

ASSIGNMENT-4

Assignment Date	26 October 2022
Student Name	PADMAPRIYA.P
Team Id	PNT2022TMID30034
Maximum Marks 2	2 Marks

Question:

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 ibm cloud.

Solution:

Wokwi link: <https://wokwi.com/projects/347114584593662547>

The screenshot displays the Wokwi IDE interface. On the left, the code editor shows a C++ program for an ESP32. The code includes libraries for WiFi, WiFiClient, and PubSubClient. It defines pins for the ultrasonic sensor (trigPin = 5, echoPin = 18) and sets the speed of sound. The main logic is in the void callback function, which sends distance data to IBM Cloud IoT Platform using the PubSubClient library. The code includes comments for credentials and device information.

```
1 #include <WiFi.h>
2 #include <WiFiClient.h>
3 #include <PubSubClient.h>
4 const int trigPin = 5;
5 const int echoPin = 18;
6 //define sound speed in cm/uS
7 #define SOUND_SPEED 0.034
8 #define CM_TO_INCH 0.393701
9 long duration;
10 float distanceCm;
11 float distanceInch;
12
13 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
14 {
15     //-----credentials of IBM Accounts-----
16
17     #define ORG "evpj6t"//IBM ORGANITION ID
18     #define DEVICE_TYPE "priya"//Device type mentioned in ibm watson IOT Platform
19     #define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
20     #define TOKEN "12345678" //Token
21     String data3;
22
23
24
25     //----- Customise the above values -----
26     char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
27     char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of even
28     char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd REPRESENT command
29     char authMethod[] = "use-token-auth";// authentication method
30     char token[] = TOKEN;
```

On the right, the simulation window shows a visual representation of the ESP32 and the ultrasonic sensor. Below the simulation, the console output displays the distance readings and the data being sent to IBM Cloud:

```
Distance (inch): 85.41
Sending payload: {"Distance (cm)":216.94}
Publish ok
Distance (cm): 216.94
Distance (inch): 85.41
Sending payload: {"Distance (cm)":216.94}
Publish ok
```

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 int payloadLength);
//-----credentials of IBM Accounts-----

#define ORG "evpj6t"//IBM ORGANITION ID
#define DEVICE_TYPE "priya"//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;

//----- 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);

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();//function call 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 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);  
        }  
    }  
}
```

Image of cloud:

