ME 3200 Mechatronics Laboratory

Lab Exercise 7: Ultrasonic Sensors

Introduction

The objective of this lab is to provide you with the experience of using an ultrasonic sensor. Ultrasonic sensors, or sonar sensors, are used in a wide range of applications involving range finding and object detection and avoidance. For example, the Polaroid Corporation uses sonar sensors in their auto-focusing cameras to detect the distance to an object for focusing purposes. Sonar sensors are excellent for mobile robot applications; especially if a robot needs to navigate through an area filled with obstacles.

How Sonar Sensors Work

Figure 1 below shows a simple schematic of a generic sonar sensor in action. An ultrasonic sound burst, or *ping*, of a given frequency is emitted from the transducer, the sound waves travel until reflected off of an object, and the echoed sound waves return to the transducer. The echoed wave may have a smaller amplitude than the ping, but the frequency should be the same. An external timer records the time-of-flight (the time that the sound waves take to travel to and from the object), which can be converted to a distance measurement by considering the speed of sound in air. As the transmitted sound waves propagate from the transducer, they expand in the shape of a cone of angle θ as depicted in Figure 1 below. This cone defines the area over which the sonar is effective.

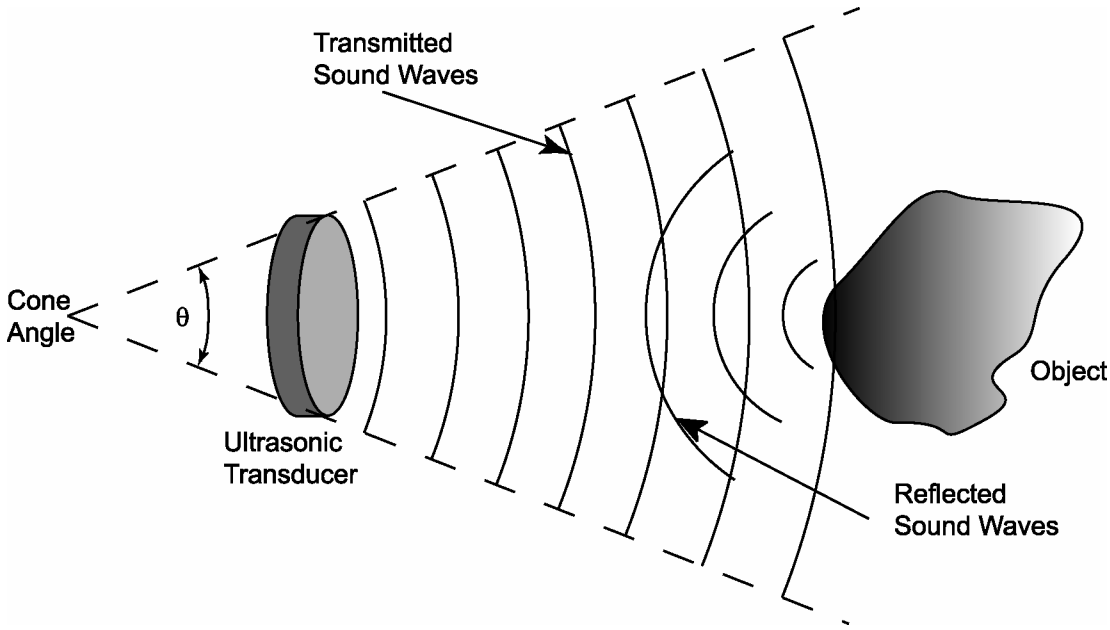

Figure 1: Functional diagram of a generic sonar sensor.

Limitations of Sonar Sensors

Sonar sensors are not ideal devices. They are limited in resolution, range, and the size of object they can detect. In addition, the external timing circuits of some sonar sensor systems may detect to false echoes. The time-of-flight values returned by the sensor may not correspond to the actual distance to the object. One solution to this problem is to average several sonar time-of-flight readings and disregard unreasonable time-of-flight readings (like negative numbers). This method can greatly decrease the effects of false echoes on sonar measurements.

