IoT Platform:

Tutorial: Distance Sensor Array

Table of Contents

Introduction 3

Things Needed 3

Overview and Principle of Operation 4

Hardware Setup 6

Software and Testing 8

References 10

| Revision history | | |
| --- | --- | --- |
| **Version** | **Date** | **Comment** |
| 1.0 | 06/15/16 | Initial release |
|  |  |  |
|  |  |  |

# Introduction

In this tutorial, you will:

1. Learn to operate an ultrasonic distance sensor
2. Read data from an array of 5 ultrasonic distance sensors

# Things Needed

* An Intel Edison
* Sparkfun blocks:
  + Battery Block
  + GPIO Block
  + Base Block
* 5x HC-SR04 Ultrasonic distance sensor
* Miscellaneous
  + Solder and soldering iron
  + Wires and wire cutter
  + Electric tape
  + Female headers
  + ABS or PLA 3D printer (optional)
  + Pliers (optional)

# Overview and Principle of Operation

**Overview**

Ultrasonic distance sensors are a low cost alternative to laser rangefinders in situations where range, precision, or resolution are not important. The HC-SR04 is a popular and easily available model of ultrasonic distance sensor.

On visual inspection of the sensor the main features are:

* Two cylindrical sound transducers: Transmitter and Receiver
* Pins: VCC, Trig, Echo, GND

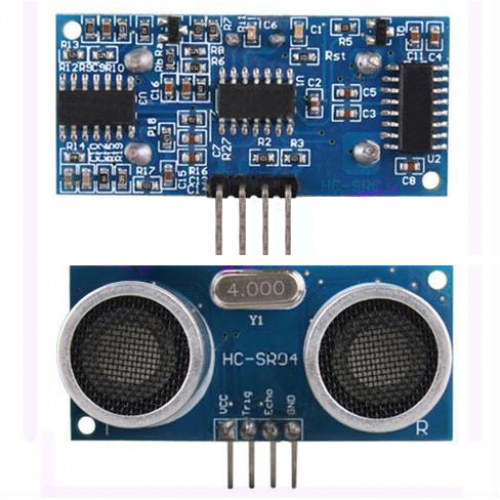
****

Figure Back and front sides of HC-SR04

**Operation**

The HC-SR04 sensor finds distance by measuring the time it takes for an ultrasonic sound wave to propagate from the sensor’s transmitter to the distant object and reflect back to the sensor’s receiver.

The use of sound to detect distance presents certain advantages and limitations. Unlike a laser rangefinder which finds the distance at the pinpoint location the laser shines on, a sound pulse from the ultrasonic sensor spreads out in a cone with an angle of about 30-40 degrees. This means that the sensor has a large “field of view” in which it can detect objects in range, but is unable to determine the position of the object within that field of view or how big it is. In addition, a laser rangefinder will work on most smooth surfaces because they scatter the laser light, but with sound, smooth surfaces will reflect the sound wave like a mirror. That means that if the ultrasonic sensor is used on a surface that is angled away from the sensor, the surface will reflect the sound away from the sensor rather than back towards it. At most, a smooth surface can face 15-20 degrees away from the sensor.

Considering these particular traits of the ultrasonic distance sensor, it is not well suited for mapping an environment, but well suited for avoiding collision with objects and walls. A front facing array of five sensors, angled 45 degrees away from each other, provides sufficient coverage for a robot to navigate corridors and avoid obstacles. Note that the sensors cannot be activated simultaneously or otherwise they will interfere with each other.

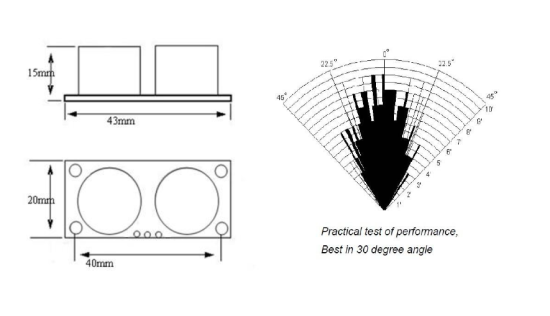


Figure HC-SR04 Sensing angle

To operate the sensor, power is applied to the VCC and GND pins. Then a 10μs pulse is sent to the Trig pin and the sensor will send and measure multiple ultrasonic pulses at 40kHz. Finally, a pulse will appear on the Echo pin and its width is proportional to distance. If there is no object in front of the ultrasonic sensor to reflect sound back, then no Echo pulse will appear. While it is recommended to power the sensor using 5V and use 5V logic on the pins, it will function if powered by a single cell lithium ion battery (3.7-4.2V) and logic voltage is the same as that of the battery.

