

ASSIGNMENT 4
WOKWI PROGRAM

Assignment Date	27 OCT 2022
Student Name	SANJAY B
Student Roll Number	727819TUCS212

PROGRAM

Smart Waste Management System for Metropolitan Cities

ASSIGNMENT 4:

Write code and connections in wokwi for ultrasonic sensors. Whenever distance is less than 100 cms send "alert" to ibm cloud and display in device recent events. Uplode document with wokwi share link and images of ibm cloud.

CODE:

```
#include <WiFi.h>

#include <PubSubClient.h>

WiFiClient wifiClient;

String data3;

#define ORG "ztcz45"

#define DEVICE_TYPE "naveen"

#define DEVICE_ID "naveen123"

#define TOKEN "123456789"

#define speed 0.034

#define led 14

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char publishTopic[] = "iot-2/evt/Data/fmt/json";

char topic[] = "iot-2/cmd/home/fmt/String";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

PubSubClient client(server, 1883, wifiClient);
```

```
void publishData();
```

```
const int trigpin=5;
```

```
const int echopin=18;
```

```
String command;
```

```
String data="";
```

```
long duration;
```

```
float dist;
```

```
void setup()
```

```
{
```

```
    Serial.begin(115200);
```

```
    pinMode(led, OUTPUT);
```

```
    pinMode(trigpin, OUTPUT);
```

```
    ...
```

```
[10:32 pm, 23/10/2022] Gogul B.E CSE: }
```

```
void mqttConnect() {
```

```
    if (!client.connected()) {
```

```
        Serial.print("Reconnecting MQTT client to "); Serial.println(server);
```

```
        while (!client.connect(clientId, authMethod, token)) {
```

```
            Serial.print(".");
```

```
            delay(500);
```

```
        }
```

```
        initManagedDevice();
```

```
        Serial.println();
```

```

    }
}

void initManagedDevice() {
    if (client.subscribe(topic)) {
        // Serial.println(client.subscribe(topic));
        Serial.println("IBM subscribe to cmd OK");
    } else {
        Serial.println("subscribe to cmd FAILED");
    }
}

void publishData()
{
    digitalWrite(trigpin,LOW);
    digitalWrite(trigpin,HIGH);
    delayMicroseconds(10);
    digitalWrite(trigpin,LOW);
    duration=pulseIn(echopin,HIGH);
    dist=duration*speed/2;
    if(dist<100){
        String payload = "{ \"Normal Distance\": ";
        payload += dist;
        payload += " }";

        Serial.print("\n");
        Serial.print("Sending payload: ");
        Serial.println(payload);
        if (client.publish(publishTopic, (char*) payload.c_str())) {
            Serial.println("Publish OK");
        }
    }
}

```

```

}

if(dist>101 && dist<111){
String payload = "{ \"Alert distance\": ";
payload += dist;
payload += "}";

Serial.print("\n");
Serial.print("Sending payload: ");
Serial.println(payload);
if(client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Warning crosses 110cm -- it automaticaly of the loop");
    digitalWrite(led,HIGH);
}else {
    Serial.println("Publish 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++){
dist += (char)payload[i];
}

Serial.println("data:" + data3);
if(data3=="lighton"){
Serial.println(data3);
}
}

```

```

digitalWrite(led,HIGH);

}

data3="";

}

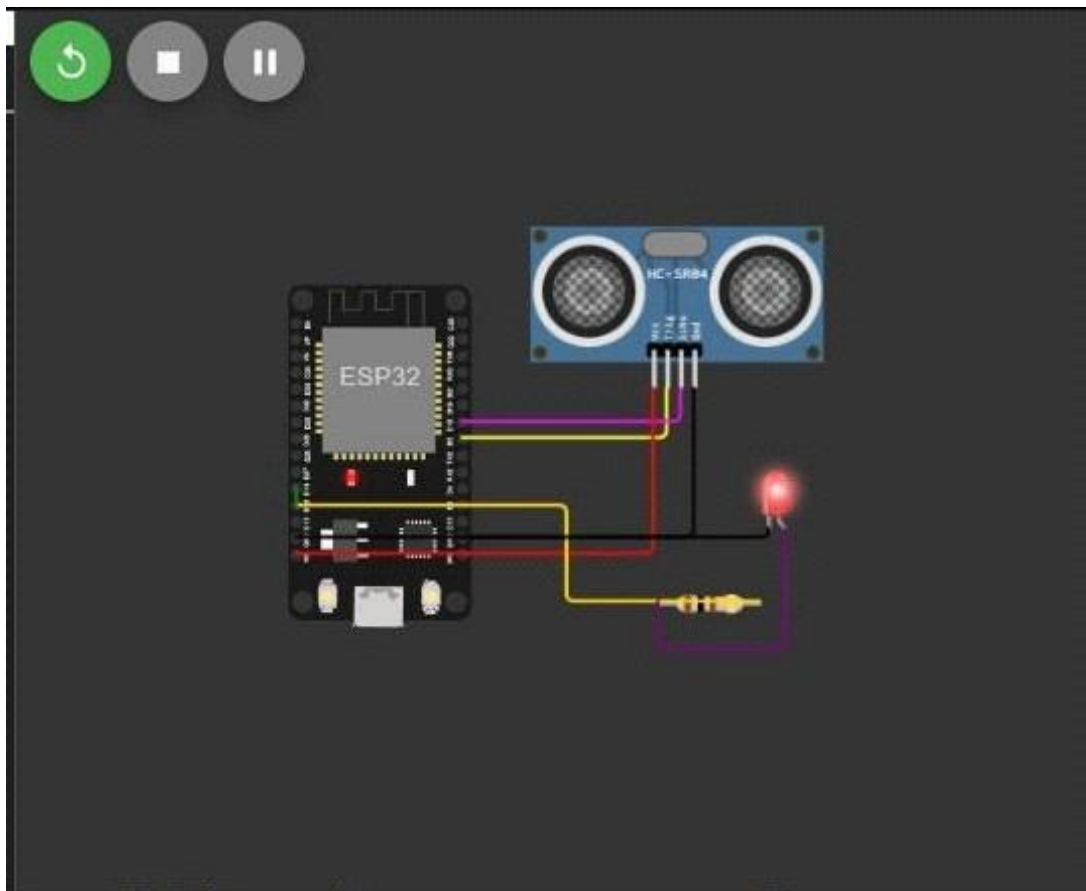
```

out put:

The screenshot displays two side-by-side windows. The left window is the Wokwi IDE, showing a C++ sketch for an ESP32. The sketch includes libraries for WiFi and MQTT, defines device credentials, and implements a loop that checks the distance from an ultrasonic sensor. If the distance is less than 100 cm, it turns an LED on and sends a JSON alert to the IBM Watson IoT Platform. The right window is the IBM Watson IoT Platform interface, showing the 'Recent Events' tab for a device named 'naveen123'. The table below lists the events sent from the device.

Event	Value	Format	Last threshold
Data	{"Alert distance":110.94}	json	a few second
Data	{"Alert distance":110.96}	json	a few second
Data	{"Alert distance":110.98}	json	a few second
Data	{"Alert distance":110.98}	json	a few second
Data	{"Alert distance":110.98}	json	a few second

1. When distance under 100 cm it wil show normal distance.



Publish OK

Sending payload:{"Normal Distance":89.95}

Publish OK

Sending payload:{"Normal Distance":89.95}

Publish OK

Sending payload:{"Normal Distance":89.95}

Publish OK

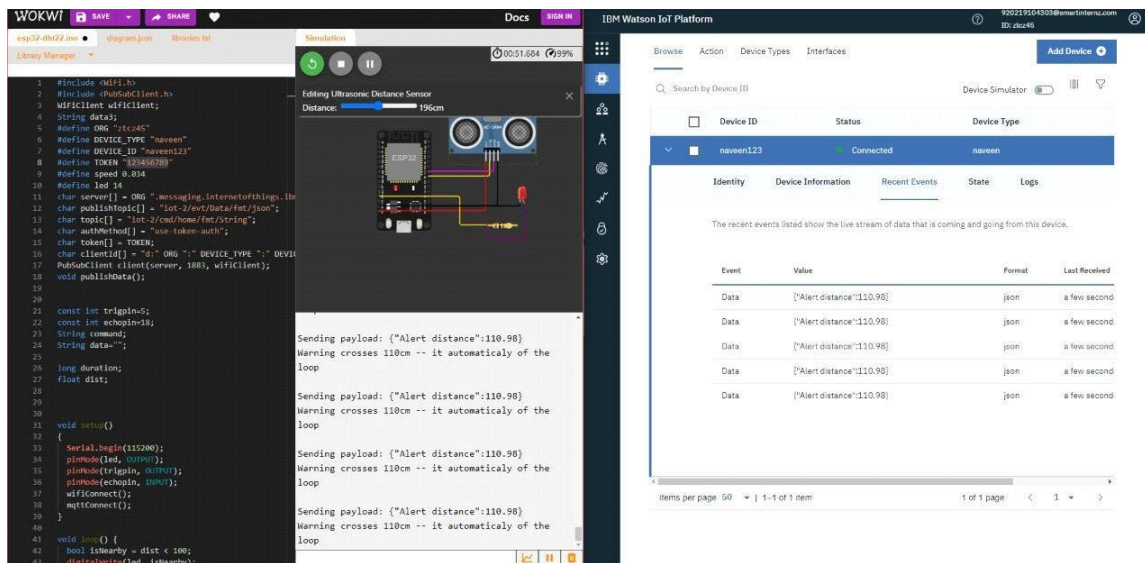
Sending payload:{"Normal Distance":89.95}

Publish OK

Sending payload:{"Normal Distance":89.95}

Publish OK

2. When distance cross 100 cm it wil show ALERT warning message distance



The screenshot displays two side-by-side windows. The left window is the Wokwi IDE, showing a C++ sketch for an ESP32 connected to an Ultrasonic Distance Sensor. The code includes headers for the PubSubClient and WiFi libraries, defines the device ID as 'naveen123', and sets up a loop that sends distance data to the IoT platform. A simulation window shows the sensor's output as 196cm. The right window is the IBM Watson IoT Platform interface, showing the device 'naveen123' in a 'Connected' state. The 'Recent Events' tab displays a log of distance readings, with the last event being 'Alert distance:110.98'.

Event	Value	Format	Last Received
Data	("Alert distance":110.98)	json	a few second
Data	("Alert distance":110.98)	json	a few second
Data	("Alert distance":110.98)	json	a few second
Data	("Alert distance":110.98)	json	a few second
Data	("Alert distance":110.98)	json	a few second

3. When it cross above 110 cm it today move to iff state once it reduce to 110 it on again

Connection information:

Basic connnection information about this device.

Organization ID : ztcz45

Device Type : naveen

Device ID : naveen123

Authentication Method : use-token-auth

Authentication Token : 123456789

▼

■

naveen123

Connected

naveen

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	{"Normal Distance":89.95}	json	a few second:
Data	{"Normal Distance":89.95}	json	a few second:
Data	{"Normal Distance":89.95}	json	a few second:
Data	{"Normal Distance":89.95}	json	a few second:
Data	{"Normal Distance":89.95}	json	a few second: