University of Florida
Department of Electrical and Computer Engineering
EEL 5666
Intelligent Machines Design Laboratory
Final Report

SPARtan

Bryan Arkins

TA: Louis Brandy
Max Koessick
William Dubel
Instructor: A. A. Arroyo

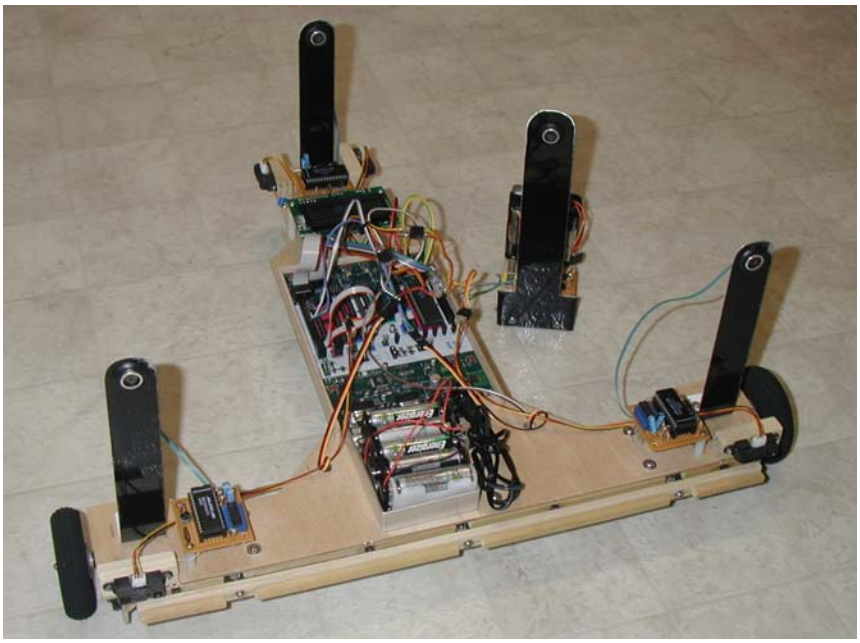
Table of Contents

Abstract

SPARtan is a Sonar Positioning Autonomous Robot. I chose this concept because it really hadn't been done before and requires a lot of mathematical concepts. I like a challenge and enjoy mathematical applications thoroughly. It is my goal to build an interesting and original robot and see how feasible this sonar technique will really be. I think working with sonar is interesting and attempting a unique concept is enjoyable. Because this type of sonar system is not readily available to purchase, the sonar would be built by me. Building such a sensor proves to be a huge challenge and gathering the right information is crucial.

Executive Summary

SPARtan attempts to follow a sonar transmitter providing a 40 kHz pulse every 1.5 ms. This transmitter is placed a fair distance across the room from SPARtan. Between the two are some obstacles in which it must try to avoid. Once the transmitter and SPARtan are set up in the room far from each other with obstacles in between, both the transmitter and SPARtan are turned on. At this point, SPARtan goes through its initialization process and waits for the appropriate switch to be pressed. Now SPARtan goes through his behaviors. The first is sonar position where it attempts to line up with the transmitter. The second behavior is to avoid obstacles. SPARtan uses IR sensors to detect where obstacles lie. The final behavior is to display the current status of SPARtan on the LCD. As a last resort, SPARtan has a bump switch panel on the front to allow him to stop when it has been hit. Once this occurs, SPARtan is then done.

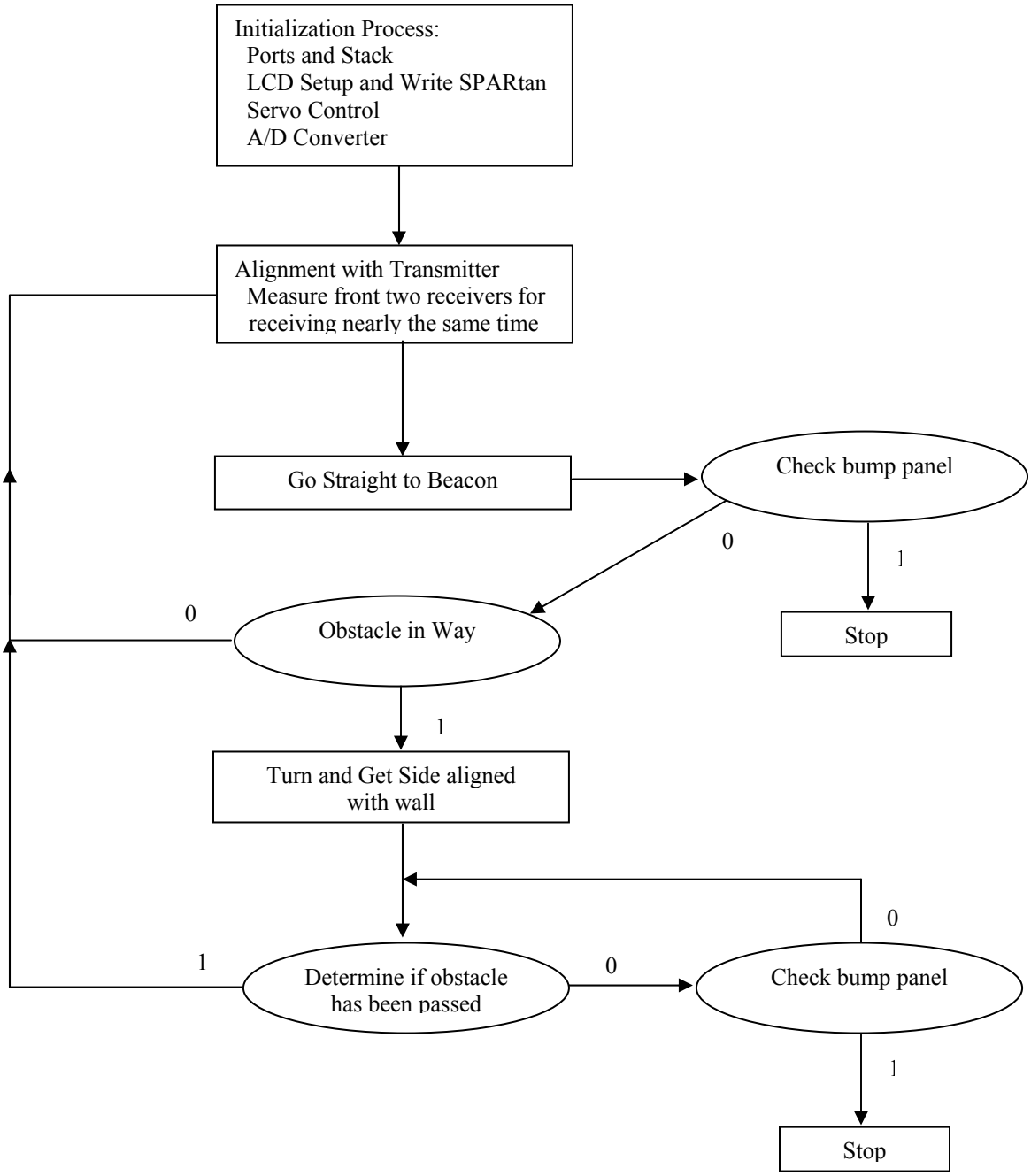


Introduction

The concept of my robot is one that deals with positioning. I wanted to choose a unique way of finding one's way to some place. Sonar has always seemed to intrigue me so this is what I have chosen. I plan to use 3 sonar receivers and 1 sonar transmitter which will allow the robot to triangulate its position according to a beacon (sonar transmitter). I find this interesting because I like the mission that my robot must perform. Its concept is it starts out lost and tries to find home while avoiding the obstacles that remain in its way. Throughout this paper, I will discuss SPARtan thought process and what things make him work the way he does.

