

SMART WASTE MANAGEMENT SYSTEM FOR METROPOLITIAN CITIES

ASSIGNMENT-4

TEAM ID : PNT2022TMID04391
NAME : MOHAMED ROSHAN M
ROLL NO : 737819CSR108

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. Upload document with wokwi share link and images of ibmcloud.

CODE:

```
#include <WiFi.h>
#include<WiFiClient.h>
#include<PubSubClient.h>
const int trigPin = 5; const int echoPin = 18;
//define sound speed in cm/uS
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701
long duration;
float distanceCm;
float distanceInch;

void callback(char* subscribetopic, byte* payload, unsigned intpayloadLength);
//-----credentials ofIBMAccounts-----

#define ORG "3defa"
#define DEVICE_TYPE "hariprasath"//Device type mentioned in ibm watson IOTPlatform
#define DEVICE_ID "12345"//Device ID mentioned in ibmwatson IOTPlatform
#define TOKEN "CpL-H1C-Pt4i9iM-F5" //Token
String data3;

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 commandtype AND
COMMAND IS TEST OF FORMAT STRING
char authMethod[]="use-token-auth";

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);
```



```

void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.println();
  wifiConnect();
  mqttConnect();

}

void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_SPEED/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  // Prints the distance in the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);

  PublishData(distanceCm);
  delay(1000);
  if (!client.loop()) {
    mqttConnect();
  }
}

void PublishData(float Cm) {

  mqttConnect();//functioncall for connecting to ibm
  /*
  creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"Distance (cm)\":";
  payload += Cm;
  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 cloudthen it will print publish
  ok in Serial monitor or else it will print publishfailed
}
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 definationforwificonnect
{
  Serial.println();
  Serial.print("Connecting to ");

  WiFi.begin("Wokwi-GUEST", "", 6);//passing thewificredentials toestablishthe connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  Serial.println("IPaddress:");
  Serial.println(WiFi.localIP());
}

void initManagedDevice()
{
  if (client.subscribe(subscribetopic))
  {
    Serial.println((subscribetopic));
    Serial.println("subscribetocmd OK");
  } else
  {
    Serial.println("subscribetocmd FAILED");
  }
}

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

```

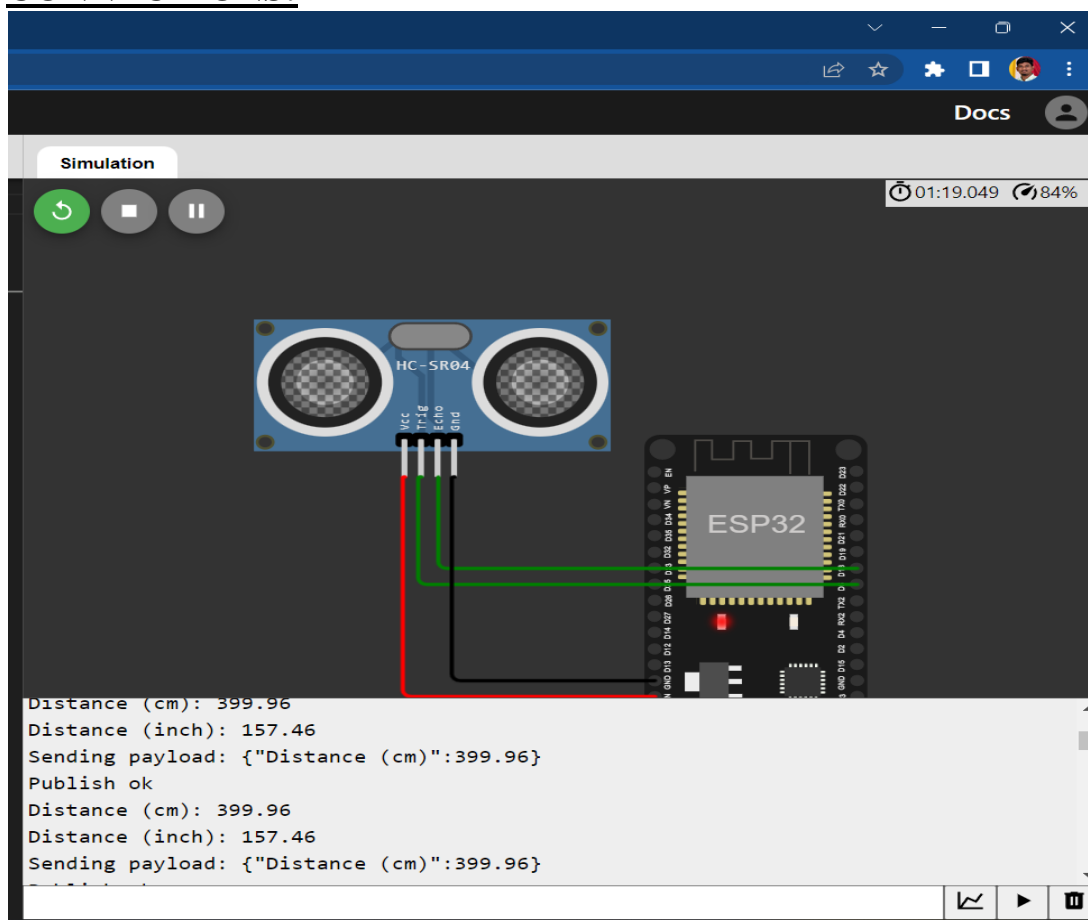


```

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];
}
}
}

