RichardVannoy.info

Lynxmotion.com 4-Wheel Drive Rover Kit

Part 3; Sensors

- Part 1: The First Steps - Assembling the documentation and deciding on the robot capabilities.
- Part 2: The Mechanics - Assembling the Rover Body
- Part 3: The Sensors - Mounting and Testing the Sensors (This page)
- Part 4: The Source Code - Implementing the Main Algorithm

Student Participation

I completed the previous sections myself. I am presently teaching an "Embedded Systems" class, so I decided to task the four teams in that class to independently research and program the different sensors. I'll post the results here soon.

Order of Operations

As a sequential person, I like to attack or solve one challenge at a time, so I will approach the sensors in order. Somewhat in the order of

difficulty, easiest first, I will work on the following:

1. Servo Rotation (Parallax Standard Servo by Futaba)
2. Infrared (IR) Detectors, front and rear (Sharp GP2D12 Distance Sensors)
3. Ping Acoustic Sensor
4. Sabertooth Motor Controller
5. Sound
6. Front and Rear Bumpers

I haven't decided exactly how to design and mount the front bumper, so I'll save that for later.

Servo Rotation

The servo used is the Parallax Standard Servo by Futaba. As always, my first task is to find everything I can about it on the net. Here are the references I found most helpful:

- This article by Tod E Kurt should go FIRST! It explains the basics of servo control in a clear concise manner for any beginner to servos. I skipped the sections about mountings, linkages and horns because they don't apply for this project, but the rest of the article is great. Very important are the two short programs at the end showing the bare bones code with and without using the servo.h library.
- The Arduino and Servos Article by Robotgrrl shows how to run two servos at once, which we don't need, but it has value in that it confirms the sequence for initialization in the previous article and it provides a different viewpoint for the program. Notice the formula

```
pulseWidth1 = (myAngle1 * 11) + 500;
```

  which converts the desired angle to the microseconds needed to get that angle.
- Now that we have the basics, it is time to get the data sheet for this servo so we can know the specifics of the hardware we are dealing with. The Parallax Standard Servo (#900-00005) data sheet is available on the Arduino web site. It confirms some that we have already seen and also provides a number of important specifics, like...
  Holds any position between 0 and 180 degrees
  Power Requirements: 4 to 6 VDC

Communication: Pulse Width Modulation (PWM)
Wiring was verified to be: Red = +Servo power supply, Black = Ground, White = Signal or PWM
The Basic Stamp code samples were not helpful for us.

- The next article, called Controlling a Parallax (Futaba) Continuous Rotation Servo with Arduino. should not be part of this project. A beginner, coming on this article might be seriously handicapped or confused if they do not understand that a continuous rotation servo operates completely from the standard 180 degree servo. I only include it here as a warning to ignore any info on continuous rotation servos for this project.
- Servo Motor Control with an Arduino is an excellent article showing what looks like two great sample sketches for the Arduino. One uses the millis() function to keep track of when the servo was last pulsed, amd another showing the use of the Servo Library. This is a must read article.
- By now, you should be getting the idea that a servo being run by the sketch is never free to go off and do other things for very long (more that 20 miliseconds). From the readings so far, you may have deduced that the Servo Library somehow runs the servo in a way that lets the sketch go off and do other tasks instead of staying married to the servo all the time. This article called Servo Library gives a clear description of the Servo Library and how to use it.
- And finally, I found these three sketches written by Tod E. Kurt of todbot.com for more examples of good servo code:
    1. servo_move_simple.pde
    2. servo_serial_simple.pde
    3. servo_serial_better.pde

At this point, I collected all of these documents and presented them to my class. I asked each of the four teams to have a working servo motor that steps from 10 to 170 degrees in steps by next week. I'll let you know how they did.

Sound

The first, and easiest, thing to set up is sound. I decided that I want the robot to produce sound signals to let me, and the viewers know what the robot is doing, so I need different audio signals to indicate things like:

- The robot sent out an acoustic pulse from the Ping Sensor.

- The robot has detected an object near by.
- The robot has hit an obstacle.
- Ans so on...

Anyone who builds robots or writes computer programs knows that sometimes the robot or the program seems to do something totally unexpected. That's because a computer program does WHAT YOU TELL IT TO DO, not WHAT YOU WANT IT TO DO! Often you think you told it do do one thing, but you actually told it to do something else. A robot or computer program that tells you what it is doing as you go along will be much easier to troubleshoot when something doesn't go the way you think it should.

I wrote a sound test program to play a number of tones. This was primarily to make sure the microcontroller is working. For the BasicATOM Pro-28, Pin 9 is permanently connected to an on-board piezo speaker, so that can be used with no additional wiring or hardware.

