

Team ID	PNT2022TMID05358
Project Name	Smart Waste Management System For Metropolitan Cities

Write code and connections in wokwi for ultrasonic sensor. Whenever the distance is less than 100 cm, send "alert" to IBM cloud and display in device recent events.

**WOKWI LINK:**

**<https://wokwi.com/projects/347388669920805460>**

**CODE:**

```
#include <WiFi.h>
#include <PubSubClient.h>
WiFiClient wifiClient;
String data3;
#define ORG "x0fxss"
#define DEVICE_TYPE "Noder"
#define DEVICE_ID "1234"
#define TOKEN "987654321"
#define speed 0.034 #define led 14 char server[] = ORG
".messaging.internetofthings.ibmcloud.com"; char publishTopic[]
= "iot-2/evt/Kaviya_assignment4/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);
```

```

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); pinMode(echopin,
INPUT); wifiConnect();
mqttConnect();
} void loop() { bool
isNearby = dist < 100;
digitalWrite(led,
isNearby); publishData();
delay(500);
if (!client.loop()) {
mqttConnect();
}
}
void wifiConnect() {
Serial.print("Connecting to "); Serial.print("Wifi");
WiFi.begin("Wokwi-GUEST", "", 6); while
(WiFi.status() != WL_CONNECTED) {
delay(500); Serial.print(".");
}
Serial.print("WiFi connected, IP address: "); Serial.println(WiFi.localIP());
}