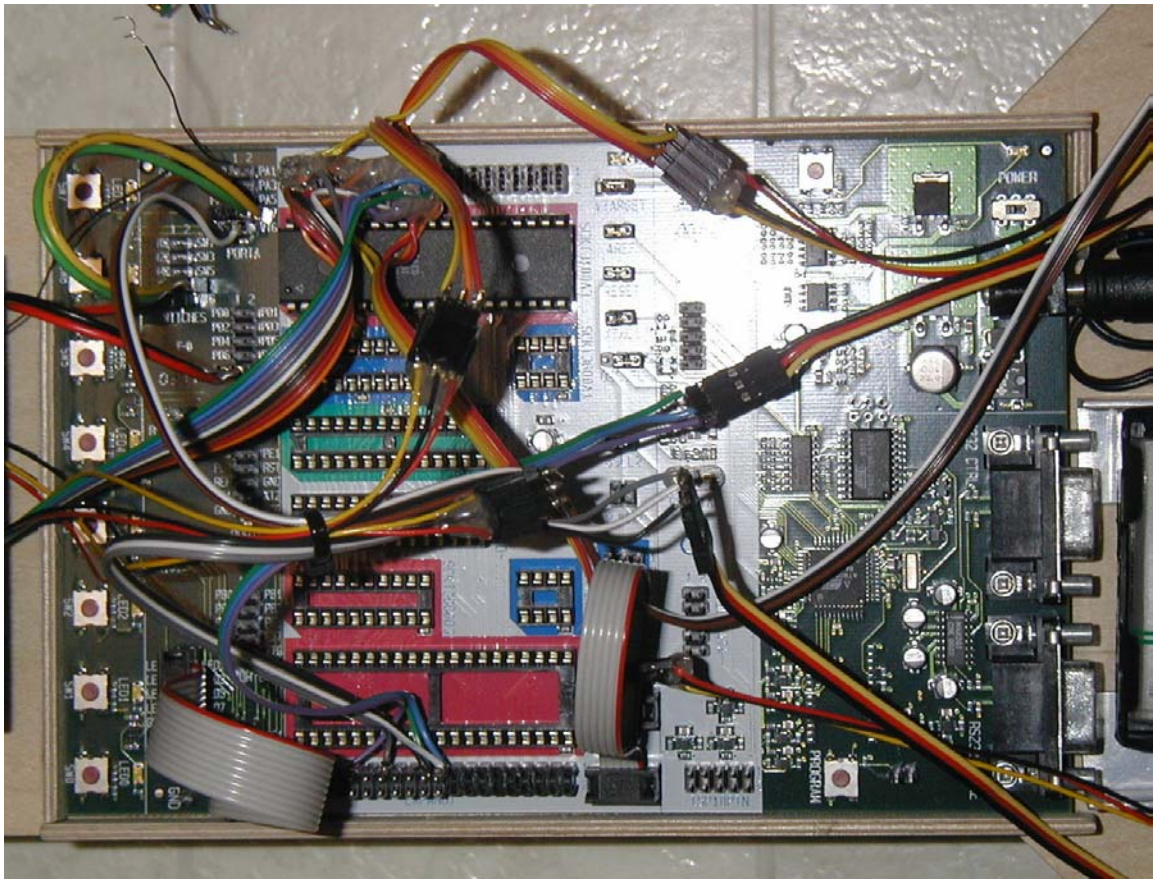
Integrated System

SPARtan's logical process is one which is not very difficult to comprehend.



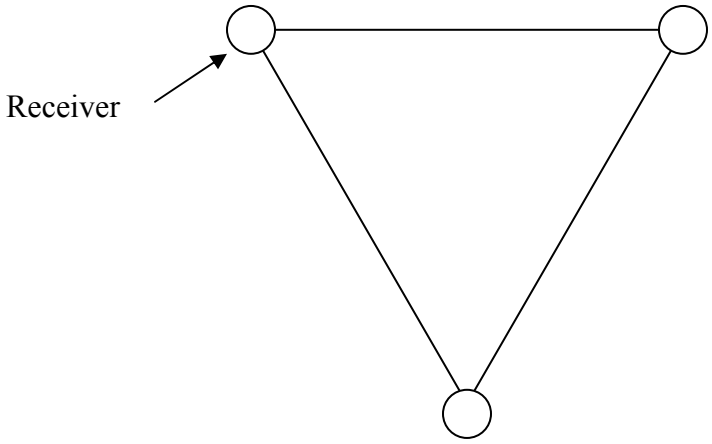
Currently SPARtan cannot receive a sonar pulse and therefore align itself towards the sonar beacon. SPARtan's functions as of now are Obstacle Avoidance and displaying the current status on the LCD display.

SPARtan uses an Atmel STK 500 processor board with an ATmega32 microprocessor.

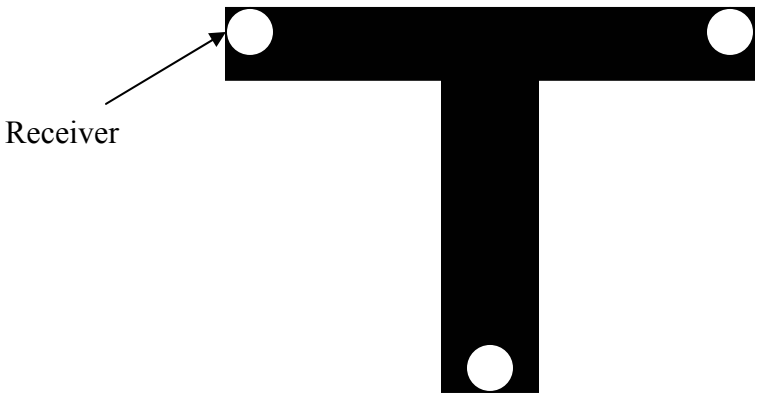


Mobile Platform

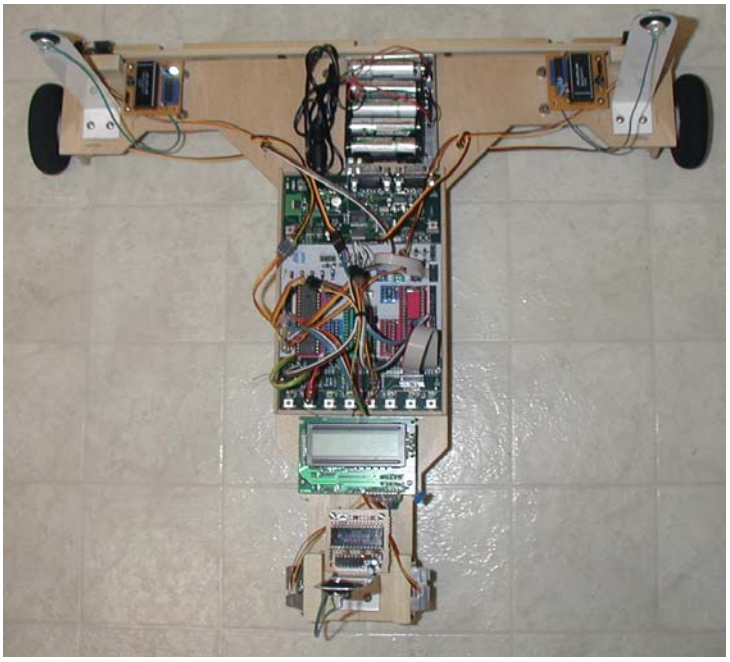
The most logical arrangement of the sonar receivers is to line them up in the shape of a triangle. This will allow all directions in the horizontal plane to be covered. Also, the receivers must be spread out so that accurate reading can be read making the robot more efficient at finding its target. So a triangular shaped platform would be the best decision.



To cut down on wood consumption, I decided to go with a T-shaped design allowing me to save wood while maintaining a sleek design.



Here is a top view of the final design:



I designed the platform in AutoCAD. I made all the appropriate pieces fit into place the first time which really surprised me. Although the platform isn't that complex, I was very happy with the way it turned out. One tricky part was making the platform level. With the servos in the front and a ball castor in the back, the correct heights had to be accounted for.

Actuation

Actuation is provided by two servos on the front two corners of SPARtan.

Each of these servos is hacked allowing continuous nonstop movement for turning wheels. I would highly recommend these servos. They work properly every time and are easy control through a PWM signal. The PWM signal required does not have to be exactly what is suggested which is nice. Hacking this servo is very easy. Performing this action requires the use of a Dremel tool on one tab and then the removal of another tab which slides off the potentiometer.

For back support, SPARtan rested on a ball castor. This ball castor was definitely heavy duty and was a little large for most applications. In my case, it fit perfectly into the back. I used this because I needed multi-directional movement in the back.

Sensor Selection:

Basic Sensors:

Bump Switches – Used in obstacle avoidance, these bump switches are a last resort in case my IR sensors fail. These bump switches should really not be used unless I'm trying to avoid a skinny object and my IR sensors fail to see anything in front of me.

Sharp GP2D12 Detector Package – Used in obstacle avoidance, these sensors detect large objects such as walls in front. This sensor uses infrared to detect objects and returns distance information to the microprocessor.

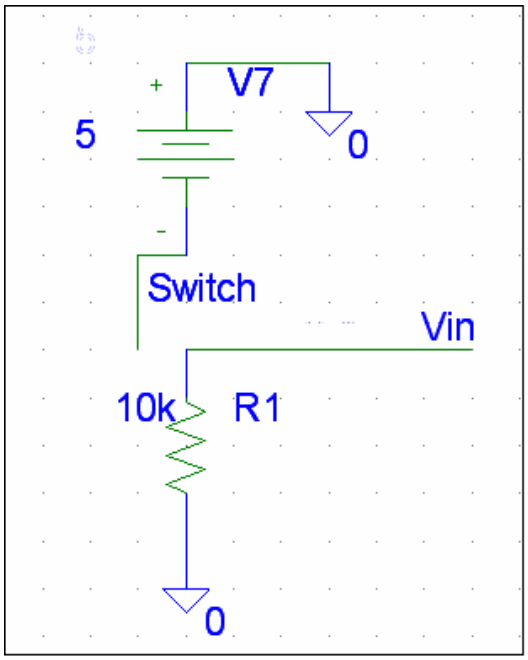
Special Sensors:

Sonar Transmitter – Built by hand, this sensor will be separate from the robot and will act as a beacon emanated a 40 kHz pulse.

Sonar Receiver – Also built by hand, this sensor will receive the pulse given from the transmitter and will relay this to the microprocessor. These will be used to give my position depending on where the transmitter is located.

Bump Switches:

Schematic:



Experimental Setup:

Once the schematic was built, I connect the Vin to one of my port pins on my microprocessor board. I've chosen not to hook it up to one of my A/D channel because I don't really find it necessary and programming is much easier when it's digitally connected to a port pin.

Here's the code I used to test my bump switches:

```
start:
        sbic    PINA, 0x07              ; Don't Start until Switch is pressed
        rjmp    start
.
.
.       It would then let me skip out of this loop and proceed to the rest of the program.
```

Data:

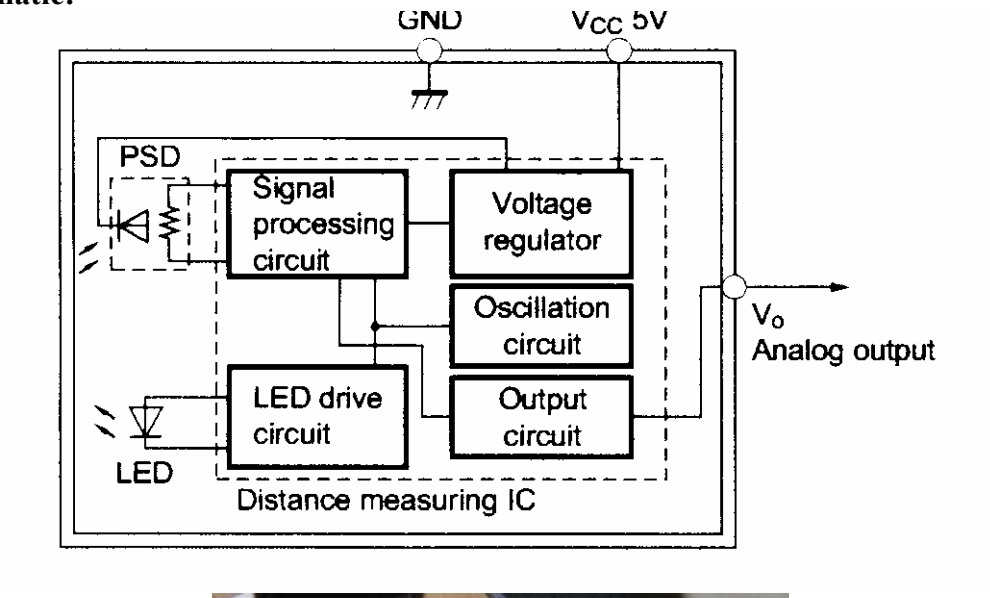
The data received was either a logical 0 or 1. Using the port pins makes this much easier to do.

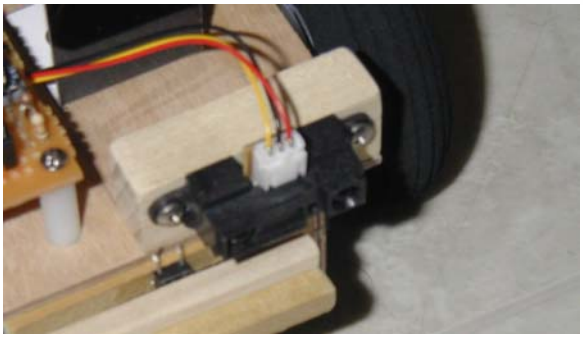
Conclusion:

The bump switches are very reliable and simple to use. Every time they are hit, there's always a change in value that the microprocessor can pick up.