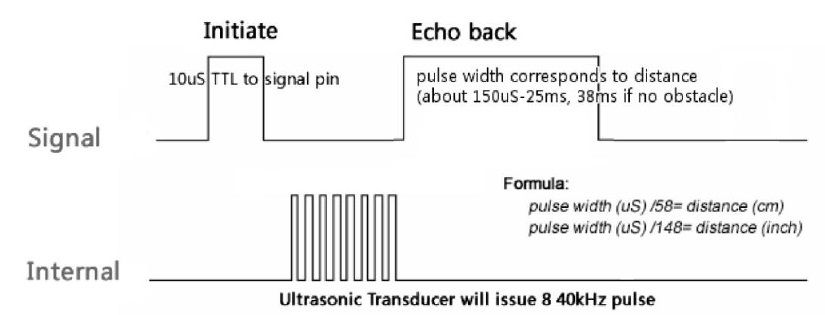


Figure HC-SR04 Operation

# Hardware Setup

**Wiring**

Refer to the following diagram for the planned wiring.

**Sensor 1**

VCC

Trig

Echo

GND

**Sensor 2**

VCC

Trig

Echo

GND

**Sensor 4**

VCC

Trig

Echo

GND

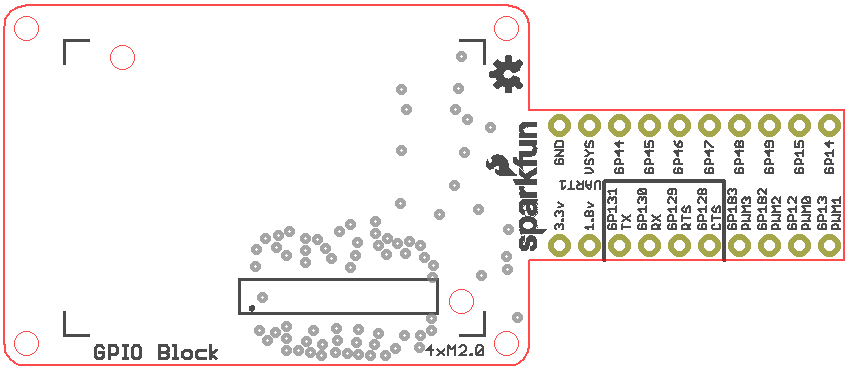
**Sensor 5**

VCC

Trig

Echo

GND



**Sensor 3**

VCC

Trig

Echo

GND

Figure Wiring diagram

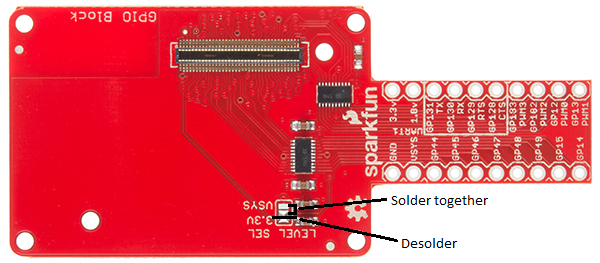
1. The Sparkfun GPIO block has a level shifter that converts between the 1.8V logic of the Intel Edison and either 3.3V or VSYS. The selection is made by a soldered jumper on the GPIO block. Re-solder the jumper to select VSYS.  
   

