

ASSIGNMENT-4

Ultrasonic sensor simulation in Wokwi

Team ID	PNT2022TMID48307
Title	INDUSTRY SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM
Name	Narmatha.P
Minimum Marks	2 Marks

Question:

Write a code and connections in Wokwi for the Ultrasonic sensor.
Whenever the distance is less than 100cms send an “Alert” to IBM cloud and display in the device recent events.

Code:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "4e8u9a"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "E8-30-FC-C9-9B-01"//Device ID mentioned in ibm watson IOT Platform
#define TOK"xDO2Cr3KSS@KSS@!f6&TBV" //Token
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/test/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);
const int trigPin = 5;
const int echoPin = 18;
#define SOUND_SPEED 0.034
long duration;
float distance;
void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  wificonnect();
  mqttconnect();
}
```

```

void loop()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * SOUND_SPEED/2;
  Serial.print("Distance (cm): ");
  Serial.println(distance);
  if(distance>100)
  {
    Serial.println("ALERT!!");
    delay(100);
    PublishData(distance);
    delay(100);
    if (!client.loop()) {
      mqttconnect();
    }
  }
  delay(100);
}

void PublishData(float dist) {
  mqttconnect();
  String payload = "{\"Distance\": ";
  payload += dist;
  payload += ", \"ALERT!!\": \"\" \"Distance less than 100cms\"";
  payload += "}";
  Serial.print("Sending payload: ");
  Serial.println(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(100);
    }
    initManagedDevice();
    Serial.println();
  }
}

void wificonnect()
{
  Serial.println();
  Serial.print("Connecting to ");
  WiFi.begin("Wokwi-GUEST", "", 6);
  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
    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="";
}

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

void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "d19wub"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "3C-91-80-49-01-C9"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "cE&QcASnabqYe18-1f" //Token
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/test/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);
const int trigPin = 5;
const int echoPin = 18;
#define SOUND_SPEED 0.034
long duration;
float distance;
void setup() {
Serial.begin(115200);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
wificonnect();
mqttconnect();
}

void loop()
{
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = duration * SOUND_SPEED/2;
Serial.print("Distance (cm): ");

```

```

Serial.println(distance);
if(distance>100)
{
Serial.println("ALERT!!");
delay(100);
PublishData(distance);
delay(100);
if (!client.loop()) {
mqttconnect();
}
}
delay(100);
}
void PublishData(float dist) {
mqttconnect();
String payload = "{\"Distance\":\"";
payload += dist;
payload += "\",\"ALERT!!\":\"\"Distance less than 100cms\"";
payload += "\"";
Serial.print("Sending payload: ");
Serial.println(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(100);
}
initManagedDevice();
Serial.println();
}
}
void wificonnect()
{
Serial.println();
Serial.print("Connecting to ");
WiFi.begin("Wokwi-GUEST", "", 6);
while (WiFi.status() != WL_CONNECTED) {
delay(100);
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="";
}

```

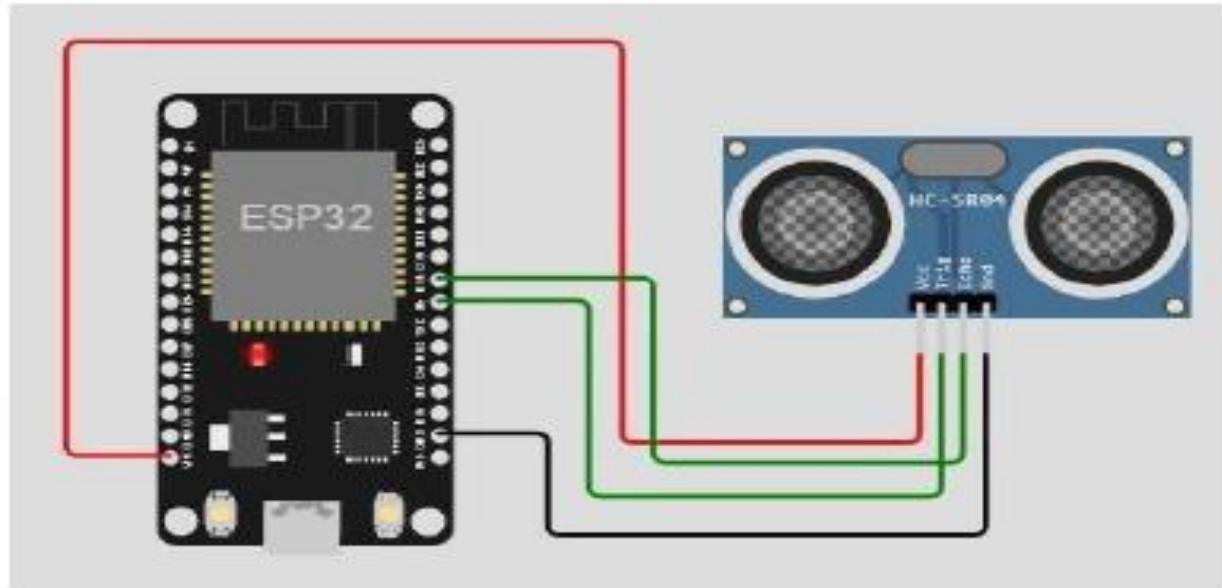
Diagram.json:

```

{
  "version": 1,
  "author": "sweetysharon",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -4.67, "left": -112.87, "attrs": {} },
    { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": 15.96, "left": 89.17, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [
      "esp:VIN",
      "ultrasonic1:VCC",
      "red",
      [ "h-37.16", "v-178.79", "h200", "v173.33", "h100.67" ]
    ],
    [ "esp:GND.1", "ultrasonic1:GND", "black", [ "h39.87", "v44.04", "h170" ] ],
    [ "esp:D5", "ultrasonic1:TRIG", "green", [ "h54.54", "v85.07", "h130.67" ] ],
    [ "esp:D18", "ultrasonic1:ECHO", "green", [ "h77.87", "v80.01", "h110" ] ]
  ]
}

```

Circuit Diagram:



Output:

Service Details - IBM Cloud | IBM Watson IoT Platform | sketchino - Wokwi Arduino and | +

wokwi.com/projects/346508314441417298

WOKWI SAVE SHARE

Docs SIGN IN

sketchino • diagram.json libraries.txt Library Manager

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int
4   payloadLength);
5 //-----credentials of IBM Accounts-----
6 #define ORG "4e8u9a"//IBM ORGANITION ID
7 #define DEVICE_TYPE "ESP32"//Device type mentioned in ibm iot Platform
8 #define DEVICE_ID "EB16CE00-4492-4719-9691-AF94AC997A88"//Device ID mentioned in ibm wat
9 #define TOKEN "lafad60g2c09jPhz5" //Token
10 String data3;
11 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
12 char publishTopic[] = "iot-2/evt/Data/fmt/json";
13 char subscribTopic[] = "iot-2/cmd/test/fmt/String";
14 char authMethod[] = "use-token-auth";
15 char token[] = TOKEN;
16 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
17 WiFiClient wifiClient;
18 PubSubClient client(server, 1883, callback ,wifiClient);
19 const int trigPin = 5;
20 const int echoPin = 18;
21 #define SOUND_SPEED 0.034
22 long duration;
23 float distance;
24 void setup() {
25   Serial.begin(115200);
26   pinMode(trigPin, OUTPUT);
27   pinMode(echoPin, INPUT);
28   wifiConnect();
29   mqttconnect();
30 }
31 void loop()
32 {
33   digitalWrite(trigPin, LOW);
34   delayMicroseconds(2);
35   digitalWrite(trigPin, HIGH);
36   delayMicroseconds(10);
```

Simulation

00:04.798 68%

Distance (cm): 399.94
ALERT!!
Sending payload: {"Distance":399.94,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 399.94
ALERT!!
Sending payload: {"Distance":399.94,"ALERT!!":"Distance less than 100cms"}

IBM Cloud Output:

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes tabs for 'Service Details - IBM Cloud', 'IBM Watson IoT Platform', and 'sketchuno - Wokwi Arduino and'. The main header shows the user's email '913019106004@smartinternz.com' and ID '4e8u9a'. The left sidebar contains icons for various platform features. The main content area is titled 'IBM Watson IoT Platform' and shows the 'Browse' tab selected. A device card for 'E8-D0-FC-C9-98-01' is displayed, indicating it is 'Connected' and an 'ESP32' device. Below the device card, the 'Logs' tab is active, showing two sections: 'Diagnostic Logs' and 'Connection Logs'. The 'Diagnostic Logs' section states 'No logs are available.' and features a large icon of a person with a gear. The 'Connection Logs' section lists three events: 'Token auth succeeded: ClientID=d:4e8u9...', 'Closed connection. The connection timed ...', and 'Token auth succeeded: ClientID=d:4e8u9...'. The 'Add Device' button is visible in the top right corner.

Severity	Message	Timestamp
No logs are available.		

Message	Timestamp
Token auth succeeded: ClientID=d:4e8u9...	Nov 7, 2022 2:34 AM
Closed connection. The connection timed ...	Nov 7, 2022 2:33 AM
Token auth succeeded: ClientID=d:4e8u9...	Nov 7, 2022 2:33 AM