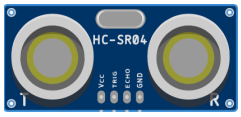
ME 220

HC-SR04 and Arduino

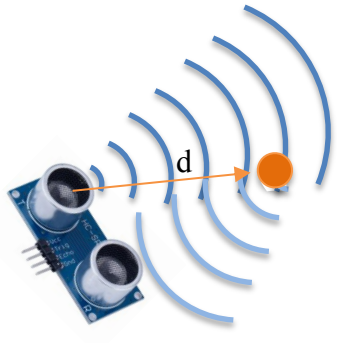
Objective:
The objective of this lab is to interface HC-SR04 to Arduino, understand how it works, characterize its response.

Pre-Lab:
Check out the datasheet of HC-SR04 which is provided along with the lab.

The operational principal of HC-SR04 (and for that matters similar time of flight sensors) is quite simple. There is a transmitter which sends out an ultrasonic pulse. The pulse propagates out and if this pulse hits an object which is located at a distance *d*, some part of this wave is reflected back, which is then senses by the receiver. Total distance travelled by the returning pulse is *2d*. Using the total time of flight (the wave traveling from the transmitter to the object, and then from the object back to the receiver) and speed of sound, distance *d* to the object can be calculated. HC-SR04 package and operating principle is illustrated in Figure 1.



a) HC-SR04 package



b) Operation principle

Figure 1. HC-SR04: an ultrasonic range finder

Note that, **ideally** the generated pulse expands out in a cone in 3D. Let's say an object is detected at a distance *d*. We cannot really tell where the object exactly is, but in 2D we can presume that it is on an arc with radius *d* as shown in Figure 1.b. Hence, the larger the *d* is, the longer the arc that the object is on. In other words, the longer the distance to the object, the less accurate is our guess about its location.

Cross-talk: In case of using multiple HC-SR04 sensors you have to be careful. The pulse of one sensor can be received by another resulting in ghost obstacles. To avoid cross-talk, cheapest solution is to make sure that only one sensor is active at a time. A more elaborate solution will be using modulated pulses. However, if there are only a few sensors and you are not in a hurry, firing one at a time is a simple, sure to work solution.

Lab:

In this lab you are expected to:

1. interface HC-SR04 to Arduino as shown in Figure 2 in Tinkercad first
2. fix the code given in Table 1, so that what the code prints on the serial port and what the sensor reads in Tinkercad are the same. Hint: Focus on: `microseconds2Millimeters()`
3. Attach a servo motor to any pin you find appropriate.

4. Modify the code so that when you move your hand in a certain range (as you find fit) back and forth towards the sensor, the RC servo rotates right and left.
5. Deploy your code to the real life.
6. Shoot a short video (< 1 minute) and submit it.

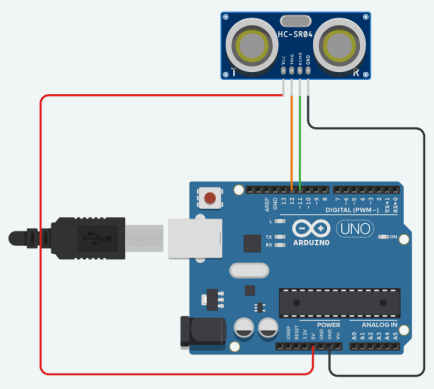


Figure 2. Interfacing HC-SR04 to Arduino

Table 1: Code to fix

```
/*
          Name, Lastname, StudentID
          Date Finished

*/

const int pingPin = 12; // Trigger Pin of Ultrasonic Sensor
const int echoPin = 11; // Echo Pin of Ultrasonic Sensor
long duration; // duration of signal


void setup() {
          // setup pin directions
          pinMode(pingPin, OUTPUT);
          pinMode(echoPin, INPUT);

          Serial.begin(115200); // Starting Serial Terminal
}

void loop() {
    duration = readUltraSonic();
    Serial.print("Total flight time: ");
    Serial.print(duration);
    Serial.print("us, total distance: ");

    Serial.print(microseconds2Millimeters(duration));
    Serial.print("mm");
    Serial.println();
    delay(250); // rest for a while
}

long readUltraSonic(){
          /*
          Reads the ultrasonic sensor
        This is a standart function where
                    a ping signal is sent through trigger pin
        and an echo is expected in return
    */
          digitalWrite(pingPin, LOW);
          delayMicroseconds(2);
          digitalWrite(pingPin, HIGH);
          delayMicroseconds(10);
          digitalWrite(pingPin, LOW);
          // listen for echo and return it, and use a time out since
          // sensor has a range of 4 meters, why is there a 25000
          // pleare read the help for pulseIn
          return pulseIn(echoPin, HIGH, 25000);
}

long microseconds2Millimeters(long microseconds) {
          /*
                    convert microsecond to centimeters:
                    speed of sound is 343 m / s * 1000 mm / 1 m
                    speed of sound is 343 * 1000 mm / s * 1 s / 1000000 us
                    speed of sound is 0.343 mm / us
                    however, use of floating point is not a good idea here
                    best long integer approximation for us to mm is:
                    note that sound travels back and forth
                    hence distance to the obstacle shoould be half the distance
                    that the sound wave traveled... note this fact.
                    to get a proper result update the following return expression
                    so that returned value is the best integer approxiamtion of
                    the sensed distance

        NOTE that you have to fix this so that the mm values returned
        matches the cm values displayed when you click on the sensor
        and move the target circle around
          */
  return microseconds * 1 / 1;
}
```