Figure Level Select

Recommended wiring:

1. Long wires: Cut five wires for five trigger signals, five wires for five echo signals, and two wires for VCC and ground. A good length for the wires is 4-5 inches. Making the trigger and echo wires different colors will help discern between them.
2. Short wires: Cut four 1.5 inch wire segments for VCC and four segments for ground.
3. Since ground and VCC must connect to multiple sensors, the long and short wires for VCC and ground should be combined according to the following illustration:

Figure Distributing GND or VCC to multiple wires

Twist wires together and apply solder to secure them

GND on GPIO block

Sensor 1 GND

Sensor 2 GND

Sensor 3 GND

Sensor 4 GND

Sensor 5 GND

Short wire

Long wire

1. Cut a long female header to make five 4-pin female headers. These will plug onto the ultrasonic sensors. Solder the GND, VCC, and signal wires to connect the female headers and the Sparkfun GPIO block according to the wiring diagram.
2. Plug the distance sensors into the female headers. Stack the Sparkfun GPIO block onto the battery block, the Sparkfun base block onto the GPIO block, and the Edison onto the base block.

**Mounting (Optional)**

If you plan to build the complete **pathfinding robot**, it is recommended to 3D print a mount to hold the sensors. You can find a 3D model for the mount at <https://github.com/zhgary/UCLA_EE180D_Robot/blob/master/CAD/distance%20sensor%20mount_2.stl>. In order to fit the sensors into the mount while leaving space for wiring, the angled pins of the HC-SR04 sensors must be bent straight using pliers. On the mount, the sensors should be oriented with the receiver transducer on the bottom and the transmitter on top. Sensor 1 should face the left, sensor 2 should face front left, and so on. If there is a tight fit, it is easier to snap the sensors in from the front of each slot rather than slide them in from the top.

