



## American International University- Bangladesh

Department of Electrical and Electronic Engineering

EEE 4103: Microprocessor and Embedded Systems Laboratory

### Title:

Interfacing the Arduino with an external sensor using serial communication protocol for implementing an obstacle detection system.

### Objectives:

The objectives of this experiment are to-

- (i) Write code for a simple obstacle detection system in Arduino IDE.
- (ii) Implement a simple obstacle detection system using an Arduino microcontroller.

### Theory and Methodology:

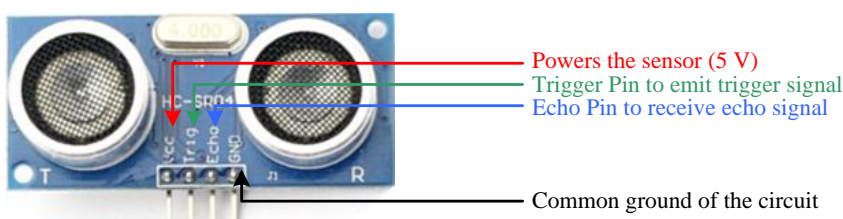
Arduino is an open-source platform used for creating interactive electronics projects. Arduino consists of both a programmable microcontroller and a piece of software, or IDE (Integrated Development Environment) that runs on your computer used to write and upload computer code to the microcontroller board. Arduino Uno also doesn't need a hardware circuit (programmer/ burner) to load a code into the board. We can easily load a code into the board just using a USB cable and the Arduino IDE (which uses an easier version of C++ to write codes).

In this experiment, we will use a sonar sensor (HCSR04) to detect the distance of an obstacle. Based on the distance between the sensor and the object being detected, one or more LEDs will glow as soon as it detects the obstacle.

The HCSR04 ultrasonic sensor uses a sonar signal to determine the distance to an object. This sensor reads from 2 cm to 400 cm (0.8 inches to 157 inches) with an accuracy of 0.3 cm (0.1 inches). The HCSR04 module consists of a transmitter, receiver, and control circuit. It has four pins, such as VCC, GND, Trigger, and Echo. A list of some features and specifications of this sensor is given below, but for more information, you should consult the sensor's datasheet:

- Power Supply: +5 V DC
- Quiescent Current: < 2 mA
- Working Current: 15 mA
- Effective Angle: < 15°
- Ranging Distance: 2 cm – 400 cm/1" – 13 ft
- Resolution: 0.3 cm
- Measuring Angle: 30°
- Trigger Input Pulse width: 10  $\mu$ s TTL pulse
- Echo Output Signal: TTL pulse proportional to the distance range

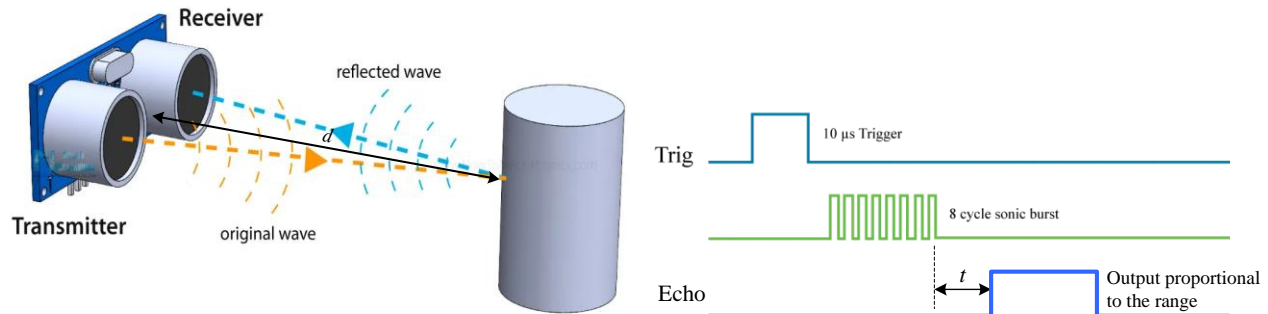
Here is the pin configuration of the sensor:



You can easily interface it with Arduino boards. Using the output Trigger pin, the module automatically sends eight 40 kHz pulse signals and detects whether there is a pulse signal back at the Echo pin. The Trigger pin of the sensor is connected to digital pin 11 and the Echo pin to digital pin 12 of the Arduino Uno R3 board with connecting wires. An LED is connected to pin 2 to show that

an obstacle is detected. Here, pins 11 and 2 will act as output pins because the trigger will be generated from Arduino, and the LED state (HIGH/LOW) will also be changed by the Arduino board. If more than one LED is to be connected then we can use more digital pins, for example, 3 and 4. All these pins must be declared as output pins in the setup function.

