

WOKWI SIMULATION

Team ID
Project Name

PNT2022TMID33002
Project-Smart Waste Management System
for Metropolitan Cities

CODE :

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT
#define ECHO_GPIO 12
#define TRIGGER_GPIO 13
#define MAX_DISTANCE_CM 100 // Maximum of 5 meters
#include "Ultrasonic.h"
Ultrasonic ultrasonic(13, 12);
int distance;
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);

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

#define ORG "i3869j" //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/command/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
{
    float w;
    w=random(0,5);
    distance = ultrasonic.read(CM);

    if(distance < 100){
        Serial.print("Distance in CM: ");
        Serial.println(distance);
        Serial.println("Weight in kg: ");
        Serial.println(w);

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

    }

    delay(1000);
}

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

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

    if((distance>=100))
        Serial.println("Trash container is full (100%)");

    else if((distance>=50))
        Serial.println("Trash container is half full (50%)");

    else
        Serial.println("Trash container is not full");
}

```

```

    delay(2000);
}
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);
    if(data3=="lighton")
    {
        Serial.println(data3);
    }
    else
    {
        Serial.println(data3);
    }
    data3="";
}

```

SCREENSHOT :

The screenshot displays the Wokwi IoT simulator interface. On the left, the 'sketch.ino' file is open, showing the following code:

```

1  #include <WiFi.h> //library for wifi
2  #include <PubSubClient.h> //library for MQTT
3
4  #define ECHO_GPIO 12
5  #define TRIGGER_GPIO 13
6  #define MAX_DISTANCE_CM 100 // Maximum of 5 meters
7  #include "Ultrasonic.h"
8
9  Ultrasonic ultrasonic(13, 12);
10 int distance;
11
12 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
13
14 //-----credentials of IBM Accounts-----
15
16 #define ORG "i3869j" //IBM ORGANIZATION ID
17 #define DEVICE_TYPE "abcd" //Device type mentioned in ibm watson IOT Platform
18 #define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
19 #define TOKEN "12345678" //Token
20 String data3;
21 float h,t;
22
23
24
25
26 //----- Customise the above values -----
27 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
28 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform a
29 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND
30 char authMethod[] = "use-token-auth"; // authentication method
31 char token[] = TOKEN;
32 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
33
34

```

On the right, the 'Simulation' window shows a 3D model of the hardware setup. An ESP32 microcontroller is connected to an HC-SR04 ultrasonic sensor and an HX711 load cell amplifier. The amplifier is connected to a trash container with two weight sensors. The simulation output at the bottom shows the following data:

```

Weight in kg:
1.00
Trash container is half full (50%)
Distance in CM: 91
Weight in kg:
4.00
Trash container is half full (50%)

```