

NAME : E.KALAIVANI

REG NO : 513219104702

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="";
}
```

output:

The image shows a Wokwi IDE window on the left and a TTP IoT Platform web interface on the right.

Wokwi IDE: The code editor shows a C++ program for an Arduino Uno. The program initializes an ESP8266 module, connects to a Wi-Fi network, and sends data to a cloud server. The output window shows the following log:

```

Sending payload: ["Alert distance":118.96]
Warning crosses 118m -- It automatically of the
loop

Sending payload: ["Alert distance":118.96]
Warning crosses 118m -- It automatically of the
loop

Sending payload: ["Alert distance":118.96]
Warning crosses 118m -- It automatically of the
loop

Sending payload: ["Alert distance":118.96]
Warning crosses 118m -- It automatically of the
loop

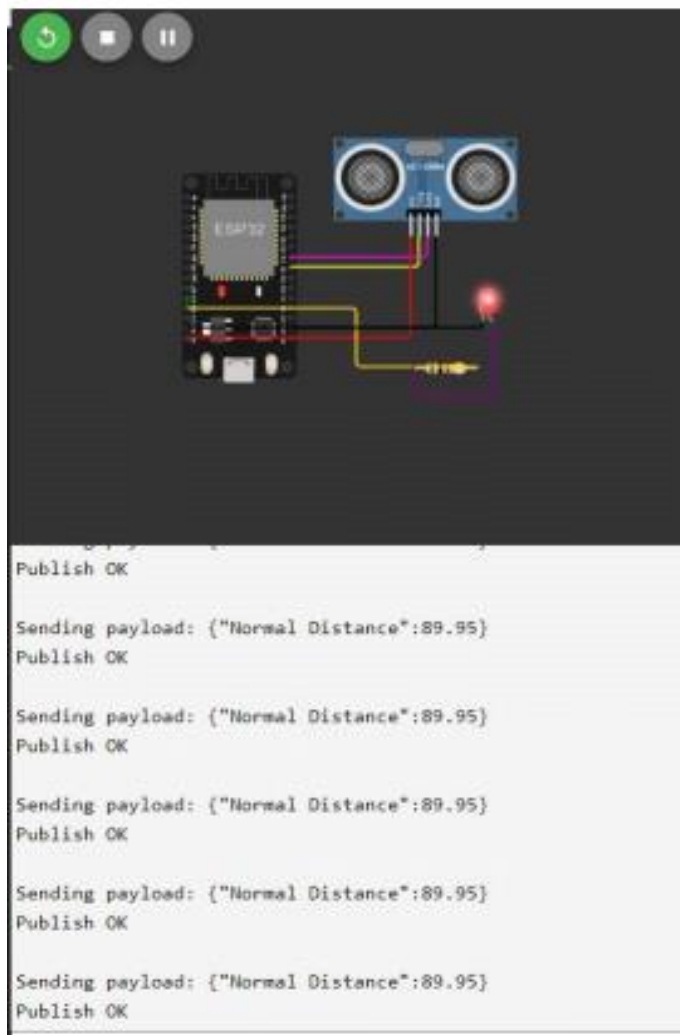
```

TTP IoT Platform: The web interface shows a list of devices. The table below displays the data for the device 'esp8266-01'.

Device ID	Status	Device Type
esp8266-01	Connected	esp8266

The table also includes columns for Identity, Device Information, Recent Events, State, and Logs. The Recent Events section shows a stream of data being sent from the device.

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



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

The screenshot shows the Wokwi IDE on the left and the IBM Watson IoT Platform on the right. The Wokwi IDE displays a C++ code snippet for an Arduino Uno connected to an Ultrasonic Distance Sensor. The code includes comments and logic for sending distance data to the cloud via MQTT. The simulation window shows the sensor's output, with a distance of 110cm. The IBM Watson IoT Platform shows the device 'akash123' in a 'Connected' state. The 'Recent Events' tab displays a stream of data points, including an alert message when the distance crosses 110cm.