The ultrasound transmitter (Trigger pin) emits a high-frequency ultrasonic sound wave (40 kHz) that travels through the air. If it finds an object, it bounces back to the module. The ultrasound receiver (Echo pin) receives the reflected sound wave (echo) as shown in the following schematic diagram:

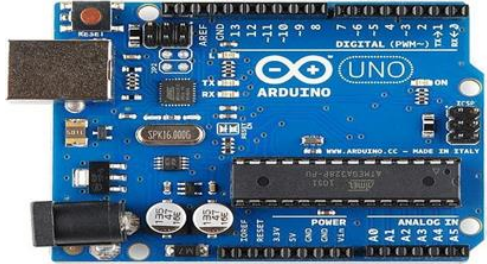




As the trigger signal generated from the Arduino board travels out from the Trigger pin and comes back to the Echo pin, the travel distance of the signal from the microcontroller to the object and back again to the microcontroller is double the distance ( $d$ ) between the microcontroller and the board. We know that the sound velocity ( $v$ ) in the air is 340 m/s.

$$2d = vt \text{ or simply, } d = \frac{t}{2} v$$

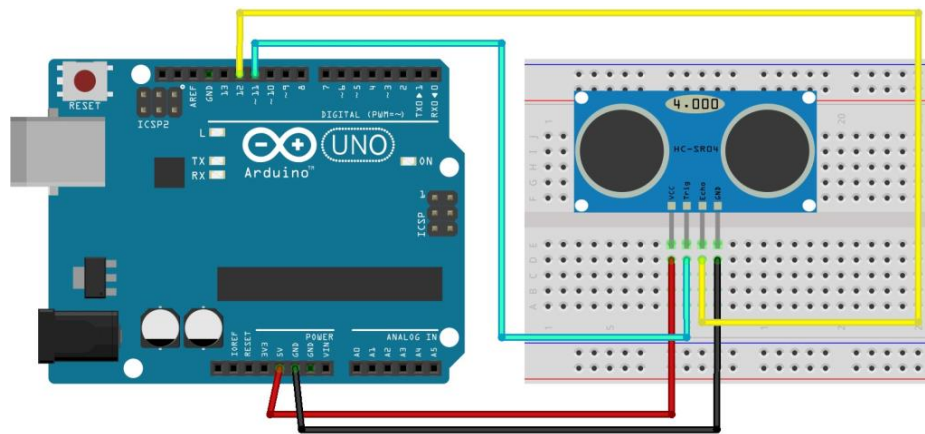
,where the  $t$  = travel time.

### Apparatus:

1) Arduino IDE (any version)	Software
2) Arduino Uno (R3) board	
3) Sonar Sensor (HCSR04)	
4) LED	

### Experimental Procedure:

Connect the circuit diagram as follows, upload the program into the board, and then run the program. Connect three LEDs of red, green, and yellow to pins 2, 3, and 4, respectively through three 100  $\Omega$  resistors to limit the current flow through LEDs (though it is not shown in the circuit diagram). Observe the object detection distances in the serial monitor and based on that LEDs turning ON.



Connection diagram

**Program:**

Write the following codes into the IDE and upload them into the Arduino board after compiling them correctly. **Please set up the Tools with the appropriate COM port before uploading the code.**

```
// define the pin numbers
const int trigPin = 11;
const int echoPin = 12;

// define variables
long duration;
int distance;
float distanceinches;
float distanceThreshold=80;

void setup() {
  Serial.begin(9600); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  pinMode(2, OUTPUT); // Sets pins 2, 3, and 4 as the Output pin
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
}

void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state 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/2) * 1e-6 * 340 * 100;
  distanceinches = (distance / 2.54);
  // Prints the distance on the Serial Monitor
```

The **pulseIn()** function reads a HIGH or a LOW pulse on a pin. It accepts as arguments the pin and the state of the pulse (either HIGH or LOW). It returns the length of the pulse in microseconds. The pulse length corresponds to the time it took to travel to the object plus the time traveled on the way back.

```

Serial.print("Distance = ");
Serial.print(distance);
Serial.print("cm; ");
Serial.print("Distance = ");
Serial.print(distanceinches);
Serial.println("inches");

// set threshold distance to activate LEDs
distanceThreshold = 80;

if (distance > distanceThreshold) {
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
}

if (distance < distanceThreshold && distance > distanceThreshold-30) {
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
}

if (distance < distanceThreshold-30 && distance > distanceThreshold-50) {
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
}

if (distance < distanceThreshold-50 && distance > distanceThreshold-70 ) {
  digitalWrite(2, HIGH);
  digitalWrite(3, HIGH);
  digitalWrite(4, HIGH);
}
delay(200); // Wait for 200 millisecond(s)
}

```

### **Questions for Report Writing:**

1. Include all codes with explanations in the laboratory report by following the writing template.
2. Perform the obstacle detection system in the TinkerCad and Proteus simulation platform and show all the results.

### **Reference(s):**

- [1] Arduino IDE, <https://www.arduino.cc/en/Main/Software> accessed on May 3, 2019.
- [2] Arduino and Proteus Library, <https://etechnophiles.com/add-simulate-ultrasonic-sensorproteus-2018-edition/> accessed on May 3, 2019.
- [3] Ultrasonic Distance Sensor in Arduino With TinkerCad <https://www.instructables.com/id/Ultrasonic-Distance-Sensor-Arduino-Tinkercad/> accessed on May 3, 2019.