

ASSIGNMENT-4

Assignment Date	15 October2022
Student Name	Anuvratha Vijay
Student Roll Number	715519106003
Maximum marks	2 marks

Question: Write code and connections in wokwi for the ultrasonic sensor. Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display in the device recent events. Upload document with wokwi share link and images of IBM cloud

PROGRAM:

```
#include <WiFi.h>
#include<PubSubClient.h>
#include <ArduinoJson.h>

void callback(char* subscribetopic,byte* payload,unsigned int payloadLength);
#define ORG "o16sqk"
#define DEVICE_TYPE "NodeMCU"
#define DEVICE_ID "12345"
#define TOKEN "715519106004"
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701

const int trigPin = 5;
const int echoPin = 18;

long duration;
float distanceCm;
float distanceInch;
String data;

char server[]=ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/event_1/fmt/json";
char subscribeTopic[] = "iot-2/cmd/home/fmt/String";
char authMethod[] = "use-token-auth";
char token[] =TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

WiFiClient 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
  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);

    delay(1000);

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

void PublishData(float distanceCm)
{

    mqttconnect ();
    String payload;

    if(distanceCm<100.0)
    {
        payload = "{\"Alert\":\"";
        payload += distanceCm;
        payload += "\"}";
    }
    else
    {
        payload = "{\"distanceCm\":\"";
        payload += distanceCm;
        payload += "\"}";
    }
}

```

```
}
```

```
Serial.print("Sending payload: ");  
Serial.print(payload);
```

```
if(client.publish(publishTopic , (char*) payload.c_str())){  
  Serial.println("Publish ok");}  
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()  
{
```

```
  Serial.println();  
  Serial.print("Connecting to");
```

```
  WiFi.begin("Wokwi-GUEST","",6);  
  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++)
    {
        data +=(char)payload[i];
    }
    Serial.println("data: "+data);
    if(data>(char)100)
    {
        Serial.println("Alert!");
    }*/
}

```

CIRCUIT:

The screenshot displays the Wokwi web-based simulation environment. On the left, the code editor shows an Arduino sketch for an ESP32 microcontroller. The code includes necessary libraries, defines constants for the sensor pins and I2C address, and implements a callback function for the ultrasonic sensor. The main function sets up the serial port and the sensor, and then enters a loop that triggers the sensor and prints the distance to the serial monitor.

On the right, the simulation window shows the physical connection between the ESP32 and the HC-SR04 ultrasonic sensor. The sensor is connected to the ESP32's pins: VCC to 5V, GND to GND, Trig to pin 5, and Echo to pin 18. The simulation output at the bottom shows the sensor's readings and the corresponding JSON payloads being sent over the I2C interface.

```

1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 #include <ArduinoJson.h>
4
5 void callback(char* subscribetopic,byte* payload,unsigned int payloadLength)
6 #define ORG "o16sqk"
7 #define DEVICE_TYPE "NodeMCU"
8 #define DEVICE_ID "12345"
9 #define TOKEN "715519106004"
10 #define SOUND_SPEED 0.034
11 #define CM_TO_INCH 0.393701
12
13 const int trigPin = 5;
14 const int echoPin = 18;
15
16 long duration;
17 float distanceCm;
18 float distanceInch;
19 String data;
20
21 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
22 char publishTopic[] = "iot-2/evt/event_1/fmt/json";
23 char subscribeTopic[] = "iot-2/cmd/home/fmt/String";
24 char authMethod[] = "use-token-auth";
25 char token[] = TOKEN;
26 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
27

```

Simulation output:

```

Sending payload: {"distanceCm":399.92}Publish ok
Distance (cm): 399.96
Sending payload: {"distanceCm":399.96}Publish ok

```

OUTPUT:

The screenshot displays the Wokwi web-based simulation environment. On the left, the code for `wifi_scan.ino` is shown, which includes libraries for WiFi, PubSubClient, and ArduinoJson. It defines an IoT device with ID '12345' and publishes distance data to the topic `iot-2/evt/event_1/fmt/json`. The simulation window on the right shows an ESP32 microcontroller connected to an HC-SR04 ultrasonic sensor. The output log displays the following messages:

```
Sending payload: {"distanceCm":399.96}Publish ok
Distance (cm): 399.92
Sending payload: {"distanceCm":399.92}Publish ok
```

The screenshot shows the IBM Watson IoT Platform dashboard. The page title is 'Device Drilldown - 12345'. The left sidebar contains navigation options: Back, Recent Events, State, Device Information, Metadata, Diagnostics, Connection Logs, and Device Actions. The main content area displays the 'Recent Events' section, which shows a live stream of data from the device. The events are listed in a table:

Event	Value	Format	Last Received
event_1	{"distanceCm":399.98}	json	a few seconds ago
event_1	{"distanceCm":399.92}	json	a few seconds ago
event_1	{"distanceCm":399.96}	json	a few seconds ago
event_1	{"distanceCm":400.18}	json	a few seconds ago
event_1	{"distanceCm":399.96}	json	a few seconds ago

IBM Watson IoT Platform

715519106003@smartinternz.com
ID: o16sqk

← Back

Device Drilldown - 12345

- Connection Information
- Recent Events
- State
- Device Information
- Metadata
- Diagnostics
- Connection Logs
- Device Actions

State

This table shows a list of data points that are reported by this device.

Showing Raw Data | No Interfaces Available

Property	Value	Type	Event	Last Received
distanceCm	399.98	Number	event_1	a few seconds ago

Device Information

View basic device information including location and manufacturer.

Edit Device Information

LINK: <https://wokwi.com/projects/347387564691292754>