Sharp GP2D12 Detector Package:

Schematic:





Experimental Setup:

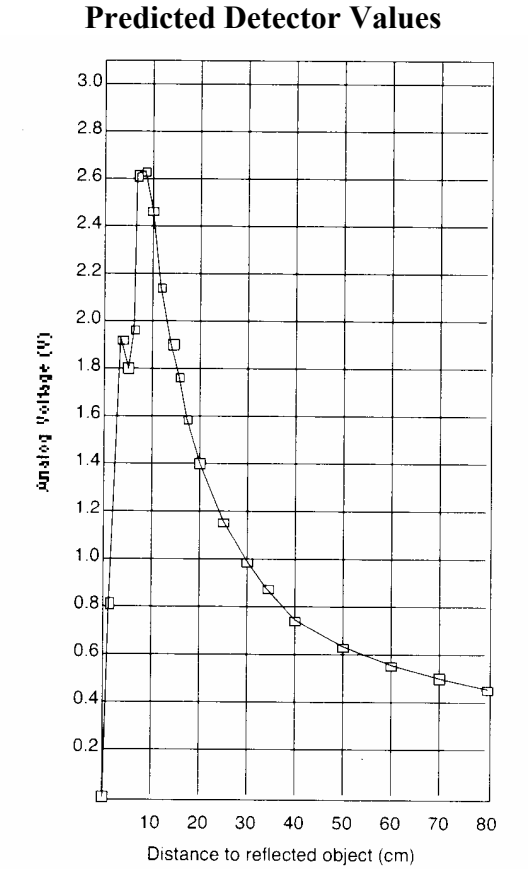
To test this sensor, I connected it to one of my A/D converter channels on my microprocessor. I then laid out a measurement system in front of the sensor ranging from 0 cm to 80 cm. I then tested the analog voltage and binary value given by the analog output Vo from the sensor. I obtained the analog voltage from a voltmeter. I obtained the binary value of the analog output by the LEDs on my microprocessor board. As an obstacle, I used a thin, 8.5" x 11", piece of metal with a piece of white computer paper covering it. I then took measurements starting at 10 cm with a step size of 5 cm.

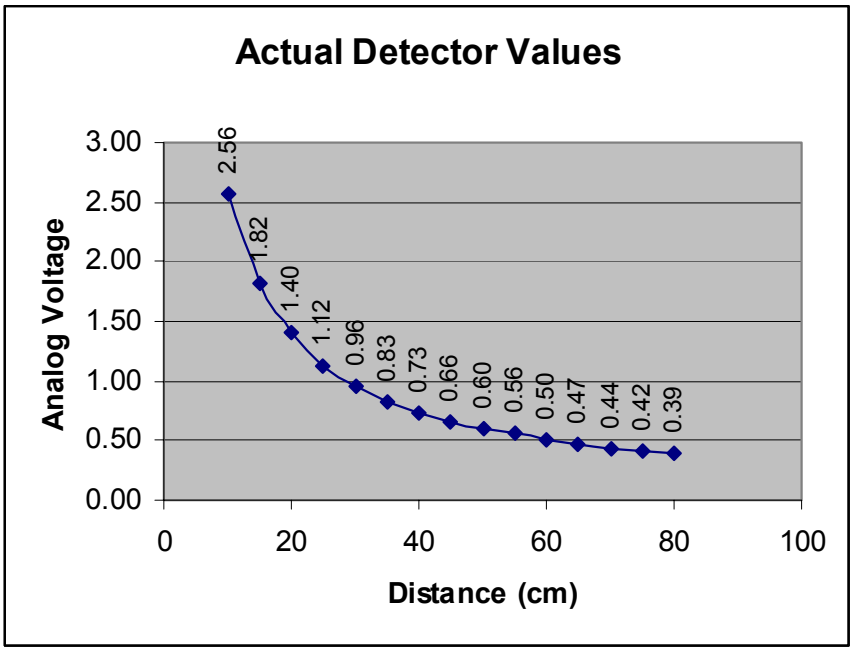
Data:

Distance (cm)	Analog Voltage	Binary Value
10	2.56	10000000
15	1.82	10100000
20	1.40	10111000
25	1.12	11000100
30	0.96	11001111
35	0.83	11010100
40	0.73	11011010
45	0.66	11011110
50	0.60	11100001
55	0.56	11100011
60	0.50	11100110
65	0.47	11101001
70	0.44	11101001
75	0.42	11101011
80	0.39	11101100
>> 80	0.23	11110000

Note: The binary values shown are from the LED display on the board, therefore these values are active low. Complement these values to get real actual values.

Graphs:


Predicted Detector Values
Analog Voltage (V)
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
10 20 30 40 50 60 70 80
Distance to reflected object (cm)


Actual Detector Values
Analog Voltage
3.00
2.50
2.00
1.50
1.00
0.50
0.00
2.56
1.82
1.40
1.12
0.96
0.83
0.73
0.66
0.60
0.56
0.50
0.47
0.44
0.42
0.39
0 20 40 60 80 100
Distance (cm)

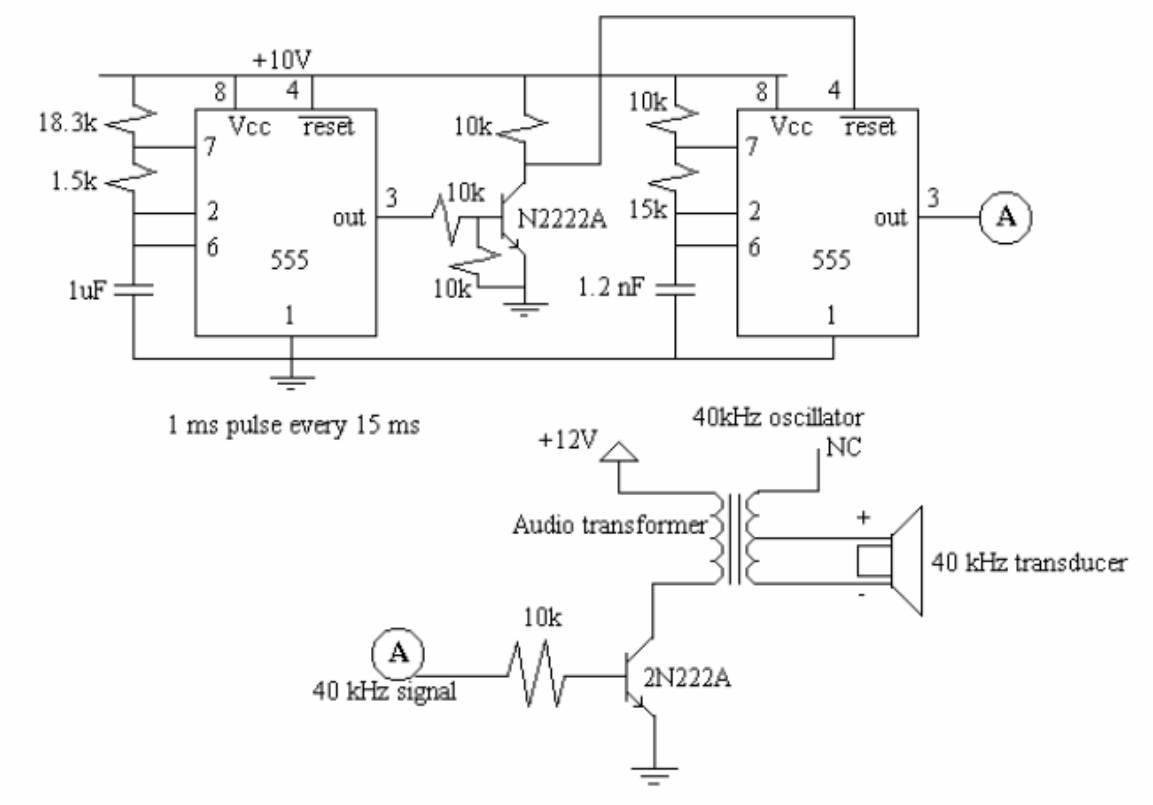
Conclusion on Sharp GP2D12 Detector Package:

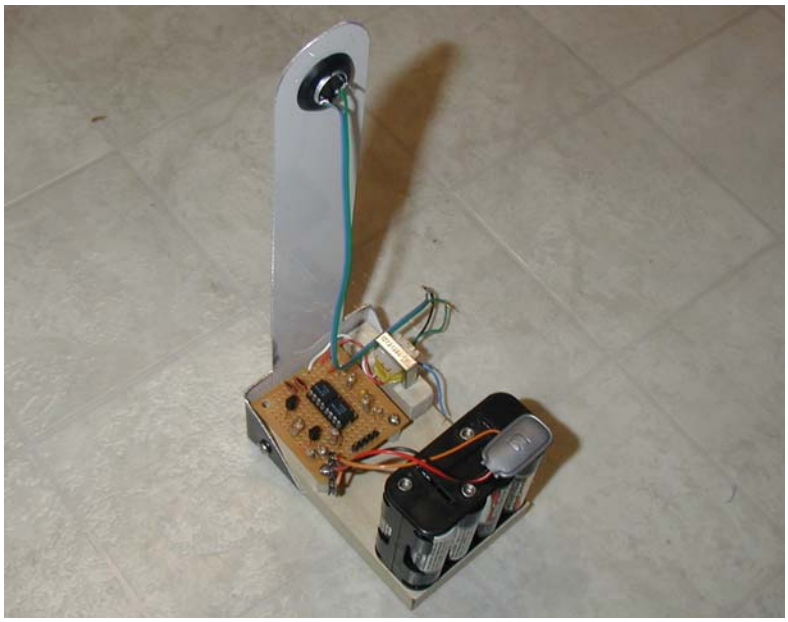
The actual values received from the analog voltage out pin were pretty close to the predicted values given. The range was surprisingly further than I expected. Sharp gave a range of up to 80 cm but one could use this sensor for probably up to 120 cm and still have somewhat accurate readings.

The obstacle's position according to the direction in which the sensor faced is very important. The obstacle had to almost be in a straight line for the sensor to detect that something was in front of it.

Sonar Transmitter:

Schematic:





Sonar Receiver:

Schematic:

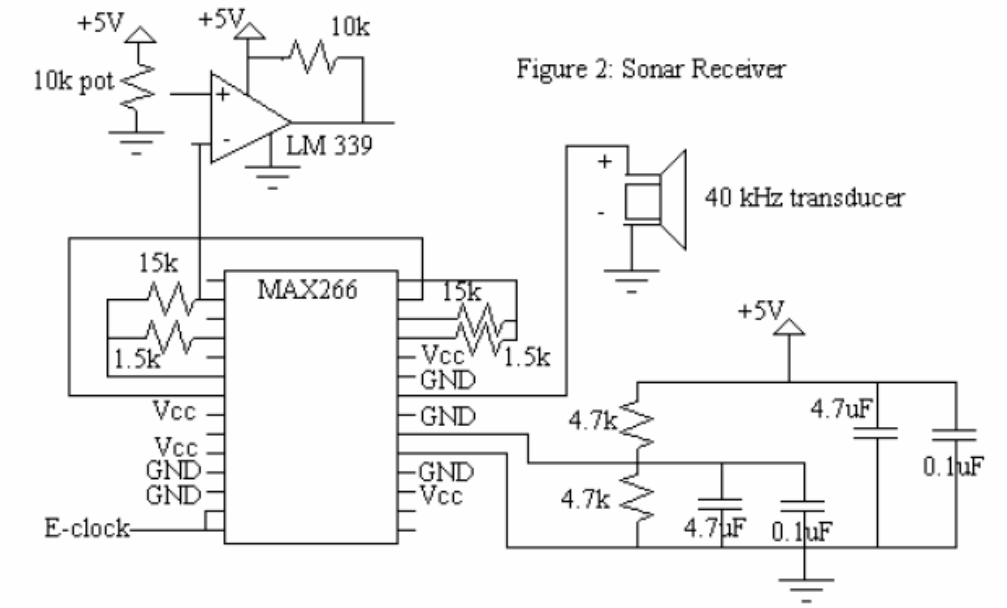
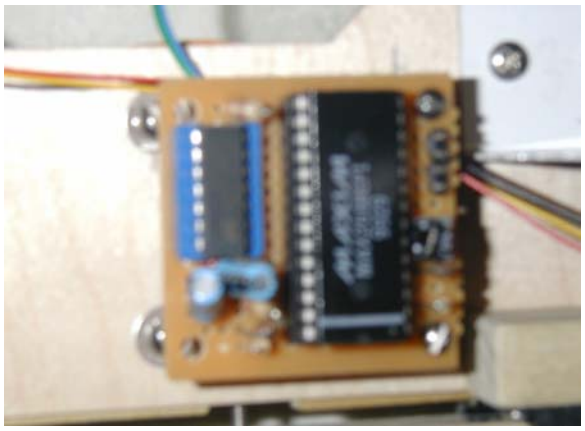
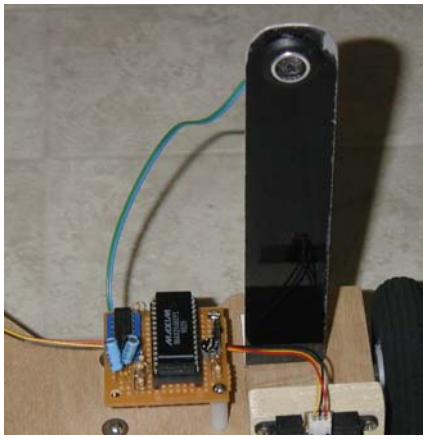


Figure 2: Sonar Receiver





Experimental Setup:

I used an oscilloscope to measure the filter pulse that this receiver would obtain. The values obtain were not correct. I don't think the schematic was setup properly. With numerous attempts at trying to get this to work, I did try an RCK filter to try and get a signal. This actually did work but both transducers had to be about an inch away from each other. Therefore the transmitter did work but the receivers were faulty.

Behaviors

Acquiring Direction:

This behavior determines in which direction the sonar beacon is. SPARtan continues to receive the pulse in each of its three sonar receivers. Based on when the pulse get to each receiver, I can then determine which direction SPARtan must turn in order to be aligned with the target.

For example:
Back Receiver much sooner than Front 1 and Front 2 => Facing opposite
Front 1 sooner than Front 2 and back => Turn left slightly until Front 1 = Front 2
Front 2 sooner than Front 1 and back => Turn right slightly until Front 1 = Front 2

Once this is achieved, this behavior is done.

Obstacle Avoidance:

Bump Switch Panel: Anytime the panel is hit, SPARtan stops and is done.

IR detection: When the IR reaches tolerance because of an obstacle, SPARtan stops, backs one wheel and then presses onward.

LCD Display:

This behavior shows the current status of SPARtan when it is either going forward or reaching an obstacle and avoiding.

Conclusion

In summary, I was able to get obstacle avoidance to work well. SPARtan reacts very fast to obstacles at a close distance and moves out of the way. During all of his actions, the LCD displays describes his current actions as he moves.

My work is limited because the sonar system didn't perform. Building these schematics proves to be time consuming and the equipment always isn't the best when assembling these on a board. Because of this problem, I wasn't able to perform the main functions of my program.

If I were to do this all over, I would not have chosen sonar in this manner. Companies don't make sonar for this type of case and the resources to build your own are not very accurate.

