

Final Project Report

Brandon Bernier

ECE 3525: Embedded Systems

Professor Eom

May 8, 2012

## 1. Objective

The goal of this final project was to design, code, and test a functional final project on the STK500 board using the AVR ATmega16 microcontroller. For this specific project, the student tied in material that was already used throughout the semester as well as some components being used for their senior design project. The objective of the project was to implement an ultrasonic rangefinder capable of reading distances and displaying them to a computer terminal via the serial port of the STK500. Additionally, the rangefinder would control a speaker outputting eight separate tones depending on the distance to the nearest object. This system could be used as an instrument of sorts. The STK500 board's LEDs would also light depending on the tone currently being played. Also, an extra LED was attached that was dimmed via PWM depending on the distance reported back from the rangefinder. The entire code was planned out, written in AVR Studio, and tested on the STK500 to ensure it was working properly.

## 2. Procedure

The design procedure for the final project resembled much of what has been done for every lab experiment throughout the semester. First, the student needed to determine what features learned throughout the semester should be used and how to best integrate all of them. One positive from this semester was that much of the code used in the lab experiments was reusable and simple to transfer into larger programs to implement that functionality. For instance, code for serial communication and PWM could be directly copied from previous labs and integrated to get a lot of the necessary functionality in this project.

### **Specifications:**

- Prints distances to screen in feet and inches to the nearest inch
- Plays 8 different tones between 6in and 30in at 3in intervals
- LEDs light from left to right as the note goes higher and are off when no note is playing
- External LED has 8 different dim levels shown during the 8 different intervals

### **Equipment and Components:**

- (1) PC with AVR Studio 4
- (1) Atmel AVR ATmega16 Microcontroller
- (1) Atmel STK500 Project Board
- (1) Breadboard
- (3) Break Away Headers
- (1) DB9 Serial RS-232 Cable
- (1) 470 $\mu$ F Electrolytic Bypass Capacitor
- (1) Maxbotix LV-EZ3 Ultrasonic Range Finder
- (1) SN74LS04 Hex Inverter
- (1) Green LED
- (1) 2N3904 Transistor
- (1) 6W Speaker
- (1) 4.7k $\Omega$  Resistor
- Assorted lengths of wire

## Inputs & Outputs:

Inputs:

- Ultrasonic Sensor – Serial RX (PD0)

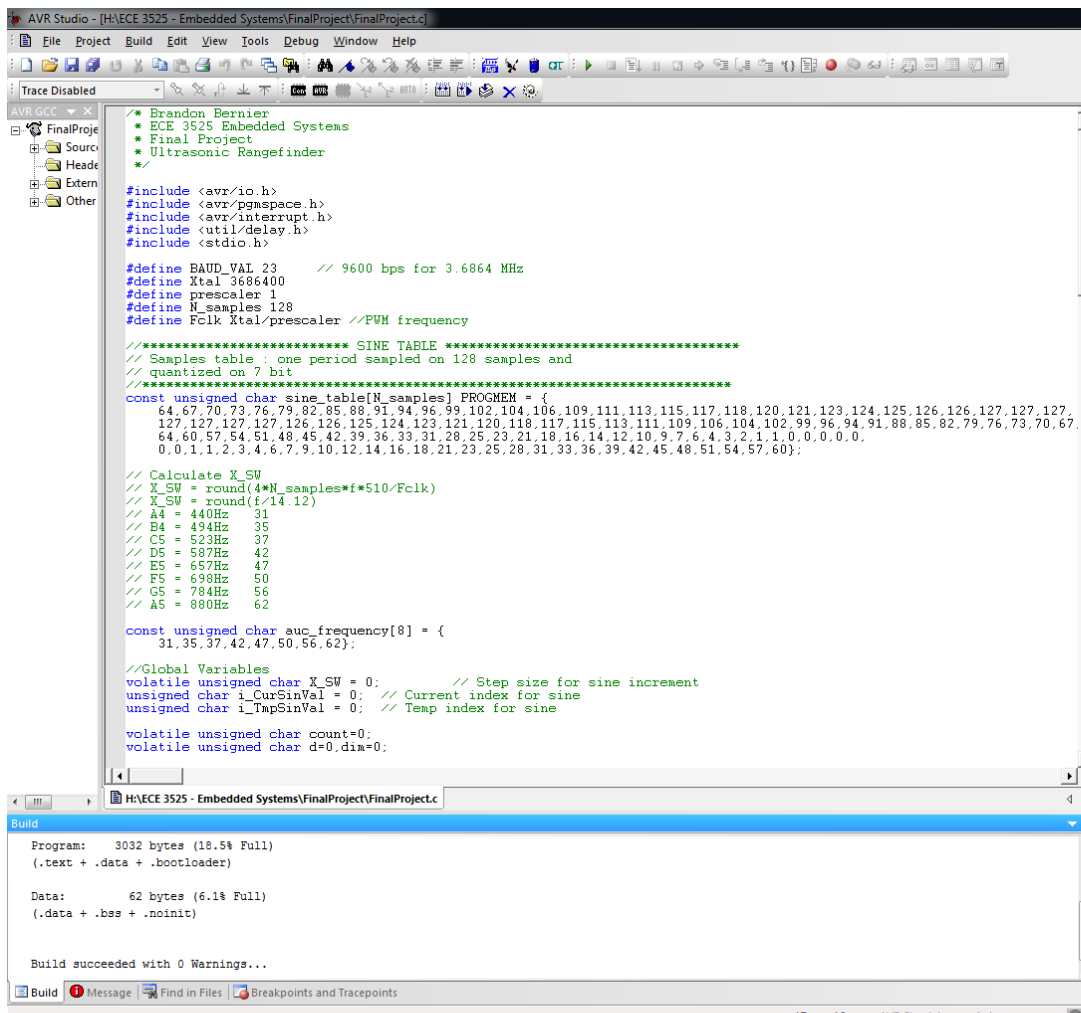
Outputs:

- LEDs 0-7 – PB0-PB7
- Speaker – OCR1A (PD5)
- External LED – OCR1B (PD4)

## Further Design Considerations:

After some initial testing it was determined that the nonstandard RS-232 serial output needed to be inverted before being fed into the ATmega16 to properly communicate. After this, the output was as expected. It was also necessary to determine how often to update the values and read from the sensor. A fast read rate results in better performance in the overall system. The capabilities of the ATmega16 were used extensively including both the RX and TX of the USART, PWM to dim an LED and create tones, program memory for the sine table, as well as various interrupts.

## 3. Results



```
AVR Studio - [H:\ECE 3525 - Embedded Systems\FinalProject\FinalProject.c]
File Project Build Edit View Tools Debug Window Help
Trace Disabled
AVR GCC
FinalProje
Source
Header
Extern
Other
/* Brandon Bernier
 * ECE 3525 Embedded Systems
 * Final Project
 * Ultrasonic Rangefinder
 */
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>

#define BAUD_VAL 23 // 9600 bps for 3.6864 MHz
#define Xtal 3686400
#define prescaler 1
#define N_samples 128
#define Fclk Xtal/prescaler //PWM frequency

//***** SINE TABLE *****
// Samples table : one period sampled on 128 samples and
// quantized on 7 bit
//*****
const unsigned char sine_table[N_samples] PROGMEM = {
  64,67,70,73,76,79,82,85,88,91,94,96,99,102,104,106,109,111,113,115,117,118,120,121,123,124,125,126,126,127,127,127,
  127,127,127,127,126,126,125,124,123,121,120,118,117,115,113,111,109,106,104,102,99,96,94,91,88,85,82,79,76,73,70,67,
  64,60,57,54,51,48,45,42,39,36,33,31,28,25,23,21,18,16,14,12,10,9,7,6,4,3,2,1,1,0,0,0,0,0,
  0,0,1,1,2,3,4,6,7,9,10,12,14,16,18,21,23,25,28,31,33,36,39,42,45,48,51,54,57,60};

// Calculate X_SW
// X_SW = round(4*N_samples*f*510/Fclk)
// X_SW = round(f/14.12)
// A4 = 440Hz 31
// B4 = 494Hz 35
// C5 = 523Hz 37
// D5 = 587Hz 42
// E5 = 657Hz 47
// F5 = 698Hz 50
// G5 = 784Hz 56
// A5 = 880Hz 62

const unsigned char auc_frequency[8] = {
  31,35,37,42,47,50,56,62};

//Global Variables
volatile unsigned char X_SW = 0; // Step size for sine increment
unsigned char i_CurSinVal = 0; // Current index for sine
unsigned char i_TmpSinVal = 0; // Temp index for sine

volatile unsigned char count=0;
volatile unsigned char d=0,dim=0;
```

Build

Program: 3032 bytes (18.5% Full)  
(.text + .data + .bootloader)

Data: 62 bytes (6.1% Full)  
(.data + .bss + .noinit)

Build succeeded with 0 Warnings...

Figure 1 – Compiled C Code for the Final Project

#### 4. Discussion

While the design process of this device has already been described at length, it is necessary to outline some of the difficulties and challenges faced when implementing the code to fulfill this design. The major difficulty encountered was getting the tones to sound natural with everything else going on in the code. When USART communication was implemented with PWM tone generation, the tones would be cut and choppy. This is due to the fact that the interrupts of the USART interfere with the PWM. To overcome this, the delay between reads of the sensor was cut down in an attempt to reduce any additional time taken. In the end, the tones were not exactly as they should be, but this seemed to just be a limitation of the implementation and microcontroller itself. If the device were clocked higher, it may have been possible to reduce the chopping even further.

Additionally, the issue concerning the nonstandard RS-232 serial output from the sensor itself took some time to figure out. Reading through the data sheet as well as the manufacturer's website detailing how to interface with the sensor led to the discovery that the output should be inverted in order to obtain the standard RS-232 values. Originally, the values range between  $0V-V_{cc}$  but standard RS-232 puts positive voltage as a 0 and negative voltage as a 1. Inverting the signal gives the desired results and was necessary to get the proper data.

The easiest way of being able to actually implement this code turned out to breaking it down into many sections. The main part of the code is actually fairly short and much of the functionality of the program happens within interrupts outside of the main function. This was useful because each of those components for the most part were imported from previous lab sessions and used to implement similar functionality. The integration of many different components and capabilities was most important for the success of this project.

An appendix with the entire source code for the final project is attached in the **Code Appendix**, with some accompanying comments.

#### 5. Conclusion

This project challenged students to design, code, and test a functional and useful device using the ATmega16 microcontroller on the AVR STK500 project board. While the project was fairly straightforward in its execution, the integration of all of the components allowed students to gain a much better understanding of the full capabilities of microcontrollers in embedded systems. Overall, students were challenged by the requirements and implementation of this project, but the results were extremely fulfilling and rewarding. Watching as the design and implementation came together to create an interesting project mimicking a form of musical instrument was great to see. The skills learned throughout this class and particularly during this project will be extremely useful and valuable moving forward.

#### 6. References

- |                                      |   |
|--------------------------------------|---|
| [1] Ultrasonic Sensor Page           | <a href="https://www.sparkfun.com/products/8501">https://www.sparkfun.com/products/8501</a>               |
| [2] 2N3904 Transistor Data Sheet     | <a href="http://www.fairchildsemi.com/ds/2N/2N3904.pdf">http://www.fairchildsemi.com/ds/2N/2N3904.pdf</a> |
| [3] SN74LS04 Hex Inverter Data Sheet | <a href="http://www.ti.com/lit/ds/symlink/sn74ls04.pdf">http://www.ti.com/lit/ds/symlink/sn74ls04.pdf</a> |
| [4] Breakaway Headers Page           | <a href="https://www.sparkfun.com/products/116">https://www.sparkfun.com/products/116</a>                 |
| [5] ATmega16 Data Sheet              | <a href="http://www.atmel.com/Images/doc2466.pdf">http://www.atmel.com/Images/doc2466.pdf</a>             |
| [6] AVR STK500 User Guide            | <a href="http://www.atmel.com/Images/doc1925.pdf">http://www.atmel.com/Images/doc1925.pdf</a>             |

## 7. Code Appendix

```
/* Brandon Bernier
 * ECE 3525 Embedded Systems
 * Final Project
 * Ultrasonic Rangefinder
 */

#include <avr/io.h>
#include <avr/pgmspace.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>

#define BAUD_VAL 23 // 9600 bps for 3.6864 MHz
#define Xtal 3686400
#define prescaler 1
#define N_samples 128
#define Fclk Xtal/prescaler //PWM frequency

//***** SINE TABLE *****
// Samples table : one period sampled on 128 samples and
// quantized on 7 bit
//*****
const unsigned char sine_table[N_samples] PROGMEM = {
64,67,70,73,76,79,82,85,88,91,94,96,99,102,104,106,109,111,113,115,117,118,120,121,123,
124,125,126,126,127,127,127,127,127,127,127,126,126,125,124,123,121,120,118,117,115,113,
111,109,106,104,102,99,96,94,91,88,85,82,79,76,73,70,67,64,60,57,54,51,48,45,42,39,36,33,
31,28,25,23,21,18,16,14,12,10,9,7,6,4,3,2,1,1,0,0,0,0,0,0,0,1,1,2,3,4,6,7,9,10,12,14,16,
18,21,23,25,28,31,33,36,39,42,45,48,51,54,57,60};

// Calculate X_SW
// X_SW = round(4*N_samples*f*510/Fclk)
// X_SW = round(f/14.12)
// A4 = 440Hz 31
// B4 = 494Hz 35
// C5 = 523Hz 37
// D5 = 587Hz 42
// E5 = 657Hz 47
// F5 = 698Hz 50
// G5 = 784Hz 56
// A5 = 880Hz 62

const unsigned char auc_frequency[8] = {
31,35,37,42,47,50,56,62};

//Global Variables
volatile unsigned char X_SW = 0; // Step size for sine increment
unsigned char i_CurSinVal = 0; // Current index for sine
unsigned char i_TmpSinVal = 0; // Temp index for sine

volatile unsigned char count=0;
volatile unsigned char d=0,dim=0;

//Function Prototypes
void initialize(void);
unsigned char decode(unsigned char x);
unsigned char decode_sw(unsigned char x);

//USART Function Definitions
int usart_putchar(char c, FILE *stream){
    if( c == '\n' ) usart_putchar('\r', stream);
    while( !(UCSRA&(1<<UDRE)) ); // Wait until UDR is empty
    UDR = c;
    return 0;
}
```

```

int usart_getchar(FILE *stream){
    if( UCSRA&(1<<RXC) ) return UDR;
    else return 0;
}

FILE usart_str = FDEV_SETUP_STREAM(usart_putchar, usart_getchar, _FDEV_SETUP_RW);

int main(void)
{
    initialize();
    printf("\n\nDistance:\n");
    while(1);
    return 0;
}

ISR(USART_RXC_vect){
    unsigned char x,y;
    x = decode( getchar() );
    if( x == 'R' )
        d = 0;
    else if( x == '\r' )
        printf("%c",x);
    else
    {
        if( count == 1 )
            d += x*100;
        else if( count == 2 )
            d+= x*10;
        else if( count == 3){
            d += x;
            if( (d<36) )
            {
                if(d<9){ y=1; dim=5; PORTB=0x7F;}
                else if(d<12){ y=2; dim=36; PORTB=0x3F;}
                else if(d<15){ y=3; dim=72; PORTB=0x1F;}
                else if(d<18){ y=4; dim=108; PORTB=0x0F;}
                else if(d<21){ y=5; dim=144; PORTB=0x07;}
                else if(d<24){ y=6; dim=180; PORTB=0x03;}
                else if(d<27){ y=7; dim=216; PORTB=0x01;}
                else{ y=8; dim=255; PORTB=0x00;}
                X_SW = auc_frequency[y-1];
                TCCR1A |= (1<<COM1A1); // Turn on PWM
            }
            else{
                TCCR1A &= ~(1<<COM1A1); // Turn off PWM
                dim = 0;
                PORTB = 0xFF;
            }
            printf("\t%2d ft %2d in\t",d/12,d%12);
            _delay_ms(1);
        }
    }
    count = (count+1)%5;
}

ISR(TIMER1_OVF_vect)
{
    i_CurSinVal += X_SW; // Increment index based on
    frequency // Round after fixing resolution
    i_TmpSinVal = (char)(((i_CurSinVal+2)>>2)&0x7F); // Round after fixing resolution
    and then 7-bit mask
    OCR1A = pgm_read_byte(&sine_table[i_TmpSinVal]); // Output to speaker
    OCR1B = dim;
}

```

```

void initialize(void)
{
    DDRA    = 0x00;
    PORTA   = 0xFF;
    DDRB    = 0xFF;
    PORTB   = 0xFF;

    //PWM
    DDRD    = (1<<PD4)|(1<<PD5); // OCR1A as output
    TIMSK   |= (1<<TOIE1); // Timer 1 interrupt enable
    TCCR1A  |= (1<<COM1A1)|(1<<COM1B1); // set at top
    TCCR1A  |= (1<<WGM10); // phase-correct, 8 bit PWM
    TCCR1B  |= (1<<CS10); // PWM prescaler = 1

    //USART
    stdout = stdin = &usart_str;
    UBRRH   = (unsigned char)(BAUD_VAL>>8); // Set Baud Rate
    UBRRL   = (unsigned char)(BAUD_VAL&0x00FF);
    UCSRB   = (1<<RXEN)|(1<<TXEN); // Enable RX & TX
    UCSRB   |= (1<<RXCIE); // Enable RXC Interrupt
    UCSRC   = (1<<URSEL)|(3<<UCSZ0); // Set 8-N-1
    sei();
}

unsigned char decode(unsigned char x)
{
    unsigned char temp;
    switch(x){
        case '0': temp=0; break;
        case '1': temp=1; break;
        case '2': temp=2; break;
        case '3': temp=3; break;
        case '4': temp=4; break;
        case '5': temp=5; break;
        case '6': temp=6; break;
        case '7': temp=7; break;
        case '8': temp=8; break;
        case '9': temp=9; break;
        case 'R': temp='R'; break;
        case '\r': temp='\r'; break;
        default: break;
    }
    return temp;
}

unsigned char decode_sw(unsigned char x)
{
    unsigned char y;
    switch(x){
        case 0xFE: y = 1; break;
        case 0xFD: y = 2; break;
        case 0xFB: y = 3; break;
        case 0xF7: y = 4; break;
        case 0xEF: y = 5; break;
        case 0xDF: y = 6; break;
        case 0xBF: y = 7; break;
        case 0x7F: y = 8; break;
        default: y = 0; break;
    }
    return y;
}

```