The Devantech SRF04 Ultrasonic Range Finder

The sonar used in the lab and on the project is the Devantech SRF04 Ultrasonic Range Finder. An important feature that distinguishes itself from the generic sonar sensor in Figure 1 is that it has a separate transmitter and receiver. This allows for a smaller minimum detectable distance than sonar sensors that rely on a single transducer to act as both a transmitter and a receiver. The Devantech sonar produces a 40kHz ping—roughly twice the highest frequency detectable by the human ear. This frequency is used to reduce the chance of false echoes returning from other sources in the environment since it is unlikely that a 40 kHz sound wave will come from any other source than the actual ultrasonic echo. The external timing circuit starts timing when the ping is generated and listens for the arrival of a 40 kHz echo to stop the timer. An advantage of the SRF04 (over the Polaroid units) is that the frequency generating circuit is integrated with the printed circuit board that supports the ultrasonic transducers. This minimizes the amount of wiring needed to interface the sensor with a control platform. For more information on the SRF04 sonar sensor, follow the links on the class web page associated with this lab exercise or visit the SRF04 distributor, Acroname, at www.acroname.com. Acroname also sells many other useful robotics related devices. Following are direct links to the SRF04 information pages:

- http://www.acroname.com/robotics/parts/R93-SRF04.html
- http://www.acroname.com/robotics/info/examples/srf04-3/srf04-3.html

Interfacing the SRF04 with the Handy Board

In order to interface the SRF04 with the Handy Board, the user must wire the sonar to the Handy Board and download a sonar program to the Handy Board. The wiring diagram for the SRF04—Handy Board circuit is shown in Figure 2 below. The red wire is connected to the regulated five volt power supply bus of the Handy Board, the black wire is connected to the ground bus, the green wire is connected to port nine of the digital input bus, and the yellow wire is connected to port seven of the digital input bus. After making these connections, a sonar program must be loaded onto the Handy Board. The program used in lab is `devantech_sonar.c` ; the example on the Acroname website (second hyperlink above) may also be used. Once this program is loaded, the Handy Board must be initialized to work with the sonar unit by calling the function `sonar_init()`—a sub-function of the `devantech_sonar.c` program. This function changes the digital nine input into a digital output and initializes the sonar unit. The remaining code in the `devantech_sonar.c` program initiates a ping and measures the time since the ping was sent till the echo returns. This time is returned as an integer equivalent to the number of CPU clock cycles (or ticks) of the Handy Board's microprocessor. Upon completion of these steps the Handy Board is ready to use the SRF04 range finder.

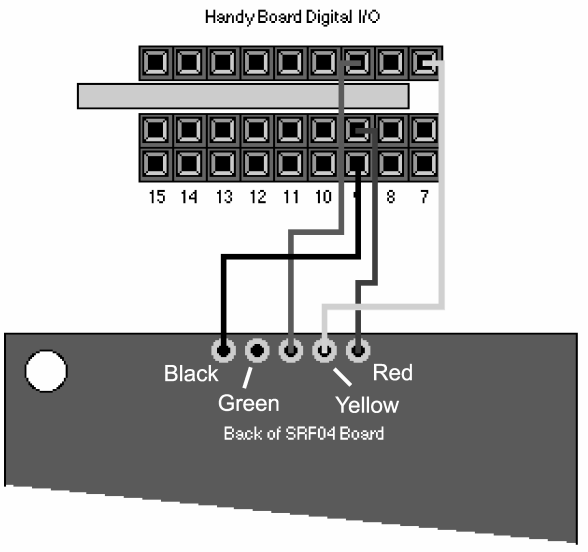

Figure 2: Wiring diagram for the SRF04—Handy Board interface.
Figure used with permission from Acroname
(http://www.acroname.com/robotics/info/examples/srf04-3/srf04-3.html)

Laboratory Exercise

Equipment Needed:

- A CutieBot equipped with a Devantech SRF04 Ultrasonic Range Finder
- An RJ11 Cable
- A 12 Volt, 500mA DC transformer.
- A Yardstick

1. Interface the Handy Board of the CutieBot with the computer using the procedure described in the *Preparing the Handy Board for Use* section of the *Introduction to the Handy Board* lab.

2. Open a web browser and download the Interactive C program called `devantech_sonar.c` from the Lab Handouts page below the section referring to Ultrasonic Sensors; save this file to your disk drive.

3. Open the `devantech_sonar.c` in Interactive C and download it to the Handy Board

4. With the Handy Board in interactive mode, initialize the CutieBot's sonar by executing the `sonar_init();` command at the Interactive C command prompt.