Here is the code I wrote to test the piezo speaker:

```
'  File Name: sound.bas
'Description: A program to test the piezo speaker on the
'             BasicATOM Pro 28. On this board, Pin 9 is
'             hardwired to the speaker, so this requires
'             no wiring or components.
'     Author: Richard T. Vannoy II, M.S.I.T., B.S.E.E.T.
'     E-Mail: RoboticsProfessor@ GMail.com
'   Web Site: http://www.RichardVannoy.info
'  More Info: For more info on this project, see
'             http://www.RichardVannoy.info/rover.php
'       Date: 3 March, 2011
'  Revisions: None


'*************
'* Variables *
'*************
freq var word ; The frequency to be sent to the speaker.
'               Word is used to allow numbers up to 65,535.

'*************
'* Main Loop *
```

```
'*************

for freq = 500 to 10000 step 1000
  debug ["Sending: ",dec freq, 13]
  sound P9, [1000\freq] ;step through the frequencies
  '               freq = The frequency sent to the speaker
  '          1000 = 1000 milliseconds = one second
  '     P9 = Pin 9, the output to the speaker
  pause 200     ' Pause briefly after each note
next

end
; You will notice that some of the tones will sound very
; clear and loud. Other frequencies will sound with
; lower intensity or will sound "fuzzy" and not produce
; a pure tone. This is one of the problems with trying
; to use PWM output to produce tones. PWM is actually
; a square wave. A square wave is the sum of that
; frequency and all of its odd harmonics, so a PWM
; signal will often have several audible harmonic
; frequencies present. This distorts and fuzzes the sound.
```

Later, I can fool with notes, tunes or bugle calls to make the robot sounds more interesting, but for now, I just need some sounds I can use when the robot does something. This helps me see what the robot is doing during the testing phases of construction.

Sharp GP2D12 IR Sensor

The Sharp GP2D12 is an inexpensive, simple, accurate sensor that determines the distance to objects from 10 to 80 centimeters. The sensor is constantly reading when powered up, so you just need to read the pin it is connected to. The command

```
adin P16,scanrange
```

in the program below reads the sensor and places the analog reading in the variable 'scanrange'.

The Voltage vs. distance curve for the GP2D12 is not linear, so a simple math formula to convert input volts to distance will not be accurate. The data table below takes this curve into account and provides the correct centimeters based on the input voltage.

The

```
sound P9, [400\val*50 + 2000]
```

command below converts the 10-80 centemeters reading to 2500 Hz-6000 Hz to provide a different tone for each distance. As the target gets farther away, the frequency of the tone increases. This helps to verify that the readings are being correctly read and converted.

Here is my code to test the IR Sensor:

```
'  File Name: Sharp GP2d12 Test.bas
'Description: A program to test the Sharp GP2d12 Infrared
'             (IR) Sensor on the BasicATOM Pro 28.
'     Author: Richard T. Vannoy II, M.S.I.T., B.S.E.E.T.
'     E-Mail: RoboticsProfessor@ GMail.com
'   Web Site: http://www.RichardVannoy.info
'  More Info: For more info on this project, see
'             http://www.RichardVannoy.info/rover.php
'       Date: 5 March, 2011
'  Revisions: None
'
' Hardware Connections on BasicATOM Pro-28:
'
'Sound Output
'-------------
'     P9
'
' IR Inputs:
' ----------------------
' P16 = Mounted on Servo
' P17 = Looking Aft (Not tested in this program.)
'

'-----[ Variables ]-----------------------------------
---- scanrange var word ' A/D result variable floating
var float ' Floating point math result storage val var
byte ' Table conversion result storage ctr var word '
General loop counter '----[ Table Setup ] ------------
------------------------ scantable bytetable
80,80,80,80,80,80,80,80,80,78,|
76,74,72,70,68,66,64,62,60,59,|
58,57,55,53,52,51,50,49,48,47,|
```

```
45,43,42,41,40,39,38,37,35,33,|
32,31,30,30,29,29,28,28,27,27,|
26,26,26,25,25,25,24,24,24,23,|
23,22,22,21,21,20,20,20,19,19,|
18,18,18,17,17,16,16,16,15,15,|
15,14,14,13,13,13,12,12,11,11,|
11,10,10,10,10,10,10,10,10,10 '----[ Main Loop ] -----
---------------------------------- main adin
P16,scanrange ' Read sensor value if scanrange > 512
then ' Test for obstacle to close to detector debug
["Too Close To Measure", 13] ' Send "Too Close"
message else floating = tofloat( scanrange) / 5.12 '
Limit value to < 200 values val = scantable(toint
floating) ' Convert A/D to measurement ' Send results
to debug window debug ["Scanrange = ",dec scanrange,"
Floating: ",real floating," Converted = ",DEC val,13]
sound P9, [400\val*50 + 2000] endif pause 500 goto
main ' Loop back and get another reading
```

Please email me at RoboticsProfessor@gmail.com if you have any questions or comments.