```

CONNECTIONS:



WOKWI LINK:

Code link:

<https://wokwi.com/projects/347500360707867219>



Edit with WPS Office

OUTPUT:

Wokwi IDE interface showing the initial setup of an ESP32 simulation. The code on the left defines variables for server, topic, token, and client ID, and sets up a callback function. The simulation window on the right shows the hardware components (HC-SR04 sensor and ESP32) and the console output indicating a successful connection to the IBM Watson IoT Platform.

```
1 #include <WiFi.h>
2 #include <WiFiClient.h>
3 #include <PubSubClient.h>
4 const int trigPin = 5; const int echoPin = 18;
5 //define sound speed in cm/us
6 #define SOUND_SPEED 0.034
7 #define CM_TO_INCH 0.393701
8 long duration;
9 float distanceCm;
10 float distanceInch;
11
12 void callback(char* subscribtopic, byte* payload, unsigned intpayloadlength);
13 //-----credentials ofIBMAccounts-----
14
15 #define ORG "3defa"
16 #define DEVICE_TYPE "hariprasath"//Device type mentioned in ibm watson IOTPlatform
17 #define DEVICE_ID "12345"//Device ID mentioned in ibm watson IOTPlatform
18 #define TOKEN "cpl-HIC-PT419IM-F5" //token
19 String data;
20
21 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
22 char publishTopic[] = "iot-2/evt/data/fmt/json"; // topic name and type of event perform a
23 char subscribtopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT commandtype AND COM
24 char authMethod[] = "use-token-auth";
25
26 char token[] = TOKEN;
27 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
28
29 WiFiClient wifiClient; // creating the instance for wifiClient
30 PubSubClient client(server, 1883, callback ,wifiClient);
31
32 void setup() {
33   Serial.begin(115200); // Starts the serial communication
34   pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
35   pinMode(echoPin, INPUT); // Sets the echoPin as an Input
36 }
```

Connecting to
Wifi connected
IPAddress:
10.10.0.2
Reconnecting client to 3defa.messaging.internetofthings.ibmcloud.com
iot-2/cmd/test/fmt/String
subscribtopiccmd OK

Wokwi IDE interface showing the simulation results. The code on the left includes the setup and loop functions. The simulation window on the right shows the hardware components (HC-SR04 sensor and ESP32) and the console output displaying the distance measurements in inches and centimeters, and the payload being sent to the IoT platform.

```
19 String data;
20
21 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
22 char publishTopic[] = "iot-2/evt/data/fmt/json"; // topic name and type of event perform a
23 char subscribtopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT commandtype AND COM
24 char authMethod[] = "use-token-auth";
25
26 char token[] = TOKEN;
27 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
28
29 WiFiClient wifiClient; // creating the instance for wifiClient
30 PubSubClient client(server, 1883, callback ,wifiClient);
31
32 void setup() {
33   Serial.begin(115200); // Starts the serial communication
34   pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
35   pinMode(echoPin, INPUT); // Sets the echoPin as an Input
36   Serial.println();
37   wifiConnect();
38   mqttConnect();
39 }
40
41 void loop() {
42   // Clears the trigPin
43   digitalWrite(trigPin, LOW);
44   delayMicroseconds(2);
45   // Sets the trigPin on HIGH state for 10 micro seconds
46   digitalWrite(trigPin, HIGH);
47   delayMicroseconds(10);
48   digitalWrite(trigPin, LOW);
49
50   // Reads the echoPin, returns the sound wave travel time in microseconds
51   duration = pulseIn(echoPin, HIGH);
52 }
```

Distance (inch): 157.46
Sending payload: {"Distance (cm)":399.96}
Publish ok
Distance (cm): 399.96
Distance (inch): 157.46
Sending payload: {"Distance (cm)":399.96}
Publish ok



IBM Watson IOT platform connected:

The screenshot displays the IBM Watson IoT Platform interface. At the top, there's a navigation bar with tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for navigation. The main content area shows a device named '12345' with a status of 'Connected'. Below this, there's a section for 'Recent Events' which displays a table of data events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. The events show a stream of data with values like '{"Distance (cm)":399.96}' in JSON format, received a few seconds ago.

Event	Value	Format	Last Received
Data	{"Distance (cm)":399.96}	json	a few seconds ago
Data	{"Distance (cm)":399.96}	json	a few seconds ago
Data	{"Distance (cm)":399.96}	json	a few seconds ago
Data	{"Distance (cm)":399.96}	json	a few seconds ago