SOCIETY OF ROBOTS

Search Here

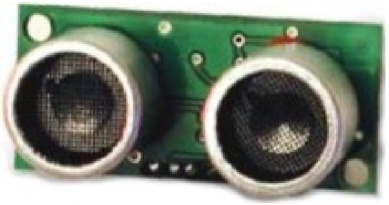
MISC
Parts List
Robot Forum
Chat
Member Pages
Axon MCU
Robot Books
Shop
Contact

SKILLS
How To Build
A Robot
Tutorial
Calculators
Mechanics
Programming
Miscellaneous
Robots
Space

HARDWARE
Actuators
Batteries
Electronics
Materials
Microcontrollers
Sensors

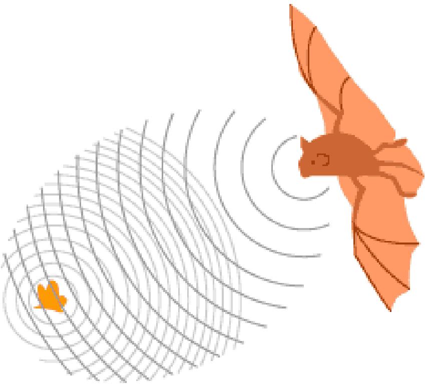
SCIENCE
Robot Journals
Robot Theory
Conferences

SENSORS - ROBOT SONAR



Sonar

Everyone knows how sonar works. A sound gets emitted, then you 'see' your surroundings based on the sound coming echoing back. This is because sound takes time to travel distances. Farther the distance, the longer it takes for the sound to come back. This sonar tutorial will talk about how to implement sonar into your robot.



How Sonar Works

Your **microcontroller** tells your sonar to go. Then your sonar emits a mostly inaudible sound, time passes, then detects the return echo. It then immediately sends a voltage signal to your microcontroller, which by keeping track of the time that passes, can calculate the distance of the object(s) detected. Here I have an example of a **robot that uses sonar**.

Availability and Cost

Robot sonar today can be bought very cheap, typically around $20-$30 each. Just check the **robot parts list** and the ad window on the right.

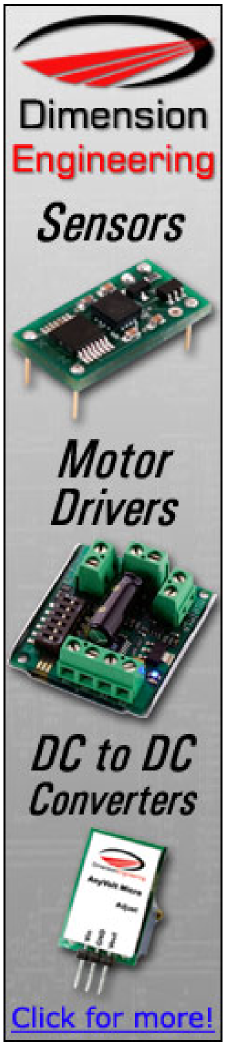
Calculating Distance vs Time

The speed of sound in air is about 343 m/s, with minor dependence on temperature and humidity. This is roughly 0.9ms/foot. The speed of sound in saltwater is about 1500 m/s, and in freshwater 1435 m/s.

example:
Suppose your robot is on land and has a sonar. The sonar sends out a sound to an object that is an unknown distance away. That means the sound has to travel this unknown distance twice (there and back). Now suppose your microcontroller says the time passed was .03 seconds. How many meters away is the object from the robot?

calculating:

Speed_of_Sound * Time_Passed / 2 = Distance_from_Object



343 m/s * .03 s / 2 = 5.145 m

The Sonar Process, English Pseudo Code

- Microcontroller sends square wave signal to control line of sonar
- Sonar detects square wave 'go' command, starts sonar process
  - Several 100us delay from 'go' command passes for electronics to work
  - Microcontroller starts a timer (to track time it takes for sound to travel)
  - Sonar emits a set of pulses at some frequency
  - Pulses stop
  - A return echo is detected (or a timeout is called if no echo)
  - Sonar immediately emits a square wave on output line to microcontroller
- Microcontroller detects square wave
- Microcontroller stops timer
- knowing elapsed time, use the equation to calculate distance

Pseudo CODE in C

```
output_high(pin_D5); //bring command pin high
delay_us(12); //wait for sonar circuitry to detect high pin
output_low(pin_D5); //bring command pin down

delay_us(210); //time it takes for a sonar to emit sound after recieving a command
set_TIMER1(0); //reset timer so no overflows
while(get_TIMER1()100); //wait until output goes low or times out

distance=get_TIMER1()/conversion_factor; //convert passed time to inches
```

Power Requirements

Typical sonar require **ground**, **power**, **signal transmit** (the 'go' command), and **signal recieve** (signals when a sound returns) lines.