Documentation

Sonar Transmitter and Receiver Schematics and Information:
Megan Grimm
Alph and Ralph
Fall 1998

Appendix A: Vendor Information

Item	Qty	Price
ATMEL ATSTK500 Board Starter Kit • www.digikey.com		$79.00 plus $10.00 shipping
ATMEL ATmega32-16PC-ND MCU • www.digikey.com		$9.89 plus $5.00 shipping
LCD Display • EEL 4744: Microprocessor Applications		Free
Sharp GP2D12 Detector Package • www.acroname.com	(4)	$11.50/each plus $12.00 shipping
HS-425BB Hitec Servo • www.servocity.com	(2)	$14.99/each plus $7.00 shipping
5730 Treaded Lite Wheel 3” • www.towerhobbies.com	(2)	$5.49 together plus $7.00 shipping
MAX266 Filter Chip • www.maxim-ic.com	(2) (1)	Free samples $19.50/each plus $10.00 shipping
LM339 Analog Comparator 40 kHz Transducer Pair • www.jameco.com	(3) (5)	$0.21/each $6.95/each plus $7.00 shipping
Resistors, Capacitors, Transistors, Potentiometers, etc. Anti-Static Kit 1kct/8Ohm Audio-transformer PC board mount • RadioShack	 (1) (1) (1)	$18.00 $24.99 $2.99 $2.99
Flange-mount Ball Castor • www.mcmaster.com	(1)	$3.13 plus $3.75 shipping
30 Minute AA Battery Charger (includes 4 batts) Rechargable NiMH Batteries (4 pack) AA Batteries (8 pack) • Best Buy	(1) (1) (1)	$41.99 $13.99 $3.99
	Total:	**$404.55**

