Smart City Waste Management System with Connected Trash Cans

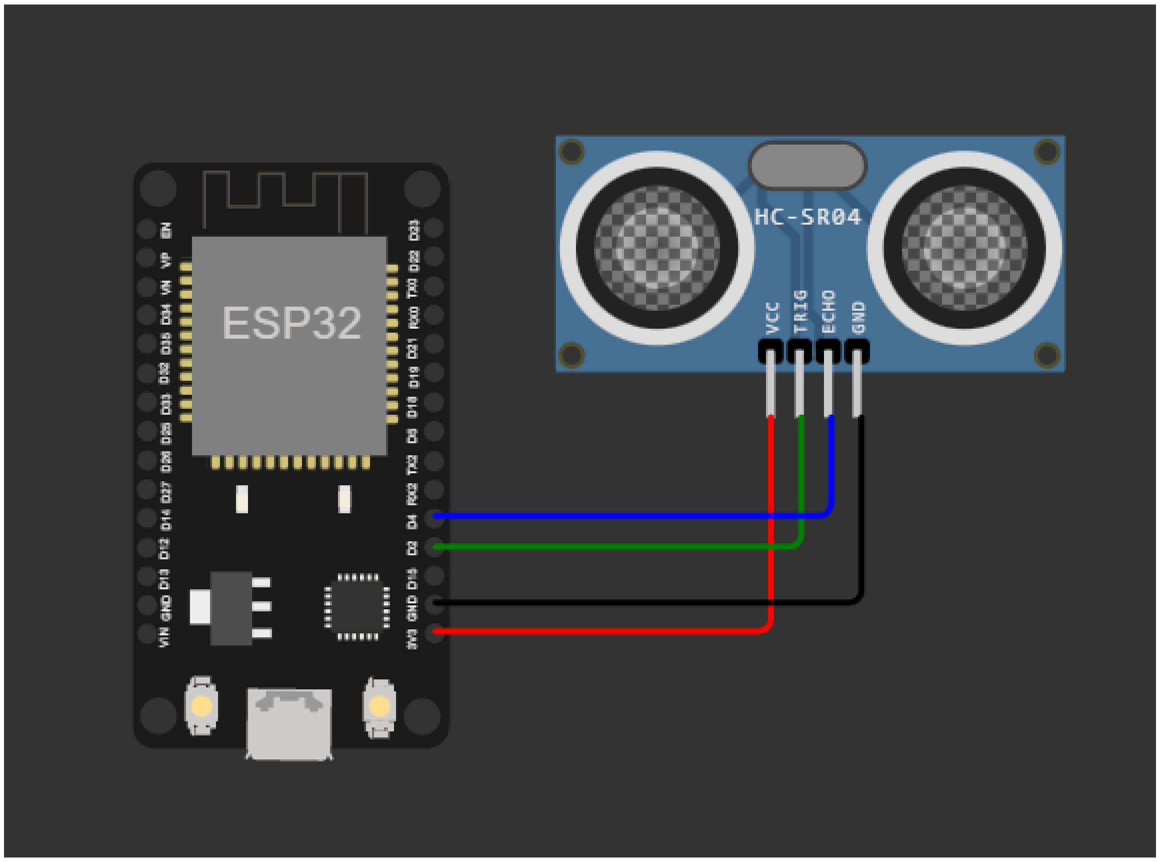
Wokwi Link: https://wokwi.com/projects/364777764114793473

Wokwi Code:

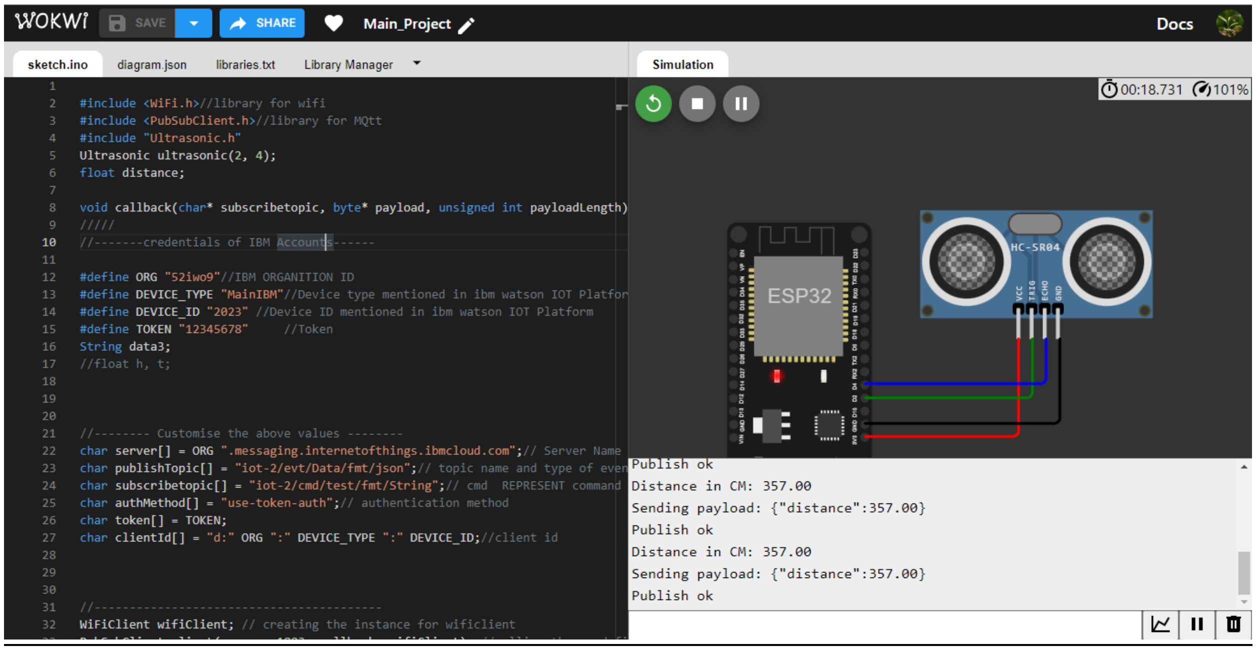
```
#include <WiFi.h>//library for wifi

#include <PubSubClient.h>//library for MQtt

#include "Ultrasonic.h"

Ultrasonic ultrasonic(2, 4);

float distance;


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

/////

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


#define ORG "52iwo9"//IBM ORGANITION ID

#define DEVICE_TYPE "MainIBM"//Device type mentioned in ibm watson IOT Platform

#define DEVICE_ID "2023" //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/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


//-----------------------------------------

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

{


  distance = ultrasonic.read(CM);


  Serial.print("Distance in CM: ");

  Serial.println(distance);
  delay(1000);


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


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


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


  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);
    }

     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);
data3="";
}
```

Wokwi Connection:



Wokwi output:



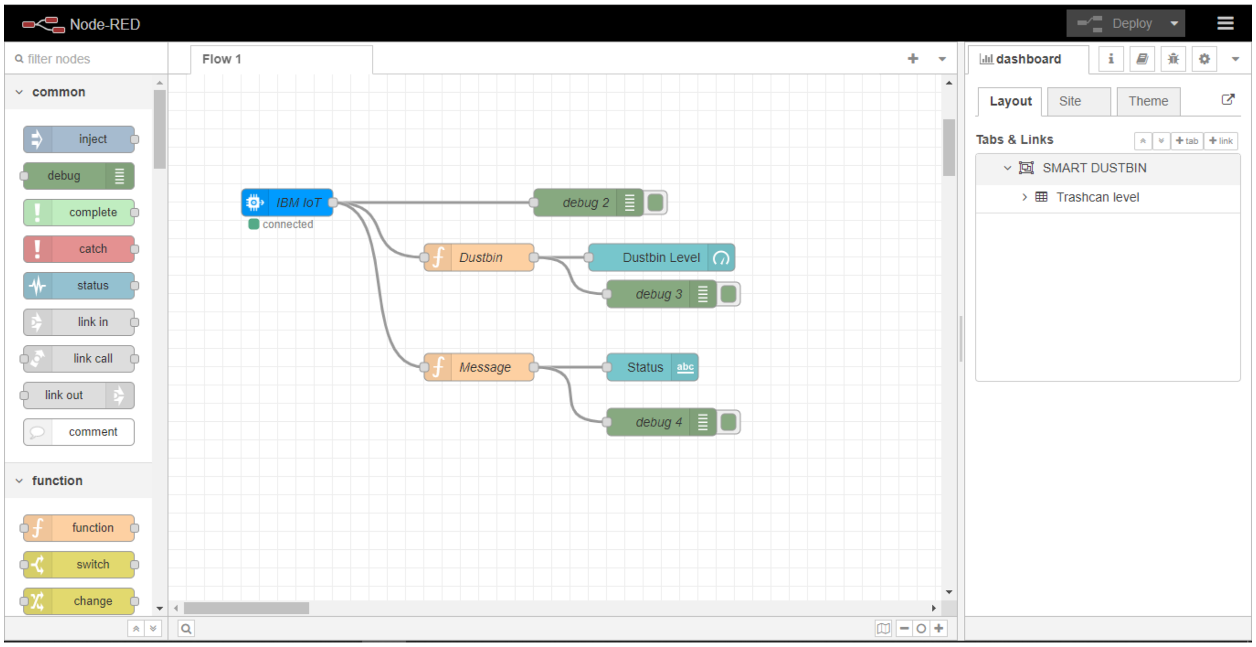
```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "Ultrasonic.h"
Ultrasonic ultrasonic(2, 4);
float distance;

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
/////
//-------credentials of IBM Accounts------

#define ORG "52iwo9"//IBM ORGANITION ID
#define DEVICE_TYPE "MainIBM"//Device type mentioned in ibm watson IOT Platfor
#define DEVICE_ID "2023" //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 even
char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd  REPRESENT command
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
```

```
Publish ok
Distance in CM: 357.00
Sending payload: {"distance":357.00}
Publish ok
Distance in CM: 357.00
Sending payload: {"distance":357.00}
Publish ok
```