```

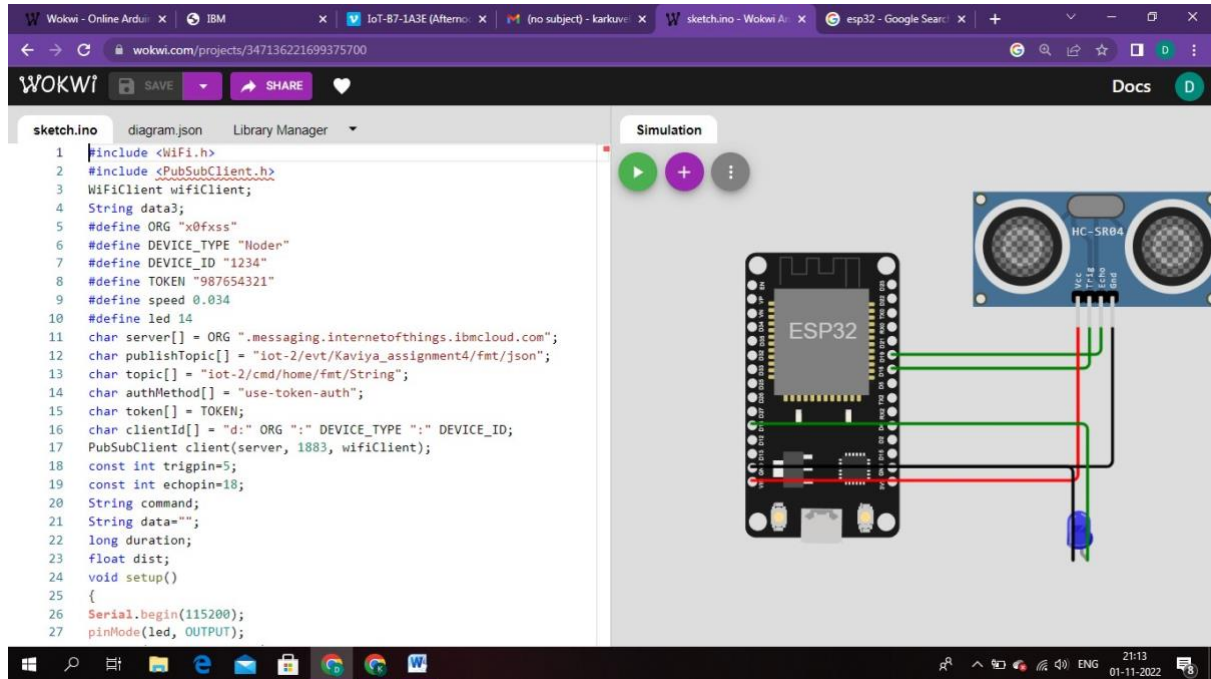
```

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 = "{\"Alert Distance\":";

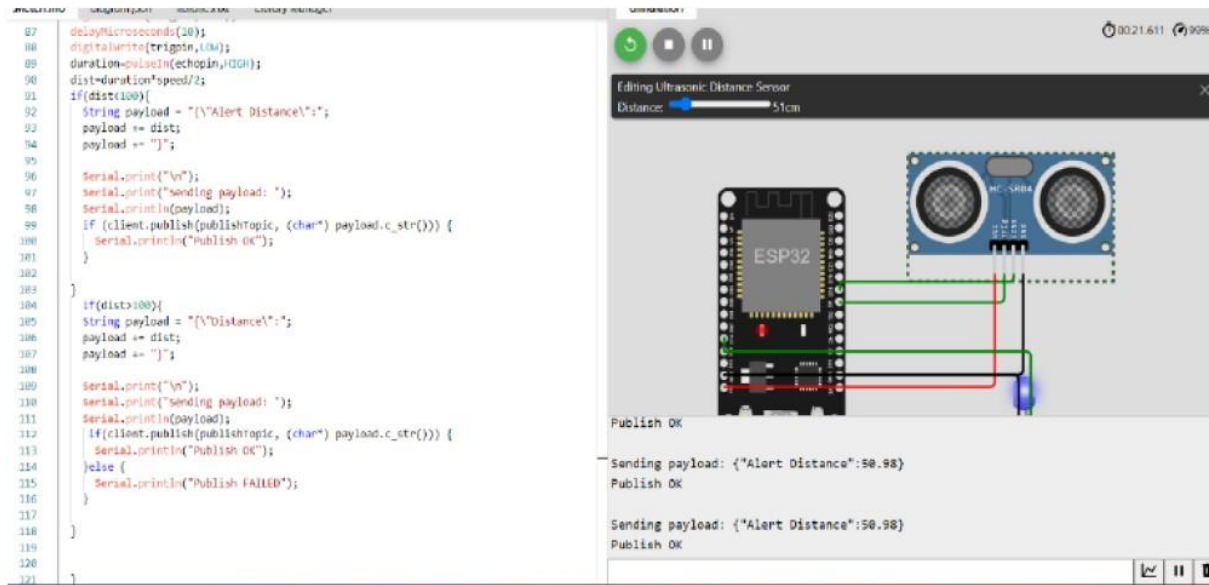
```

```
payload += dist; payload
+= "}";
Serial.print("\n");
Serial.print("Sending payload: "); Serial.println(payload);
if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish OK");
}
} if(dist>100){
String payload = "{\\"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");
}else {
Serial.println("Publish FAILED");
}
}
}
```

## OUTPUT:



A) When distance greater than 100 ,



B) When distance less than 100 ,

The screenshot displays the Arduino IDE interface during a simulation. The left pane shows the code for an ESP32-based distance measurement project. The code includes necessary libraries, defines constants for server, topic, and token, and sets up pins for a trigger and echo. The right pane shows a simulation of the hardware, featuring an ESP32 microcontroller and an HC-SR04 ultrasonic sensor connected via jumper wires. The simulation console on the right shows the sensor's output, indicating a distance of approximately 399 cm.

```
10  diagram.json  libraries.txt  Library Manager  ▾  
  
#include <WiFi.h>  
#include <PubSubClient.h>  
WiFiClient wifiClient;  
String data3;  
#define ORG "x0fxss"  
#define DEVICE_TYPE "Node"  
#define DEVICE_ID "1234"  
#define TOKEN "987654321"  
#define speed 0.034  
#define led 14  
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";  
char publishTopic[] = "iot-2/evt/Kaviya_assignment4/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);  
  
const int trigpin=5;  
const int echopin=18;  
String command;  
String data="";  
  
long duration;  
float dist;
```

Simulation

ESP32

HC-SR04

Publish OK

Sending payload: {"Distance":399.92}

Publish OK

Sending payload: {"Distance":399.96}

Publish OK