Appendix B: Assorted Code

```
;***** LCD Interface *****
;** Port A is our access to the LCD:
;** PC0 =  DB0        (pn7)
;** PC1 =  DB1        (pn8)
;** PC2 =  DB2        (pn9)
;** PC3 =  DB3        (pn10)
;** PC4 =  E          (pn6)
;** PC5 =  RS         (pn4)
;** PC6 =  Nothing
;** PC7 =  Nothing
;** GND =             VSS       (pn1)
;** GND =             R/W       (pn5)
;** VTG = VDD         (pn2)
.include "m32def.inc"


.def      del1      =r16      ; X Reg
.def      del2      =r17      ; Y Reg
.def      a         =r18      ; A Reg
.def      b         =r19      ; B Reg
.def      Temp      =r20      ; Temp Reg
.def      DReg1     =r21      ; Delay Reg 1
.def      DReg2     =r22      ; Delay Reg 2

;**** Macros

.macro letter
 ldi      a,@0
 out      PORTC,a
 sbi      PORTC,5
 rcall    Latch
 ldi      a,@1
 out      PORTC,a
 sbi      PORTC,5
 rcall    Latch
 rcall    Delay                    ; 80 * 1 * 0.5us = 40us
.endmacro


;** Initialize Port



 ser      Temp
 out      DDRC,Temp                ; Set PortC to output only (LCD)

 ldi      Temp,low(RAMEND)                    ; Set stackptr to ram end
 out      SPL,Temp
 ldi      Temp, high(RAMEND)
 out      SPH, Temp

;** Command Mode

 clr      Temp
 out      PORTC,Temp
;** Initialize 4-bit mode

 ldi      DReg1,150                ; Delay 1 number (
 ldi      DReg2,100                ; Delay 2 number (multiple)
 rcall    Delay                    ; 150 * 100 * 1us = 15ms


 ldi      a,3
 out      PORTC,a
 rcall    Latch
```

```
  ldi       DReg1,41          ; Delay 1 number
  ldi       DReg2,100                 ;
  rcall     Delay                     ; 41 * 100 * 1us = 4.1ms


  rcall     Latch
  ldi       DReg1,100                 ; Delay 1 number
  ldi       DReg2,1                   ;
  rcall     Delay                     ; 100 * 1 * 1us = 100us


  rcall     Latch
  ldi       DReg1,41          ; Delay 1 number
  ldi       DReg2,100                 ;
  rcall     Delay                     ; 41 * 100 * 1us = 4.1ms


  ldi       a,2
  out       PORTC,a
  rcall     Latch

;** Enable 2-line Mode

  ldi       DReg1,40          ; Delay 1 number
  ldi       DReg2,1                   ;
  rcall     Delay                     ; 40 * 1 * 1us = 40us

  rcall     Latch
  ldi       a,12
  out       PORTC,a
  rcall     Latch

;** Display, Cursor, Blink

  ldi       DReg1,40          ; Delay 1 number
  ldi       DReg2,1                   ;
  rcall     Delay                     ; 40 * 1 * 1us = 40us

  ldi       a,0
  out       PORTC,a
  rcall     Latch
  ldi       a,15
  out       PORTC,a
  rcall     Latch
  ldi       DReg1,40          ; Delay 1 number
  ldi       DReg2,1                   ;
  rcall     Delay                     ; 40 * 1 * 1us = 40us


;** Clear Home

  ldi       a,0
  out       PORTC,a
  rcall     Latch
  ldi       a,1
  out       PORTC,a
  rcall     Latch
  ldi       DReg1,164                 ; Delay 1 number
  ldi       DReg2,10          ;
  rcall     Delay                     ; 164 * 10 * 1us = 1.64ms


;** Initialization Complete


;*** Write Name
;** Set RS High for Data Mode
```

```
  sbi       PORTC,5
  ldi       DReg1,40            ; Delay 1 number
  ldi       DReg2,1                       ;

;** My name

  letter    4,2                 ; load "B"
  letter    7,2                 ; load "r"
  letter    7,9                 ; load "y"
  letter    6,1                 ; load "a"
  letter    6,14        ; load "n"
  letter    2,0                 ; load " "
  letter    4,1                 ; load "A"
  letter    7,2                 ; load "r"
  letter    6,11        ; load "k"
  letter    6,9                 ; load "i"
  letter    6,14        ; load "n"
  letter    7,3                 ; load "s"
done:
  rjmp      done

;**** Subroutines

Latch:
  sbi       PORTC,4             ; set E=1
  cbi       PORTC,4             ; set E=0
  ret

Delay:                                  ; 0.5us delay
  mov       del1,DReg1
  mov       del2,DReg2
loop:
  nop
  nop
  nop
  nop
  nop
  nop
  dec       del1
  brne      loop
  mov       del1,DReg1
  dec       del2
  brne      loop
  ret
```

```
;**** Servo Control ****
;  ** Straight Line Forward **
; Our clock is running at 8MHz, therefore we divide
; the clock by the 256 prescalar and let our top be $FF.
;
; 1/8MHz = .125us * 256 * 256 = 8.2ms  from bottom to top
;
; So it's 16.4ms for one period and that's as close to 20ms
; as we get.
;   /\    /\    /  <-- $FF
;  /  \  /  \  /
; /    \/    \/    <-- $00
;
; 31 ---> Full Forward
; 27 ---> Half Forward
; 23 ---> Neutral
; 19 ---> Half Backward
; 15 ---> Full Backward



.include "m32def.inc"

.def        temp        =r16                                ; Temporary Reg 1
.def        Lservo      =r17                                ; Left Servo Reg
.def        Rservo      =r18                                ; Right Servo Reg

            ldi                 Temp, 0b00001000
            out                 DDRB,Temp                   ; Set OC0 (PB3) to output

            ldi                 Temp, 0b10000000
            out                 DDRD,Temp                   ; Set OC2 (PD7) to output

            ldi                 Temp,low(RAMEND)            ; Set stackptr to ram end
            out                 SPL,Temp
            ldi                 Temp, high(RAMEND)
            out                 SPH, Temp

            ldi                 Temp, 0b11100100            ; Output compare setup regs
            out                 TCCR0, Temp                 ; 6(WGM 00) = 1, 3(WGM 01) = 0
            ldi                 Temp, 0b11100110            ; 5(COM 01) = 1, 4(COM 00) = 0
            out                 TCCR2, Temp                 ; for TCCR0: 2:0(CS2:0) = 100
                                                            ; for TCCR2: 2:0(CS2:0) = 110

            ldi                 Temp, $00                   ; Start TCNT's at $00
            out                 TCNT0, Temp
            out                 TCNT2, Temp

            ldi                 Lservo, 31
            ldi                 Rservo, 15
            out                 OCR0, Lservo
            out                 OCR2, Rservo


loop:
            rjmp        loop
```

```
;**** AD Program ****
.include  "m32def.inc"

.def      Temp1    =r16
.def      Temp2    =r17
.def      Del1     =r18
.def      Del2     =r19
.def      Del3     =r20

          clr      Temp1
          out      DDRA, Temp1

          ser      Temp1
          out      DDRB, Temp1

          ldi      Temp1, 0b11100000
          out      ADMUX, Temp1

          ldi      Temp1, 0b11100110
          out      ADCSR, Temp1

          clr      ZH
          ldi      ZL, SFIOR

          ld       Temp1, Z
          sbr      Temp1, 0b11110000
          st       Z, Temp1

loop:
          in       Temp1, ADCH
          out      PortB, Temp1

          ldi      Del1, 100
          ldi      Del2, 100
          ldi      Del3, 50
again:    nop
          nop
          nop
          nop
          nop
          nop
          dec      Del1
          brne     again
          ldi      Del1, 100
          dec      Del2
          brne     again
          ldi      Del2, 100
          dec      Del3
          brne     again
          rjmp     loop
```

```
;**** Obstacle Avoidance Program ****

;** Bryan Arkins
;** EEL 5666
;** Spring 2004

;  ** Straight Line Forward **
; Our clock is running at 8MHz, therefore we divide
; the clock by the 256 prescalar and let our top be $FF.
;
; 1/8MHz = .125us * 256 * 256 = 8.2ms  from bottom to top
;
; So it's 16.4ms for one period and that's as close to 20ms
; as we get.
;   /\    /\    / <-- $FF
; / \ / \ / \ /
; /    \/    \/    <-- $00
;
; 31 ---> Full Forward
; 27 ---> Half Forward
; 23 ---> Neutral
; 19 ---> Half Backward
; 15 ---> Full Backward




.include  "m32def.inc"

.def        Temp1     =r16
.def        Del1      =r17
.def        Del2      =r18
.def        DReg1     =r19
.def        DReg2     =r20
.def        Lservo    =r21                              ; Left Servo Reg
.def        Rservo    =r22                              ; Right Servo Reg
.def        Temp2     =r23
.def        ADval     =r24
.def        Tol       =r25


;**** Macros
.macro letter
            ldi                 Temp1,@0
            ldi                 Temp2,@1
.endmacro

            .org        ADCCaddr
            rjmp        AD_ISR



            .org        $0050
            rjmp        reset



reset:
;*** Setup of Ports and Stack Pointer ***


            clr                 Temp1
            out                 DDRA, Temp1               ; Set A/D and Initial Switch Bit7

            ldi                 Temp1, 0b00001000
            out                 DDRB, Temp1               ; Set OC0 (PB3) to output

            ser                 Temp1
```

```
        out     DDRC, Temp1                     ; Set PortC to output only (LCD)

        ldi     Temp1, 0b10000000
        out     DDRD, Temp1                     ; Set OC2 (PD7) to output

        ldi     Temp1,low(RAMEND)               ; Set stackptr to ram end
        out     SPL,Temp1
        ldi     Temp1, high(RAMEND)
        out     SPH, Temp1


        ldi     Tol, $0b01000000

;*** Initialization of LCD ***

;** Command Mode

 clr    Temp1
 out    PORTC,Temp1
;** Initialize 4-bit mode

 ldi    DReg1,150                               ; Delay 1 number
 ldi    DReg2,10                                ; Delay 2 number (multiple)
 rcall  DelayLCD                                ; 150 * 100 * 1us = 15ms


 ldi    Temp2,3
 out    PORTC,Temp2
 rcall  LatchLCD


 ldi    DReg1,41                                ; Delay 1 number
 ldi    DReg2,10
 rcall  DelayLCD                                ; 41 * 100 * 1us = 4.1ms


 rcall  LatchLCD
 ldi    DReg1,1                                 ; Delay 1 number
 ldi    DReg2,10
 rcall  DelayLCD                                ; 100 * 1 * 1us = 100us


 rcall  LatchLCD
