

Assignment - 4
ESP 32 – Ultrasonic Sensor

| | |
|---------------------|-----------------|
| Assignment Date | 3 NOVEMBER 2022 |
| Student Name | LATHA V |
| Student Roll Number | 621319106052 |
| Maximum Marks | 2 Marks |

Question-1:

Write code and Connection in wokwi for ultrasonic sensor. Whenever distance is less than 100cms send “alert” to the ibm cloud and display in device recent events.

Solution:

Program:

```
#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 "b31tni"//IBM ORGANITION ID
#define DEVICE_TYPE "Assignment4"//Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "assignment"//Device ID mentioned in ibm watson IOT
Platform#define TOKEN "6r?TKCIuy+okJ?9B+7" //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);
        }

        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];
    }
}

```

Wokwi Simulation:

The screenshot shows the Wokwi simulation environment. On the left, the code editor displays the `sketch.ino` file with C++ code for an ESP32. On the right, the simulation window shows an ESP32 Dev Board connected to an HC-SR04 ultrasonic sensor. The sensor's trig pin is connected to digital pin 5, and its echo pin is connected to digital pin 18. The simulation interface includes a toolbar at the top and a library manager at the bottom.

```

1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribetopic, byte* payload, unsigned int
4 payloadLength);
5 //-----Credentials of IBM Accounts-----
6 #define ORG "0lxcom"//IBM ORGANITION ID
7 #define DEVICE_TYPE "ESP32PROJECT"//Device type mentioned in ibm watson IOT Platform
8 #define DEVICE_ID "ESP32"//Device ID mentioned in ibm watson IOT Platform
9 #define TOKEN "ESP32PROJECT" //Token
10 String data3;
11 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
12 char publishTopic[] = "iot-2/evt/data/fmt/json";
13 char subscribetopic[] = "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 }
```

wokwi.com/projects/347290193940709972

Apps Gmail YouTube LinkedIn IBM Tinkercad GitHub Rocket chat cloud ibm

WOKWI SAVE SHARE ❤ Docs SIGN UP

sketch.ino diagram.json libraries.txt Library Manager ▾

Simulation

00:10.729 100%

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
4 //-----credentials of IBM Accounts-----
5 #define ORG "9lxobn"/IBM ORGANITION ID
6 #define DEVICE_TYPE "ESP32PROJECT"/Device type mentioned in ibm watson IOT Platform
7 #define DEVICE_ID "ESP32"/Device ID mentioned in ibm watson IOT Platform
8 #define TOKEN "ESP32PROJECT" //Token
9 String data3;
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/data/fmt/json";
12 char subscribetopic[] = "iot-2/cmd/test/fmt/String";
13 char authMethod[] = "use-token-auth";
14 char token[] = TOKEN;
15 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
16 WiFiClient wifiClient;
17 PubSubClient client(server, 1883, callback ,wifiClient);
18 const int trigPin = 5;
19 const int echoPin = 18;
20 #define SOUND_SPEED 0.034
21 long duration;
22 float distance;
23 void setup() {
24 Serial.begin(115200);
25 pinMode(trigPin, OUTPUT);
26 pinMode(echoPin, INPUT);
27 wificonnect();
28 mqttconnect();
29 }
30 void loop()
31 {
32 digitalWrite(trigPin, LOW);
33 delayMicroseconds(2);
34 }
```

Connecting to
WiFi connected
IP address:
10.10.0.2
Reconnecting client to 9lxobn.messaging.internetofthings.ibmcloud.com
iot-2/cmd/test/fmt/String
subscribe to cmd OK

Distance (cm): 399.92
Distance (cm): 399.94
Distance (cm): 399.96
Distance (cm): 399.94
Distance (cm): 399.92
Distance (cm): 399.94
Distance (cm): 399.96
Distance (cm): 399.96

IoT Watson Platform:

The screenshot shows the IoT Watson Platform interface. On the left is a sidebar with various icons. The main area has a header with 'Browse' (selected), 'Action', 'Device Types', 'Interfaces', and an 'Add Device' button. Below the header is a table with columns: Device ID, Status, Device Type, Class ID, Date Added, and Descriptive Location. A single row is selected, showing Device ID 567890, Status Disconnected, Device Type 908, Class ID Device, Date Added Nov 3, 2022 4:21 PM, and Descriptive Location. Below the table are tabs: Identity, Device Information, Recent Events (selected), State, and Logs. A message says 'The recent events listed show the live stream of data that is coming and going from this device.' Below this is a table of recent events with columns: Event, Value, Format, and Last Received. Four entries are shown, all in JSON format and received 'a few seconds ago':

| Event | Value | Format | Last Received |
|---------|--|--------|-------------------|
| event_1 | {"randomNumber":31,"temp":55,"hum":92} | json | a few seconds ago |
| event_1 | {"randomNumber":84,"temp":21,"hum":78} | json | a few seconds ago |
| event_1 | {"randomNumber":57,"temp":27,"hum":94} | json | a few seconds ago |
| event_1 | {"randomNumber":99,"temp":3,"hum":96} | json | a few seconds ago |

At the bottom right, it says '1 Simulation running'. At the bottom left, there's a pagination control 'Items per page 50 ▾ | 1–1 of 1 item'.

<https://wokwi.com/projects/347290193940709972>