5. Place an object about ten inches in front of the CutieBot, type "`printf("Ticks=%d\n",sonar());`" at the command prompt several times, and observe how the sensor measurement varies each time the command is executed. Using the up arrow at the command prompt allows you to access previous commands.

6. Repeat step 5, but this time, move the object further away from the CutieBot. Notice that the returned integer is proportional to the distance to the nearest object as you move the object around.

7. Use the sample code provided in Figure 3 to write a program that averages four time-of-flight readings. Save the program for your records.

```
int i;                          /* a loop counter */
int num;                        /* the number of readings to average */
int ticks;                      /* the # of clock ticks returned for one reading */
int ticksAvg;                   /* the average of num readings */

void main() {
   num = 4;                     /* set the number of readings to average */
   sonar_init();                /* initialize the sonar function */
   printf("press start\n");
   while(1) {                   /* loop continually */
      if(start_button()) {      /* take num readings every time start is pushed */
         ticksAvg = 0;          /* initialize to 0 */
         for(i=0; i<=num;){     /* take num readings */
            ticks = sonar() / num; /* dividing before summing avoids overflow error*/
            if(ticks > 0) {     /* discard negative values from sonar() */
             i += 1;
             ticksAvg += ticks;    /* sum ticks to get average */
            }
         }
         printf("ticks = %d\n",ticksAvg);      /* print the result */
      }
   }
}
```

Figure 3: Sample code needed to measure and average the time-of-flight data from the Devantech sonar.

8. Create a new file in Interactive C that contains the following text:

```
devantech_sonar.c
***.c     // this is the file you created in step 7.
```

Save this file with a `.lis` file extension and download it to your Handy Board.

Note: Files with a `.lis` file extension are "list" files. They direct the compiler of Interactive C to compile and load all the files listed simultaneously. Therefore, the `.lis` file above loads both the `devantech_sonar.c` and the code you created in step 7.

9. Using the program in Figure 3 and a yardstick, determine the calibration coefficient for your CutieBot's sonar that relates the number of ticks to a distance in centimeters.

   (a) Place the CutieBot on the floor and put the yardstick on the floor orthogonal to the sonar sensor so that you can measure the distance from the sonar transducers to a point on the yardstick.

   (b) Place a large object with a flat, vertical face (such as a book standing on end) at several positions along the yardstick, run the program you loaded in step five, and record the distance and time-of-flight data in an Excel spreadsheet.

   Note: The wider the range of distances you measure coupled with the spacing between measurements will influence the accuracy of your calibration coefficient. A distance of about 200 centimeters at 5 centimeter increments is recommended.

   (c) Plot these data in Excel or Matlab and determine a calibration coefficient that relates clock ticks with distance using linear regression (use an x-y chart with a trend line in EXCEL or the polyfit command in MATLAB).

   (d) Save your data and the linear regression results and record your calibration coefficient below:

   Calibration coefficient: _____ (cm/clock ticks)

10. Modify your program to convert the average time-of-flight measurements into a distance in centimeters and display the final result on the LCD screen. Use the following steps.

    (a) Modify the code of Figure 3 so that it is a subroutine of a different main program. Refer to the sample code below for help.

```
int yourname() {                    /* "yourname()" is the name of the subroutine */
   int num = 4;                     /* set the number of readings to average */
   sonar_init();                    /* initialize the sonar function */
   ticksAvg = 0;                    /* initialize to 0 */
   for(i=0;i<num;){                 /* take num readings */
      ticks = sonar() / num;        /* dividing before summing avoids overflow error*/
      if(ticks > 0) {               /* discard negative values from sonar() */
      i += 1;
      ticksAvg += ticks;            /* sum ticks to get average */
      }
   }
   return ticksAvg;                 /* specifies the returned variable*/
}
```

Figure 4: Subroutine sample code.

(b) Write a new main function in the same C file that calls your subroutine, performs the conversion from clock ticks to centimeters, and displays the result as a floating point variable on the LCD screen. The following code may be useful:

```
# define m ##.##                  /* Enter your calibration coefficient here */
# define b ##.##                  /* Intercept from your regression */

void main() {
   float cm;
   int x;

   printf("press start\n");
   while(1) {
      if(start_button()) {
         x  = yourname();         /*Call the subroutine*/
         cm = ((float)x)*m + b;
      }
   printf("Distance is %f cm\n", cm);
   }
}
```

Figure 5: New main function that converts the number of clock ticks to a distance in centimeters.