 ldi    DReg1,41                                ; Delay 1 number
 ldi    DReg2,10
 rcall  DelayLCD                                ; 41 * 100 * 1us = 4.1ms


 ldi    Temp2,2
 out    PORTC,Temp2
 rcall  LatchLCD

;** Enable 2-line Mode

 ldi    DReg1,10                                ; Delay 1 number
 ldi    DReg2,2
 rcall  DelayLCD                                ; 40 * 1 * 1us = 40us

 rcall  LatchLCD
 ldi    Temp2,12
 out    PORTC,Temp2
 rcall  LatchLCD

;** Display, Cursor, Blink

 ldi    DReg1,10                                ; Delay 1 number
 ldi    DReg2,2
 rcall  DelayLCD                                ; 40 * 1 * 1us = 40us
```

```
 ldi        Temp2,0
 out        PORTC,Temp2
 rcall      LatchLCD
 ldi        Temp2,15
 out        PORTC,Temp2
 rcall      LatchLCD
 ldi        DReg1,10                                        ; Delay 1 number
 ldi        DReg2,2
 rcall      DelayLCD                                        ; 40 * 1 * 1us = 40us


;** Clear Home

            rcall       ClearHomeLCD


;** Initialization Complete

;** Feel free to write
;** To clear and go again, call subroutine "ClearHomeLCD"


;** SPARtan's Name

            letter      5,3                 ; load "S"
            rcall       letterLCD
            letter      5,0                 ; load "P"
            rcall       letterLCD
            letter      4,1                 ; load "A"
            rcall       letterLCD
            letter      5,2                 ; load "R"
            rcall       letterLCD
            letter      7,4                 ; load "t"
            rcall       letterLCD
            letter      6,1                 ; load "a"
            rcall       letterLCD
            letter      6,14        ; load "n"
            rcall       letterLCD

;*** End of LCD ***



;*** Setup of A/D Conversion ***
            ldi                 Temp1, 0b01100000
            out                 ADMUX, Temp1                    ; Set up for A/D0

            ldi                 Temp1, 0b10001110
            out                 ADCSR, Temp1

            clr                 ZH
            ldi                 ZL, SFIOR

            ld                  Temp1, Z
            sbr                 Temp1, 0b00010000
            cbr                 Temp1, 0b11100000
            st                  Z, Temp1


;*** Setup of Servo Control ***
            ldi                 Temp1, 0b11100100               ; Output compare setup regs
            out                 TCCR0, Temp1                    ; 6(WGM 00) = 1, 3(WGM 01) = 0
            ldi                 Temp1, 0b11100110               ; 5(COM 01) = 1, 4(COM 00) = 0
            out                 TCCR2, Temp1                                    ; for TCCR0: 2:0(CS2:0) = 100
                                                                                        ; for TCCR2:
2:0(CS2:0) = 110

            ldi                 Temp1, $00                      ; Start TCNT's at $00
```

```
            out         TCNT0, Temp1
            out         TCNT2, Temp1

            rcall       Stop

;*** Start of Main ***
            sei

start:
            sbic        PINA, 0x07                          ; Don't Start until Switch 7 is pressed
            rjmp        start

            rcall       Go
            rcall       OnwardLCD

loopMain:

IRleft:
            sbic        PINA, 0x06
            rjmp        EndProg

            sbi         ADCSR, 7 ; disable ADEN
            cbi         ADMUX, 0            ; set A/D0
            sbi         ADCSR, 5 ; enable ADEN
            cbi         ADCSR, 4 ; Clear Flag
            sbi         ADCSR, 6 ; Start Conversion

wait1:
            sbis        ADCSR, 4
            rjmp        wait1
            in          ADval, ADCH
            cbi         ADCSR, 4
            cbi         ADCSR, 7
            cbi         ADCSR, 5

            cp          ADval, Tol
            brmi        IRleft2
            rcall       Go
            rjmp        IRright
IRleft2:
            rcall       Stop
            rcall       ObstacleLCD
            ldi         DReg1, 50
            ldi         DReg2, 150
            rcall       DelayLCD
            rcall       RetreatLCD
            ldi         Rservo, 25
            out         OCR2, Rservo
            ldi         DReg1, 50
            ldi         DReg2, 150
            rcall       DelayLCD
            rcall       OnwardLCD
            rcall       Go
            rjmp        IRleft


IRright:
            sbic        PINA, 0x06
            rjmp        EndProg

            sbi         ADCSR, 7 ; disable ADEN
            sbi         ADMUX, 0            ; set A/D1
            sbi         ADCSR, 5 ; enable ADEN
            cbi         ADCSR, 4 ; Clear Flag
            sbi         ADCSR, 6 ; Start Conversion

wait2:
            sbis        ADCSR, 4
            rjmp        wait2
            in          ADval, ADCH
```

```
        cbi     ADCSR, 4
        cbi     ADCSR, 7
        cbi     ADCSR, 5

        cp      ADval, Tol
        brmi    IRright2
        rcall   Go
        rjmp    IRleft
IRright2:
        rcall   Stop
        rcall   ObstacleLCD
        ldi     DReg1, 50
        ldi     DReg2, 150
        rcall   DelayLCD
        rcall   RetreatLCD
        ldi     Lservo, 21
        out     OCR0, Lservo
        ldi     DReg1, 50
        ldi     DReg2, 150
        rcall   DelayLCD
        rcall   OnwardLCD
        rcall   Go
        rjmp    IRright


EndProg:
        rcall   Stop
        rcall   RIPLCD
EndProg1:
        rjmp    EndProg1


;*** Interupt Service Routines ***
AD_ISR:                                                 ; Switching back and forth the
front two AD convs
        reti



;*** Subroutines ***
Go:
        ldi             Lservo, 28
        ldi             Rservo, 18
        out             OCR0, Lservo
        out             OCR2, Rservo
        ret

Stop:
        ldi             Lservo, 23
        ldi             Rservo, 23
        out             OCR0, Lservo
        out             OCR2, Rservo
        ret

FinishTurn:                                             ; Delay until turn is finished
        andi    ADval, 0b01111111
        cp              ADval, Tol
        brge    FinishTurn
        ret

DelayLCD:                               ; 0.5us delay
        mov     Del1,DReg1
        mov     Del2,DReg2
loopLCD:
        nop
        nop
        nop
        nop
        nop
```

```
        nop
        dec     Del1
        brne    loopLCD
        mov     Del1,DReg1
        dec     Del2
        brne    loopLCD
        dec     DReg2
        brne    loopLCD
        ret

LatchLCD:
        sbi     PORTC,4         ; set E=1
        cbi     PORTC,4         ; set E=0
        ret

ClearHomeLCD:
        cbi     PORTC,5
        ldi     Temp2,0
        out     PORTC,Temp2
        rcall   LatchLCD
        ldi     Temp2,1
        out     PORTC,Temp2
        rcall   LatchLCD
        ldi     DReg1,164               ; Delay 1 number
        ldi     DReg2,4         ;
        rcall   DelayLCD                ; 164 * 10 * 1us = 1.64ms
        sbi     PORTC,5
        ldi     DReg1,10        ; Delay 1 number
        ldi     DReg2,2                 ; Delay 2 number
        ret

ObstacleLCD:
        rcall   ClearHomeLCD
        letter  4,15    ; load "O"
        rcall   letterLCD
        letter  6,2             ; load "b"
        rcall   letterLCD
        letter  7,3             ; load "s"
        rcall   letterLCD
        letter  7,4             ; load "t"
        rcall   letterLCD
        letter  6,1             ; load "a"
        rcall   letterLCD
        letter  6,3             ; load "c"
        rcall   letterLCD
        letter  6,12    ; load "l"
        rcall   letterLCD
        letter  6,5             ; load "e"
        rcall   letterLCD
        letter  2,1             ; load "!"
        rcall   letterLCD
        ret

RetreatLCD:
        rcall   ClearHomeLCD
        letter  5,2             ; load "R"
        rcall   letterLCD
        letter  6,5             ; load "e"
        rcall   letterLCD
        letter  7,4             ; load "t"
        rcall   letterLCD
        letter  7,2             ; load "r"
        rcall   letterLCD
        letter  6,5             ; load "e"
        rcall   letterLCD
        letter  6,1             ; load "a"
        rcall   letterLCD
        letter  7,4             ; load "t"
        rcall   letterLCD
        letter  2,1             ; load "!"
```

```
        rcall   letterLCD
        ret

OnwardLCD:
        rcall   ClearHomeLCD
        letter  4,15            ; load "O"
        rcall   letterLCD
        letter  6,14            ; load "n"
        rcall   letterLCD
        letter  7,7             ; load "w"
        rcall   letterLCD
        letter  6,1             ; load "a"
        rcall   letterLCD
        letter  7,2             ; load "r"
        rcall   letterLCD
        letter  6,4             ; load "d"
        rcall   letterLCD
        letter  2,14            ; load "."
        rcall   letterLCD
        letter  2,14            ; load "."
        rcall   letterLCD
        letter  2,14            ; load "."
        rcall   letterLCD
        ret

RIPLCD:
        rcall   ClearHomeLCD
        letter  5,2             ; load "R"
        rcall   letterLCD
        letter  2,14            ; load "."
        rcall   letterLCD
        letter  4,9             ; load "I"
        rcall   letterLCD
        letter  2,14            ; load "."
        rcall   letterLCD
        letter  5,0             ; load "."
        rcall   letterLCD
        ret

letterLCD:
        out             PORTC,Temp1
        sbi             PORTC,5
        rcall   LatchLCD
        out             PORTC,Temp2
        sbi             PORTC,5
        rcall   LatchLCD
        ldi             DReg1,10        ; Delay 1 number
        ldi             DReg2,2                 ;
        rcall   DelayLCD                ; 80 * 1 * 0.5us = 40us
        ret
```