The typical sonar module consumes roughly 100 mA in standby mode, meaning no sound pulses are being emitted. When in use, however, the power requirement jumps momentarily up to 2 Amps. Just like with **motors**, they can draw sudden large amounts of current resulting in sudden voltage drops on your **batteries** (i.e. a **ping** causes the microcontroller to reset). This huge jump can wreak havoc on microcontrollers and other circuitry sharing the same power supply. A microcontroller won't work and will just reset if the voltage drops too low.

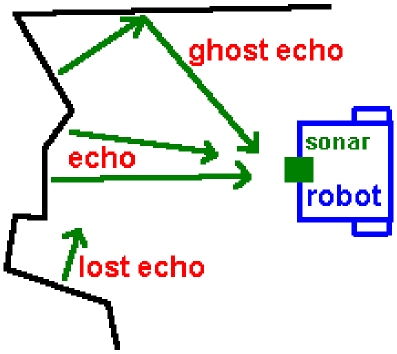
However, if you design your power supply circuit with a **capacitor**, it should suppress these sudden but short lived voltage drops. You should already have one for your motors, but if you draw from another seperate power supply, a ~500uF capacitor across the power leads should work fine.

Range (Maximum and Minimum)

Since the ping sound is spreading out radially, the signal strength as the chirp moves farther from the transducer is reduced by 1/(distance^2). This means that the maximum measuring distance drops off rapidly at the extreme maximum of the sensor. There is usually amplification electronics built in to your sonar to adjust for this, but a typical **maximum range** on a cheap sonar is still no greater than 6-25 feet.

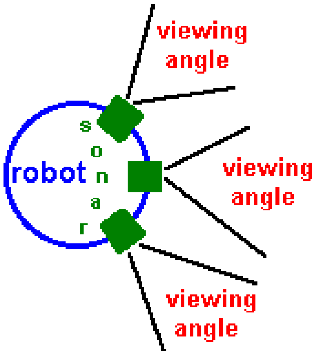
There is also a **minimum range**, meaning that if an object is too close (say within a inch or two) from the sonar emitter, your sonar will not detect the object (or at least not accurately). This is because sound is really fast, so fast that the electronics cannot work with in the time it takes for the sound to return back to the sensor. Make sure your sonar is an inch or more back from the front of your robot, or it will not detect wall collisions properly.

Sound Reflectance / Absorbance and Object Material Properties
Unfortunately, echos are not completely a product of distance. There are many other factors that can alter readings. The sound reflected from a pillow and from a solid wall will not be the same. If the object is at a sharp angle, much less sound would be reflected back. Surface properties of the objects can also make a difference. A carpet and a mirror would give different readings. Your sound can also 'get lost,' bounce around various walls for some extended period of time, then return to your sonar as a '**ghost echo**' - or even worse, it might cause false triggering making your robot 'see' objects that arent really there.



As an object sensor, you can use some of these 'problems' to your advantage. What if your blind robot needed to know if it is trying to kill humans with a brick or a pillow? The uses are vast.

Using Multiple Sonar Simulataneously
Now suppose your robot had multiple sonar sensors on it. How would you prevent one sonar from not detecting an echo caused by another sonar sensor? The hardware approach would be to point your sonar at different angles (outside of the viewing angle of the type of sonar you are using). Typically this would be around ~20 degrees.



The electronics approach to solving this problem would be to have each sonar operating at a different sound frequency (not easy to do!).

Lastly, there is the computational approach. With this method each of your sonar would fire at different times. Your robot would wait for one sonar to receive an echo, plus an additional time for ghost echos to dissipate, before the next sonar is activated. But if you have say 16+ sonar, that can take a huge amount of wasted time! Then you can do things like have only sonar on opposite sides of the robot fire together. Or do some strange sonar firing pattern so that each sonar has a small chance of interfering with each another.

Advanced Signal Interpretation

The sonar data can be processed in the same was as a **SharpIR Rangefinder**. I recommend you reading that tutorial to help you get higher quality data from your sonar. There is also a tutorial on helping you **interpret sensor data** if you wish to increase your sensor accuracy.

Has this site helped you with your robot? Give us credit - link back, and help others in the forums!