Notice that this sample code allows you to take multiple measurements without restarting the Handy Board.

(c) Save a copy of your code to use in your report.

11. Using the program you just created and the set-up from step six, determine the maximum distance the sonar is capable of detecting accurately by placing an object with a large surface area at a distance away from the CutieBot and comparing the measurement from your program with the yardstick; keep increasing the distance between the two until the measurements from the CutieBot are unreasonable. Record the distance below.

    Maximum Distance _____ (cm)

12. Determine the minimum distance the sensor is capable of measuring using a similar procedure; record the distance below.

    Minimum Distance: _____ (cm)

13. Determine the cone angle of the sonar sensor.

    (a) Place your CutieBot on the floor again and align the midline of the sonar with one of the seams of the tiles on the floor.

    (b) Place an object a known distance from the CutieBot and slide it orthogonal to the floor seam you're using while you take distance measurements with the CutieBot. Keep moving the object from the midline until the robot can no longer "see" the object.

    (c) Measure the distance between the closest point of the object and the midline with the yardstick and record the data in an Excel file.

    (d) Repeat steps ten (b) and (c) for both sides of the midline until you can create a series of points like Figure 4 below (approximately nine to twelve points per side of the midline since the cone is not always symmetrical).

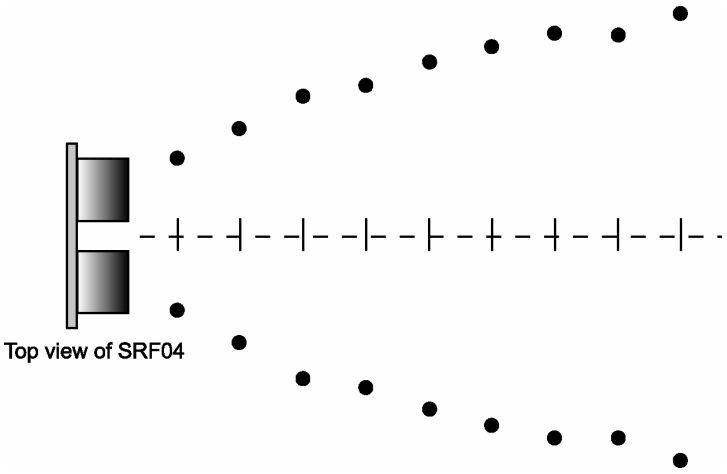

Figure 6: Measuring the cone angle of the SRF04

(e) Plot the data you measured in Excel or Matlab with measurements from one half of the midline positive and the other negative. Use linear regression to find the slope of the best-fit line for each half of the cone.

(f) Use trigonometry and the slopes calculated in the previous step to determine the cone angle of the sensor. Save your data and record the cone angle below.

Cone Angle: _____ (degrees)

14. Write a program that utilizes background processes to make a CutieBot locate and approach an object subject to the following requirements:

- The object will be placed within a one meter radius of the CutieBot.
- The CutieBot must be placed with the sonar facing away from.
- When the CutieBot has found the object, it must approach it and stop approximately 10-cm away from the object.

There are many ways to accomplish this task. Carefully consider what you know about the sonar and the way the Handy Board handles background processes; then make programming decisions that reflect your understanding. You might consider altering the `irFinder.c` program you created last lab to help you in this task. Think about how the motor speed of the CutieBot will affect the accuracy of the sonar reading. In addition, determine how the CutieBot should react if the object moves out of the cone angle of the sonar.

Questions:

1. Based on your calibration coefficient, determine how far the ping travels in a single clock tick? What is the actual processor frequency of the Handy Board of your CutieBot?

   Hint: treat the clock tick as a period, T, where the frequency of the CPU, f, is f = 1 / T. Note that the speed of sound in air is 343 m/s in air at 20° C.

2. Can the sonar module determine the exact location of an object in a 2-D space?

   Can the sonar sensor determine the size of an object?

   How might you use the sensor to determine the size of an object?

3. Based on your measurements, what are the specifications of the sonar you tested? Compare your results with the data supplied by Acroname by following the "Sonar Data" link on the lab handout page of the class webpage. List them in the space provided and discuss why there may be differences.

	Max. Distance	**Min. Distance**	**Cone Angle**
Your Measurements			
Acroname			

4. Can the sonar detect quickly moving objects? Why or why not?

5. Describe the strategy that you used to home in on the object with the CutieBot.