

## Assignment -4

### Distance Detection Using Ultrasonic Sensor

Assignment Date	26 October 2022
Student Name	MAHALAKSHMI B
Student Roll Number	830119106019
Maximum Marks	2 Marks

#### Question-1:

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

WOKWI LINK: <https://wokwi.com/projects/346574219953308244>

#### CODE:

```
#include <WiFi.h> //library for wifi #include
<PubSubClient.h> //library for MQTT

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-----credentials of IBM Accounts-----

#define ORG "f59trs" //IBM ORGANITION ID
#define DEVICE_TYPE "ultrasonicsensor" //Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "distancedetection" //Device ID mentioned in ibm watson IOT
Platform
#define TOKEN "AIGMGaaF01nawa1QA3" //Token
String data3;
float dist;

//----- 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; //clientid

//
WiFiClient wifiClient; // creating the instance for wificlient
```

```

PubSubClient client(server, 1883, callback, wifiClient);
//calling the predefined client id by passing parameter likeserver id,portand
wificredential

int LED = 4; int
trig = 5; int echo =
18; void setup()
{
  Serial.begin(115200);
  pinMode(trig, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(LED, OUTPUT);
  delay(10); wificonnect();
  mqttconnect();
}
void loop() // Recursive Function
{

  digitalWrite(trig, LOW);
  digitalWrite(trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  float dur = pulseIn(echo, HIGH); float dist
= (dur * 0.0343) / 2; Serial.print
("Distance in cm"); Serial.println(dist);

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

/* ..... retrieving to
Cloud ..... */

void PublishData(float dist) { mqttconnect(); //function call for connecting
to ibm
/*
  creating the String in in form JSon to update the data to ibm cloud
*/
  String object;

```

```

if(dist <100)
{
    digitalWrite(LED,HIGH);
    Serial.println("object is near");object =
    "Near";
}
else
{
    digitalWrite(LED,LOW); Serial.println("no
    object found");object = "No";
}

String payload = "{\"distance\":";payload +=
dist;
payload += ", \"object\":\":";payload +=
object;
payload += "\"}";

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

if(client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload dataon 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];
    }

    // Serial.println("data: " + data3);
    // if(data3=="Near")
    // {
    // Serial.println(data3);
    // digitalWrite(LED,HIGH);

    // }

    // else
    // {
    // Serial.println(data3);

```

```
// digitalWrite(LED,LOW);

//    }
data3="";

}
```

## OUTPUT:

### When object is not near to the ultrasonic sensor

sketch.ino

diagram.json

libraries.txt

Library Manager

```

1  #include <Wifi.h>//library for wifi
2  #include <PubSubClient.h>//library for MQTT
3
4
5  void callback(char* subscribetopic, byte* payload, unsigned int payloadlength);
6
7  //-----credentials of IBM Accounts-----
8
9  #define ORG "fs9trs"//IBM ORGANIZATION ID
10 #define DEVICE_TYPE "ultrasonicsensor"//Device type mentioned in ibm watson IOT Platform
11 #define DEVICE_ID "distancedetection"//Device ID mentioned in ibm watson IOT Platform
12 #define TOKEN "ALGMGaaF01nawa1QA3" //Token
13 String data3;
14 float dist;
15
16
17 //----- Customise the above values -----
18 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
19 char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and
20 char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd REPRESENT command type AND COMM
21 char authMethod[] = "use-token-auth";// authentication method
22 char token[] = TOKEN;
23 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
24
25
26 //-----
27 WifIClient wifclient; // creating the instance for wifclient
28 PubSubClient client(server, 1883, callback ,wifclient); //calling the predefined client id
29
30 int LED = 4;
31 int trig = 5;
32 int echo = 18;
33 void setup()