```

1 #include <Arduino.h>
2 #include <PubSubClient.h>
3 #include <Ultrasonic.h>
4 #include <WiFi.h>
5 #include <MQTT.h>
6 #include <String.h>
7 #include <JSON.h>
8 #include <ArduinoJson.h>
9 #include <Wire.h>
10 #include <EEPROM.h>
11 #include <EEPROM.h>
12 #include <EEPROM.h>
13 #include <EEPROM.h>
14 #include <EEPROM.h>
15 #include <EEPROM.h>
16 #include <EEPROM.h>
17 #include <EEPROM.h>
18 #include <EEPROM.h>
19 #include <EEPROM.h>
20 #include <EEPROM.h>
21 #include <EEPROM.h>
22 #include <EEPROM.h>
23 #include <EEPROM.h>
24 #include <EEPROM.h>
25 #include <EEPROM.h>
26 #include <EEPROM.h>
27 #include <EEPROM.h>
28 #include <EEPROM.h>
29 #include <EEPROM.h>
30 #include <EEPROM.h>
31 #include <EEPROM.h>
32 #include <EEPROM.h>
33 #include <EEPROM.h>
34 #include <EEPROM.h>
35 #include <EEPROM.h>
36 #include <EEPROM.h>
37 #include <EEPROM.h>
38 #include <EEPROM.h>
39 #include <EEPROM.h>
40 #include <EEPROM.h>
41 #include <EEPROM.h>
42 #include <EEPROM.h>
43 #include <EEPROM.h>
44 #include <EEPROM.h>
45 #include <EEPROM.h>
46 #include <EEPROM.h>
47 #include <EEPROM.h>
48 #include <EEPROM.h>
49 #include <EEPROM.h>
50 #include <EEPROM.h>
51 #include <EEPROM.h>
52 #include <EEPROM.h>
53 #include <EEPROM.h>
54 #include <EEPROM.h>
55 #include <EEPROM.h>
56 #include <EEPROM.h>
57 #include <EEPROM.h>
58 #include <EEPROM.h>
59 #include <EEPROM.h>
60 #include <EEPROM.h>
61 #include <EEPROM.h>
62 #include <EEPROM.h>
63 #include <EEPROM.h>
64 #include <EEPROM.h>
65 #include <EEPROM.h>
66 #include <EEPROM.h>
67 #include <EEPROM.h>
68 #include <EEPROM.h>
69 #include <EEPROM.h>
70 #include <EEPROM.h>
71 #include <EEPROM.h>
72 #include <EEPROM.h>
73 #include <EEPROM.h>
74 #include <EEPROM.h>
75 #include <EEPROM.h>
76 #include <EEPROM.h>
77 #include <EEPROM.h>
78 #include <EEPROM.h>
79 #include <EEPROM.h>
80 #include <EEPROM.h>
81 #include <EEPROM.h>
82 #include <EEPROM.h>
83 #include <EEPROM.h>
84 #include <EEPROM.h>
85 #include <EEPROM.h>
86 #include <EEPROM.h>
87 #include <EEPROM.h>
88 #include <EEPROM.h>
89 #include <EEPROM.h>
90 #include <EEPROM.h>
91 #include <EEPROM.h>
92 #include <EEPROM.h>
93 #include <EEPROM.h>
94 #include <EEPROM.h>
95 #include <EEPROM.h>
96 #include <EEPROM.h>
97 #include <EEPROM.h>
98 #include <EEPROM.h>
99 #include <EEPROM.h>
100 #include <EEPROM.h>

```

The IBM Watson IoT Platform interface shows the device 'akash123' in a 'Connected' state. The 'Recent Events' tab displays a stream of data points, including an alert message when the distance crosses 110cm.

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 connection information about this device.

Organization ID : ztcz45

Device Type : akash

Device ID : akash123

Authentication Method : use-token-auth Authentication Token : 123456789

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	