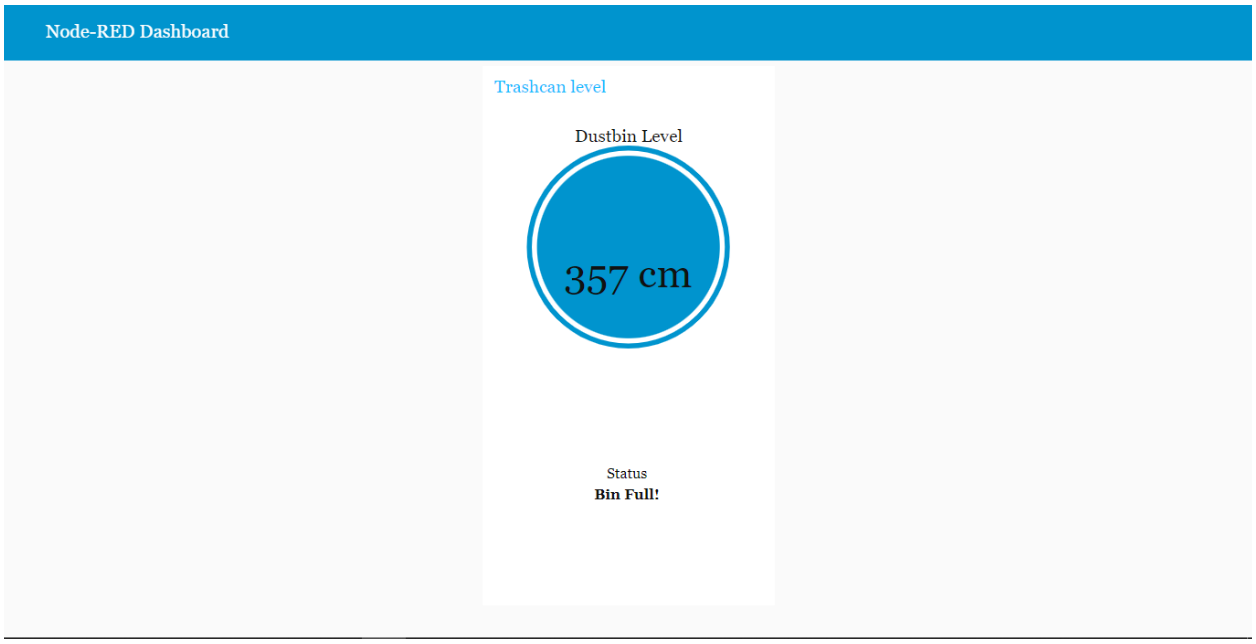
Node-red function code:

```
msg.payload = msg.payload.distance
if (msg.payload <= 4) {
    msg.payload = "Bin Empty!";
}
else if (msg.payload > 350) {
    msg.payload = "Bin Full!";
}
else {
    msg.payload = "Bin can be used!";
}
return msg;
```

Node-red Connection:



Node-red output:


Node-RED Dashboard

Trashcan level

Dustbin Level

357 cm

Status

Bin Full!