```

Simulation

00:05.682

99%

```

no object found
Sending payload: {"distance":403.45,"object":"No"}
Publish ok
Distancein cm233.00
no object found
Sending payload: {"distance":233.00,"object":"No"}
Publish ok

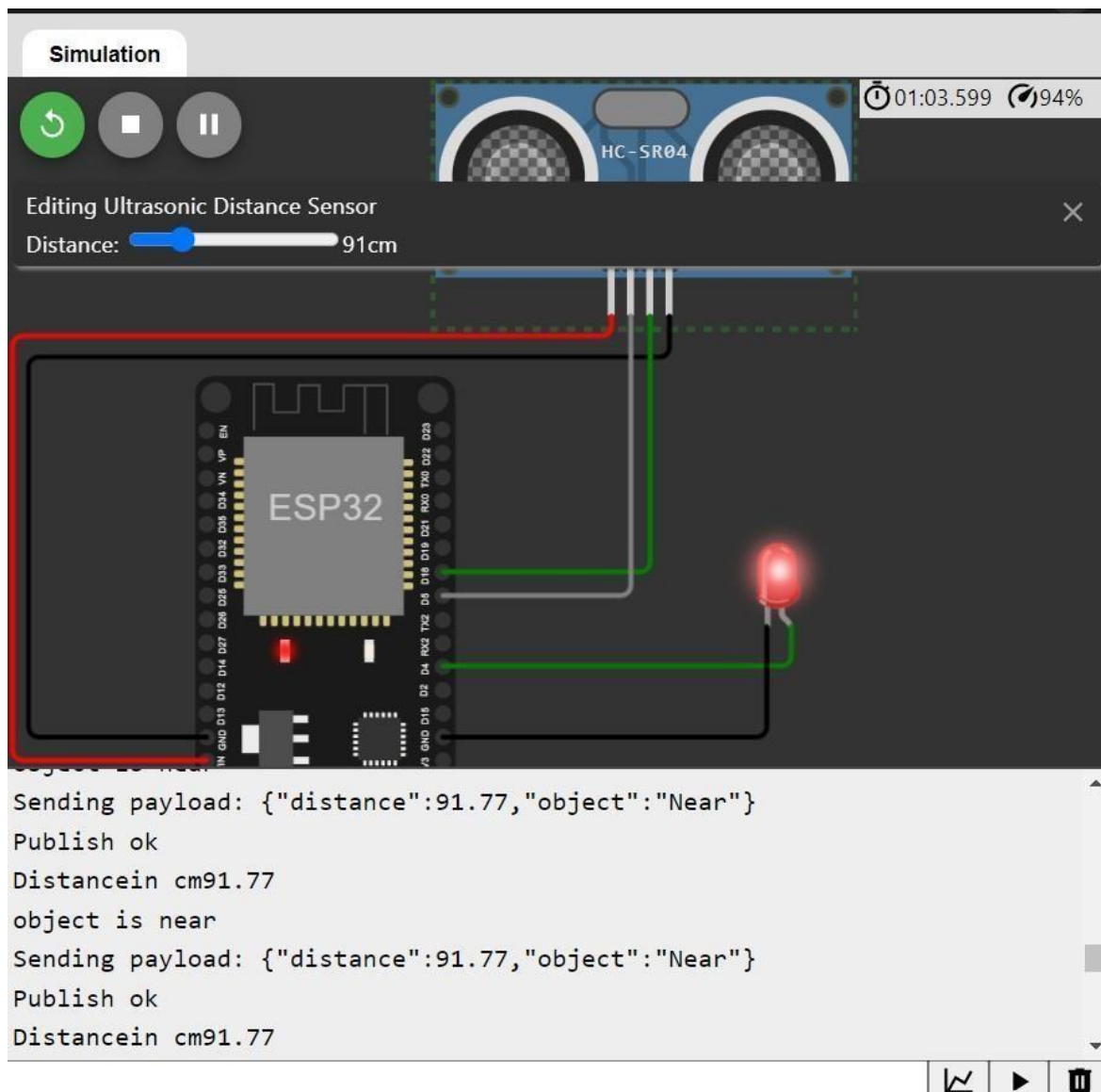
```

### Data sent to the IBM cloud device when the object is far

The screenshot displays the AWS IoT console interface for a specific device. The top navigation bar includes links for 'Browse', 'Action', 'Device Types', and 'Interfaces'. The device's name, 'distancedetection', is shown along with its status 'Connected' and the sensor type 'ultrasonicsensor'. The date and time are 'Oct 19, 2022 11:56 AM'. The 'Recent Events' tab is selected, showing a table of live data. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. All five entries are 'Data' events with a value of '{"distance":235.02,"object":"No"}' in 'json' format, received 'a few seconds ago'. The bottom status bar shows '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

## When object is nearer to the ultrasonic sensor



## Data sent to the IBM cloud device when the object is near

The screenshot displays the IBM Cloud IoT Platform console. At the top, there are tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is located below these tabs. On the right, there is a button labeled 'Add Device' with a plus icon. The main content area shows a table of devices. The first device listed is 'distancedetection', which is 'Connected' and has a 'Device Type' of 'ultrasonicsensor'. Below the device table, there is a section for 'Recent Events'. This section includes a sub-header with 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is selected, showing a message: 'The recent events listed show the live stream of data that is coming and going from this device.' Below this message is a table of events.

Event	Value	Format	Last Received
Data	{"distance":91.77,"object":"Near"}	json	a few seconds ago
Data	{"distance":91.75,"object":"Near"}	json	a few seconds ago
Data	{"distance":91.77,"object":"Near"}	json	a few seconds ago
Data	{"distance":91.79,"object":"Near"}	json	a few seconds ago
Data	{"distance":91.8,"object":"Near"}	json	a few seconds ago

At the bottom of the console, there is a status bar that reads '0 Simulations running'.



