Experiment No: 8

Experiment Name: Calibration of sensor

Objective:

1. To calibrate HC-SR04 ultrasonic sensor by interfacing it to an Arduino.
2. To display the calibrated data as a ROS message.

Theory:

The ultrasonic sensor working principle is the same as the object detection system of a bat. Also, we can say that it works on the same principle as a radar system. The ultrasonic or HC-SR04 module has two main parts, one is the Transmitter (TX) and, another one is Receiver (RX). The transmitter sends Ultrasonic (US) sound and the Receiver receives the US sound.

First of all, we need to set the Trig pin on a High State for 10 µs (microseconds) to generate the ultrasonic sound. Then it will send out an 8-cycle sonic burst which will travel at the same speed of sound. If the 8-cycle waves fall on any object and it bounces back from the object surface, then it's collected by the receiver part of the module. As a result, The ultrasonic sensor echo pin produces a high pulse output. The output pulse duration is the same as the time difference between transmitted ultrasonic bursts and the received echo signal.

$s = (v \times t)/2$

Where, s is the distance between the sensor and object. v is the speed of sound in Air, v = 0.034cm/µs or 340 m/s . t is the time sound waves take to bounce back from the object's surface. We need to divide the distance value by 2 because time will double as the waves travel and bounce back from the initial point.

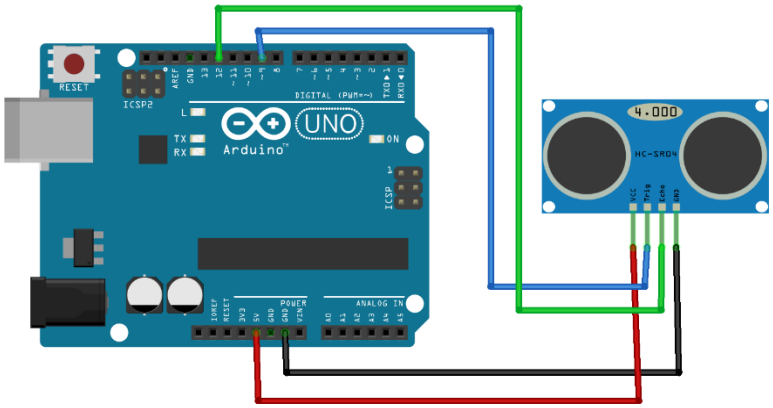
One Point Calibration

One point calibration is the simplest type of calibration. If your sensor output is already scaled to useful measurement units, a one point calibration can be used to correct for sensor offset errors in the following cases:

Only one measurement point is needed. If you have an application that only requires accurate measurement of a single level, there is no need to worry about the rest of the measurement range. The sensor is known to be linear and have the correct slope over the desired measurement range. In this case, it is only necessary to calibrate one point in the measurement range and adjust the offset if necessary.

A one point calibration can also be used as a "drift check" to detect changes in response and/or deterioration in sensor performance.

Interfacing Arduino with HC-SR04



Procedure:

- Interface ultrasonic sensor to Arduino.
- Keep an obstacle at a marked distance.
- Log the distance value recorded via serial monitor.
- Repeat the process for obstacles placed at different known distances.
- Plot a graph with actual distance measured using scale Vs the distance displayed by the serial data from the sensor.

To display the sensor data in ROS terminal

- Open a terminal in ubuntu and run roscore.
- Open a new tab, and run the code “rosrun rosserial_python serial_node.py /dev/ttyUSB0”
- From a new terminal, use “rostopic echo chatter” to display the sensor output data

Calculation:

Code:

Sample Arduino code for interfacing HC-SR04 ultrasonic sensor

This code will work only if Arduino IDE is equipped with ‘rosserial arduino’ library .

```
#include <ros.h>

#include <std_msgs/Int32.h>

ros::NodeHandle  nh;

std_msgs::Int32 str_msg;

ros::Publisher chatter("chatter", &str_msg);


#define echoPin 12 // attach pin D2 Arduino to pin Echo of HC-SR04

#define trigPin 9 //attach pin D3 Arduino to pin Trig of HC-SR04
```

```
// defines variables
long duration; // variable for the duration of sound wave travel
int distance; // variable for the distance measurement


void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate speed
  Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial Monitor
  Serial.println("with Arduino UNO R3");
  nh.initNode();
  nh.advertise(chatter);
}
void loop() {
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  // Displays the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
  str_msg.data = distance;
```

```
  chatter.publish( &str_msg );
  nh.spinOnce();
  delay(1000);
}
```

Observation:

Sl no:	Actual distance	Measured distance
1		
2		
3		
4		
5		
6		
7		

Graph:

Result: