Sensor Report

University of Florida
Department of Computer and Electrical Engineering
EEL 5666
Intelligent Machine Design Laboratory

Steven Theriault

TA: Uriel Rodriguez
Jason Plew
Instructor: A. A. Arroyo

December 10, 2002
Table of Contents

Overview ........................................................................................................... 2
Sonar Operation ............................................................................................... 3
Results ............................................................................................................... 4
Appendix A ....................................................................................................... 5
Overview

The autonomous robot that I am working on is a replica of a 13th century French trebuchet. Its function is to first search for a castle. Once it finds a castle, it will position itself at a distance that will ensure a hit by a projectile. If the trebuchet hits the castle, the castle will notify the robot that the castle was hit, and the trebuchet will signal victory by playing the French National Anthem.

To find the correct distance from the castle, the robot will use a sonar system. The SRF04 ultrasonic range finder (figure1) from Devantech will be the sonar used. It was purchased from the Mark III Robot Store for $26.00. The specifications say that it is able to detect a 3 cm pole at 2 meters, with a range from 3 cm to 3 m.

Figure 1: SRF04 Ultrasonic Range Finder
Sonar Operation

The sonar works by sending out a sound pulse at a frequency above the human hearing range. When the sound hits an object, the pulse is sent back to the sonar where is can be heard. The sonar measures the time it took for the sound to return to the sonar and can be used to calculate the distance to the object.

The SRF04 has four connections to use the sonar. Two of them are power and ground. There is an input line for initiating a sonar ping, and an output line to receive a pulse. The width of the pulse is determined length of time it takes for the sound to return to the sonar.

To initiate a sonar ping, bring the input line high for 10μs and then back low. This will send a sonic burst out (see figure 2). After 100μs the output will go high and will remain high for 100μs to 18ms. If an echo is not detected, the sonar will time out after 18ms and the output line will go low. To initiate another pulse, the controller must wait 10ms.

Figure 2: SRF04 Timing Diagram
Results

Software was written in assembly language (Appendix A) for the Atmel ATmega163 to test the SRF04. The software prints to the screen the number of cycles it takes to pulse on the output line of the sonar. These cycles have been calculated into seconds and are shown in Figure 3. The graph has imposed over the sample points a linear regression line.

Figure 3: Distance vs. Delay

As can be seen by the graph above, the sonar is very linear. The sonar has a resolution of about 2.5 inches and a useful range of 3 inches to 7 feet. After 7 feet the output is not steady, and an accurate reading cannot be obtained.
Appendix A – Coding for sonar testing

;SonarTest.asm
;
;This program test the SRF04 sonar. It initiates a pulse and then prints the pulse width
to the screen as a 2 byte hexadecimal number. The pulse width is determined by clock
frequency/8
;
;Written by: Steven Theriault
;
;10/17/02

.include "C:\Program Files\Atmel\AVR Tools\AvrAssemble\Appnotes\m163def.inc"

.def temp=r16
.def capturel=r1
.def captureh=r2
.def temp1=r17
.def temp2=r18
.def temp3=r19

.org 0
        jmp main

.org $24
main:
        ldi temp,high(ramend); Stack Pointer Setup
        out SPH,temp
        ldi temp,low(ramend)
        out SPL,temp

;set data direction
        ldi temp,$01
        out DDRB,temp
        ldi temp,0b10110000 ;pins7,5,4 as outputs
        out DDRD,temp

;init UART
        ldi temp,0 ;setup UART baud rate 9600
        out UBRRHI,temp
        ldi temp,38
        out UBRR,temp
sbi UCSRB, TXEN ; enable UART transmit

; init T1
ldi temp, 0b10000010 ; prescaler clk/8
out TCCR1B, temp ; enable input noise filter
; falling edge input capture

mainloop:
sbi PORTB, 0 ; initiate sonar ping
ldi temp, 20 ; wait 10us

wait10us:
dec temp
brpl wait10us

ldi temp, 0b00100100 ; clear overflow flag 1
out TIFR, temp ; clear IC flag

in capturel, TCNT1L ; store timer1 value
in captureh, TCNT1H

cbi PORTB, 0 ; start sonar ping

chk_ic_flg:
in temp, TIFR
sbrs temp, ICF1 ; wait for IC flag
rjmp chk_ic_flg

in temp, ICR1l ; load capture registers
in temp1, ICR1h

cp temp, capturel ; test original time with new capture time
cpc temp1, captureh

brsh captSub ; if new > original branch

; $FFFF - original time + new time
ldi temp2, $ff ; use temp3:temp2 for accumulator
ldi temp3, $ff

sub temp2, capturel ; $FFFF - original time
sbc temp3, captureh
add temp2,temp ; + new time
adc temp3,temp1

mov temp,temp2 ; move accumulator back to temp,temp1 to

print via UART
mov temp1,temp3

jmp printResults

captSub:
    sub temp,capturel ; subtract 16bit time
    sbc temp1,captureh

printResults:
    push temp1 ; push capture time for UART
    call sendHEX

    push temp
    call sendHEX
    call sendEOL

    ldi temp,$4F ; wait 10ms

wait10ms:
    ldi temp1,$FF
wait10msa:
    dec temp1
    brne wait10msa
    dec temp
    brne wait10ms

    jmp mainloop

; Method to send hex number to UART
; gets number from stack
; stack is emptied after routine call
sendHEX:
; UART TX most significant nibble
    pop temp2 ; save return addr from stack
    pop temp3
    pop temp ; pop off stack to temp reg
    push temp ; push temp back on stack

    swap temp ; make most sig nibble the least sig nibble
andi temp,$0F ;zero upper nibble

cpi temp,$a ;test for correcting a-f
brmi addThirty1 ;between 0 and 9

ldi temp1,7
add temp,temp1 ;add 7 to temp

addThirty1:
ldi temp1,$30
add temp,temp1 ;add $30 to temp

waitTX1:
sbis UCSRA,UDRE ;wait until tx reg is empty
jmp waitTX1

out UDR,temp ;send data

;UART TX least significant nibble
pop temp ; pop off stack to temp

andi temp,$0F ;zero most sig nibble

cpi temp,$a ;test for correcting a-f
brmi addThirty ;between 0 and 9

ldi temp1,7
add temp,temp1 ;add 7 to temp

addThirty:
ldi temp1,$30
add temp,temp1 ;add $30 to temp

waitTX:
sbis UCSRA,UDRE ;wait until tx reg is empty
jmp waitTX

out UDR,temp ;send data

push temp3 ;push return addr
push temp2

ret ;return from subroutine
;Method to send EOL to UART
;
;
sendEOL:
    ldi temp,$a ;send Line Feed

waitTX2:
    sbis UCSRA,UDRE ;wait until tx reg is empty
    jmp waitTX2

    out UDR,temp ;send data

    ldi temp,$d ;send Carriage Return

waitTX3:
    sbis UCSRA,UDRE ;wait until tx reg is empty
    jmp waitTX3

    out UDR,temp ;send data

    ret ;return from subroutine