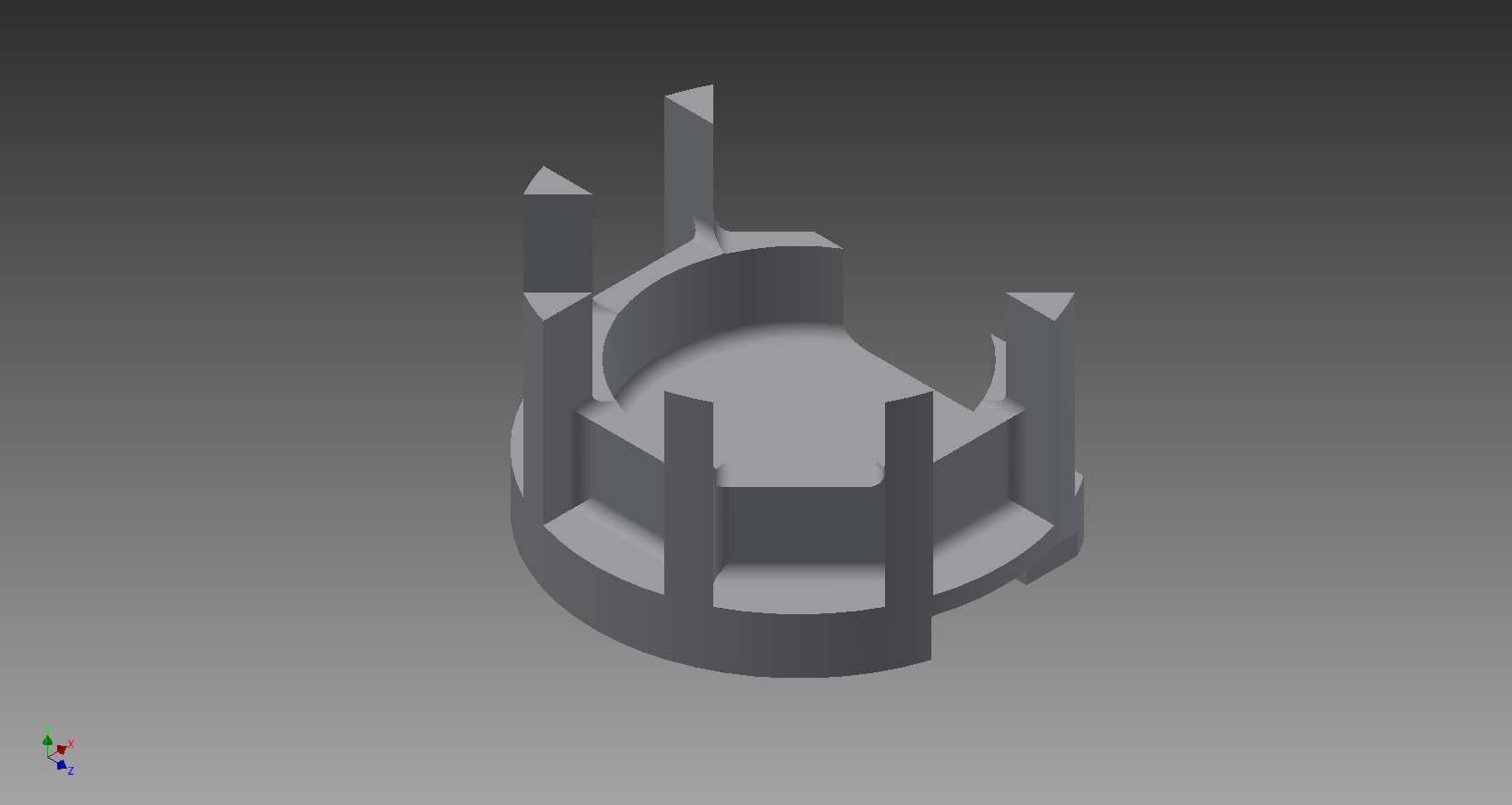
****

Figure Ultrasonic sensor mount (the front is facing towards the bottom left)

# Software and Testing

**Demonstration**

1. Create a new directory on the Edison and download the code at <https://github.com/zhgary/UCLA_EE180D_Robot/tree/master/Distance%20Sensor%20Array> into a file inside the directory.
2. Compile the code with the following command:  
   **$ gcc -lmraa test2.c –o test2**
3. Execute the code with the following command:  
   **$ ./test2**
4. The program should repeatedly output five distance values in centimeters, delimited by colons.

Software Overview

Examining the demonstration code reveals a function “get\_distance” that interfaces with the distance sensor.

First, record the time at which the trigger pulse is sent out:

gettimeofday**(&**trigTime**,** **NULL);**

Then, send out the trigger pulse (despite the specification requiring a 10 microsecond pulse, a shorter pulse will activate the sensor):

mraa\_gpio\_write**(**trigger**,** 0**);**

usleep**(**5**);**

mraa\_gpio\_write**(**trigger**,** 1**);**

usleep**(**11**);**

mraa\_gpio\_write**(**trigger**,** 0**);**

Use a loop the Echo signal for a rising edge. However, also calculate the time elapsed since the Trig signal was went; after a certain time threshold the function will return invalid data:

**while** **(**mraa\_gpio\_read**(**echo**)** **==** 0**)**

**{**

gettimeofday**(&**startTime**,** **NULL);**

**if** **(**1000000.0 **\*** **(**startTime**.**tv\_sec **-** trigTime**.**tv\_sec**)** **+** startTime**.**tv\_usec **-** trigTime**.**tv\_usec **>=** 100000.0**)**

**return** 5000.**;**

**}**

Poll for the falling edge of the echo pulse in a similar way:

**while** **(**mraa\_gpio\_read**(**echo**)** **==** 1**)**

**{**

gettimeofday**(&**endTime**,** **NULL);**

**if** **(**1000000.0 **\*** **(**endTime**.**tv\_sec **-** startTime**.**tv\_sec**)** **+** endTime**.**tv\_usec **-** startTime**.**tv\_usec **>=** 150000.0**)**

**return** 5000.**;**

**}**

Calculate the distance using elapsed time:

time\_taken **=** 1000000.0 **\*** **(**endTime**.**tv\_sec **-** startTime**.**tv\_sec**)** **+** endTime**.**tv\_usec **-** startTime**.**tv\_usec**;**

distance **=** **(**time\_taken **+** 0.00**)** **/** 58.82**;**

# References

1. https://github.com/chrisIHbaek/intel\_edison\_lego\_robot/blob/master/preloaded\_version/collision\_avoidance.c
2. https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL\_pfa39RsB-x2qR4vP8saG73rE/edit
3. https://cdn.sparkfun.com/datasheets/Dev/Edison/GPIO\_Block.zip