

## Assignment-4

### Distance Detection Using Ultrasonic Sensor

Assignment Date	05 November 2022
Student Name	SNEKA T
Student Roll Number	621319106090
Maximum Marks	2 Marks

#### Question-1:

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

WOKWI LINK : <https://wokwi.com/projects/347459854010417746>

#### CODE:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
WiFiClient wifiClient;
#define ORG "9tg03j"
#define DEVICE_TYPE "RaspberryPi"
#define DEVICE_ID "12345"
#define TOKEN "12345678"
#define speed 0.034
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/status1/fmt/json";
char topic[] = "iot-2/cmd/home/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
PubSubClient client(server, 1883, wifiClient);
void publishData();
const int trigpin=5;
const int echopin=19;
String command;
String data="";
String name="Alert";
String icon="";
long duration;
int dist;
void setup()
{
  Serial.begin(115200);
  pinMode(trigpin, OUTPUT);
  pinMode(echopin, INPUT);
  wifiConnect();
  mqttConnect();
}
void loop() {
  publishData();
  delay(500);
  if (!client.loop()) {
    mqttConnect();
  }
}
void wifiConnect() {
  Serial.print("Connecting to "); Serial.print("Wifi");
```

```

WiFi.begin("Wokwi-GUEST", "", 6);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.print("WiFi connected, IP address: "); Serial.println(WiFi.localIP());
}
void mqttConnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting MQTT client to "); Serial.println(server);
    while (!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      Serial.print("*");
      delay(1000);
    }
    initManagedDevice();
    Serial.println();
  }
}
void initManagedDevice() {
  if (client.subscribe(topic)) {
    Serial.println(client.subscribe(topic));
    Serial.println("subscribe to cmd OK");
  }
  else {
    Serial.println("subscribe to cmd FAILED");
  }
}
void publishData()
{
  digitalWrite(trigpin,LOW);
  digitalWrite(trigpin,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin,LOW);
  duration=pulseIn(echopin,HIGH);
  dist=duration*speed/2;
  if(dist<100){
    dist=100-dist;
    icon="Not-Crashed";
  }
  else{
    dist=0;
    icon="Crashed";
  }
  DynamicJsonDocument doc(1024);
  String payload;
  doc["Name"]=name;
  doc["Impact"]=icon;
  doc["Distance"]=dist;
  serializeJson(doc, payload);
  delay(3000);
  Serial.print("\n");
  Serial.print("Sending payload: ");
  Serial.println(payload);
  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish OK");
  }
  else {
    Serial.println("Publish FAILED");
  }
}

```

OUTPUT:

The screenshot displays the Arduino IDE interface. On the left, the 'sketch.ino' file is open, showing a C++ program that configures an ESP32 to connect to IBM Cloud IoT via MQTT. The code includes headers for WiFi, PubSubClient, and ArduinoJson, and defines device-specific constants like ORG, DEVICE\_TYPE, DEVICE\_ID, and TOKEN. It sets up a server URL, topics, and authentication methods. The setup function initializes the serial port and pins, while the loop function publishes data at a specified interval.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 #include <ArduinoJson.h>
4 WiFiClient wifiClient;
5 #define ORG "9tg03j"
6 #define DEVICE_TYPE "RaspberryPi"
7 #define DEVICE_ID "12345"
8 #define TOKEN "12345678"
9 #define speed 0.034
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/status1/fmt/json";
12 char topic[] = "iot-2/cmd/home/fmt/String";
13 char authMethod[] = "use-token-auth";
14 char token[] = TOKEN;
15 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
16 PubSubClient client(server, 1883, wifiClient);
17 void publishData();
18 const int trigpin=5;
19 const int echopin=19;
20 String command;
21 String data="";
22 String name="Alert";
23 String icon="";
24 long duration;
25 int dist;
26 void setup()
27 {
28   Serial.begin(115200);
29   pinMode(trigpin, OUTPUT);
30   pinMode(echopin, INPUT);
31   wifiConnect();
32   mqttConnect();
33 }
```

On the right, the 'Simulation' window shows a visual representation of the ESP32 board connected to a red LED. The simulation is running, as indicated by the play button icon. The output console at the bottom shows the MQTT payload being sent: {"Name": "Alert", "Impact": "Not-Crashed", "Distance": 100}. The status bar at the bottom right shows a battery level of 12% and a timer of 00:21.655.

The screenshot displays the Cisco Duo interface for a device named "distance-detection". The interface includes a sidebar with navigation icons, a top navigation bar with "Browse", "Action", "Device Types", and "Interfaces", and a right-hand "Add Device" button. The main content area shows the "Recent Events" tab, which lists a stream of data events. The events are displayed in a table with columns: Event, Value, Format, and Last Received. The table shows five entries, all with a value of {"distance":235.02,"object":"No"} and a format of json. A status bar at the bottom indicates "0 Simulations running".

Event	Value	Format	Last Received
Data	{"distance":235.02,"object":"No"}	json	a few seconds ago
Data	{"distance":235.02,"object":"No"}	json	a few seconds ago
Data	{"distance":235.02,"object":"No"}	json	a few seconds ago
Data	{"distance":235.02,"object":"No"}	json	a few seconds ago
Data	{"distance":235.02,"object":"No"}	json	a few seconds ago

0 Simulations running



