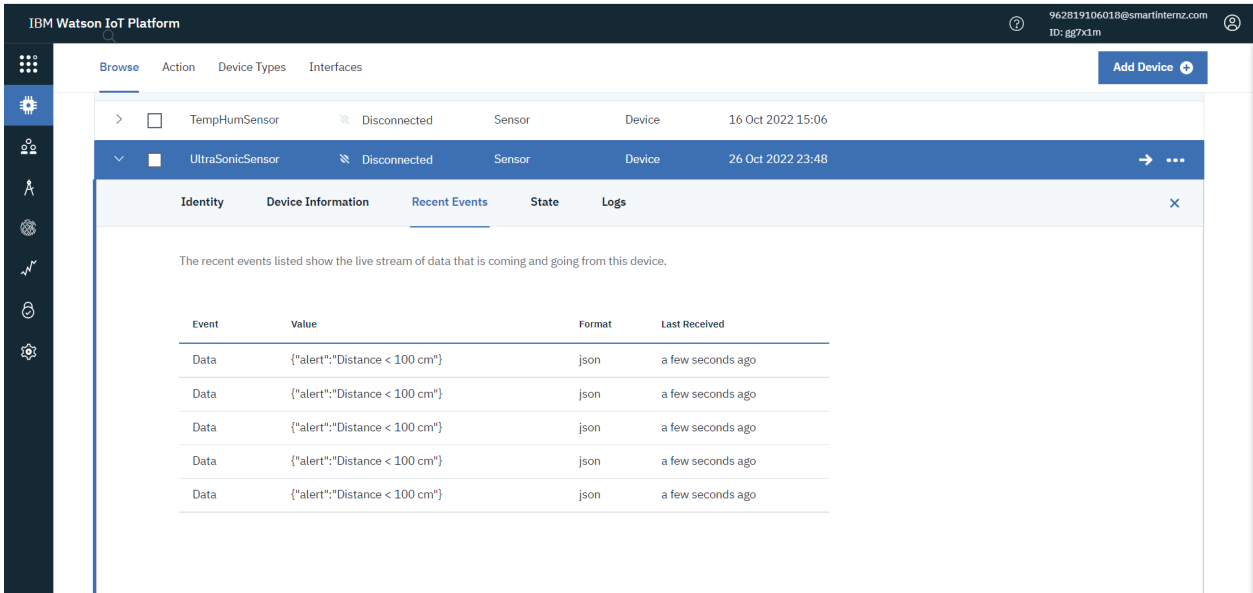
The screenshot displays the Wokwi IDE interface. On the left, the 'sketch.ino' file contains the following code:

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #define LED 2
4 #define echopin 26
5 #define trigpin 12
6
7 //-----credentials of IBM Accounts-----
8
9 #define ORG "gg7x1m" //IBM ORGANITION ID
10 #define DEVICE_TYPE "Sensor" //Device type mentioned in ibm watson IOT Platform
11 #define DEVICE_ID "ultrasonicsensor" //Device ID mentioned in ibm watson IOT Platform
12 #define TOKEN "lQtc21g)ORGhWqlzj" //Token
13 String data;
14 String data3;
15
16 //----- Customise the above values -----
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 a
19 char subscribTopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type AND CO
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 void callback(char* subscribTopic, byte* payload, unsigned int payloadLength);
25 //-----
26 WiFiClient wifiClient; // creating the instance for wificlient
27 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client
28
29 void setup() // configuring the ESP32
30 {
31   Serial.begin(115200);
32   pinMode(LED, OUTPUT);
33   pinMode(echopin, INPUT);
34   pinMode(trigpin, OUTPUT);
35 }
```

On the right, the 'Simulation' window shows a visual representation of the circuit. An ESP32 microcontroller is connected to an HC-SR04 ultrasonic sensor. The sensor's VCC pin is connected to the ESP32's 5V pin, and its GND pin is connected to the ESP32's GND pin. The sensor's Trig pin is connected to the ESP32's pin 12, and its Echo pin is connected to the ESP32's pin 26. A green LED is connected to the ESP32's pin 2 through a resistor, with its other end connected to GND. The simulation status bar at the bottom indicates: 'Publish ok', 'duration:5469.00ms', 'distance:92.97cm', 'Distance is less than 100 cm', 'Alert', 'Sending payload: {"alert":"Distance < 100 cm"}', and 'Publish ok'.



Images of IBM cloud:



	UltraSonicSensor		Disconnected	Sensor	Device	26 Oct 2022 23:48
Identity	Device Information	Recent Events	State	Logs		

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
Data	{"alert": "Distance < 100 cm"}	json	a few seconds ago
Data	{"alert": "Distance < 100 cm"}	json	a few seconds ago
Data	{"alert": "Distance < 100 cm"}	json	a few seconds ago
Data	{"alert": "Distance < 100 cm"}	json	a few seconds ago
Data	{"alert": "Distance < 100 cm"}	json	a few seconds ago

## Code:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQTT
#define LED 2
#define echopin 26
#define trigpin 12

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

#define ORG "gg7x1m">//IBM ORGANIZATION ID
#define DEVICE_TYPE "Sensor">//Device type mentioned in ibm
watson IOT Platform
#define DEVICE_ID "UltraSonicSensor">//Device ID mentioned in
ibm watson IOT Platform
#define TOKEN "lQTc21g)ORgLhwqlZj" //Token
String data;
String data3;

//----- Customize 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

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

//-----
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);  
    pinMode(LED,OUTPUT);  
    pinMode(echopin, INPUT);  
    pinMode(trigpin,OUTPUT);  
    delay(10);  
    Serial.println();  
    wificonnect();  
    mqttconnect();  
}
```

```
void loop()// Recursive Function
```

```
{  
    digitalWrite(trigpin,LOW);  
    delayMicroseconds(10);  
    digitalWrite(trigpin,HIGH);  
    delayMicroseconds(10);  
    digitalWrite(trigpin,LOW);  
    double duration=pulseIn(echopin,HIGH);  
    Serial.print("duration:");  
    Serial.print(duration);  
    Serial.println("ms");  
    double distance=(duration*0.034)/2;  
    Serial.print("distance:");  
    Serial.print(distance);  
    Serial.println("cm");  
    if(distance<100)  
    {  
        digitalWrite(LED,HIGH);  
    }  
    else  
    {  
        digitalWrite(LED,LOW);  
    }  
}
```

```

}

if (distance<100)
{
    Serial.println("Distance is less than 100 cm");
    Serial.println("Alert");
    data="Distance < 100 cm";
    PublishData(data);
    delay(1000);
    if (!client.loop()) {
        mqttconnect();
    }
}
}

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

void PublishData(String data) {
    mqttconnect();//function call for connecting to ibm
    /*
        creating the String in in form JSon to update the data to
    ibm cloud
    */
    String payload = "{\"alert\":\"";
    payload += "\"";
    payload += data;
    payload += "\"";
    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);
        }
    }
}
```

```
    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 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);
}
```