Robert Bray
CS4710
Fall 2005
Senior Project

Laser Tag

CE Senior Project

Fall 2005

Table of Contents

Proposal: The idea of a laser tag system was very appealing. For one, a working system would demonstrate a detailed knowledge and understanding of embedded systems. A laser tag rifle needs to receive information from sensors, create highly accurate outgoing IR signals, keep track of items such as ammo, health, team ID and player ID, and respond to button pushes and trigger squeezes from the user. To accomplish this, hardware must be created and software written, and must work together without any problems. Thus capturing the quintessential nature of a Computer Engineer.

Planning:

Processor Selection: The most difficult part of this project was selecting which processor to use. The initial points of interest in choosing a processor were cost, availability, and functionality. Looking in retrospect, it would have been smart to also consider availability/cost of programming tools, 3rd party support, free C compiler availability, and example code for basic functions.

The first processor in mind for this project was the PIC16f48A. The PIC was very cheap and could perform all needed tasks. However, programming tools were very expensive and there was no free C compiler. A lot of time was spent in vain trying to get the programmer working.

The second and final processor was the Atmega16. There was a gcc port available for this processor (AVR-GCC), there was a very large support base (avrfreaks.net), the programmer for the processor was less than $50 and worked without a problem, and there is plenty of example code in the processor's datasheet. Using this processor, I was able to create a prototype in a matter of days.

Atmel, maker of the Atmega16, also has a very nice demo-board which was very useful in the prototype stage. I was able to use the demo-board as the shooter, and the prototype as the receiver.

Housing Selection: Once I was able to send 2-byte packets of information over IR, I began looking for a suitable housing for the project, making sure I checked many toy stores. I found a nearly perfect choice at a toy store in Draper. The candidate was a toy gun that made sound effects and vibrated when it was triggered. The gun also had a place for a lens with 4 inches of clearance. Lastly, the gun had enough interrior and exterior room to stuff in my electronics, and mount my LCD and buttons on the outside.

Power Supply: Now that I had decided on the housing, I could now choose what power supply to use based on space constraints and power requirements. The gun came configured for 3 AA batteries. While this was sufficient to drive my processor, it was not sufficient to drive the vibrator or send large amounts of current through the IR LED.

I wanted a battery with a power rating of more than 2000 mA/hr. I found a supplier on eBay who had a large amount of 9.6V, 2300 mA/hr batteries that satisfied my space requirements.

Fabrication: Based on the prototype and the demo-board, I created two circuit boards. I used an adjustable voltage regulator to give my processor, sound chip, and IR sensor the appropriate 5V. I used FFT transformers for the IR LED and vibrator in order to source the large amount of current needed. I used a general IO pin for the vibrator and sound chip, and specialized pins for the IR out and IR in. I used the Atmega16's output compare module for the IR out since this could perform the pin toggling at 40 kHz in hardware without using CPU time. I attached the IR in to the input capture module in order to use the processors counter to keep track of the carrier on/off times.

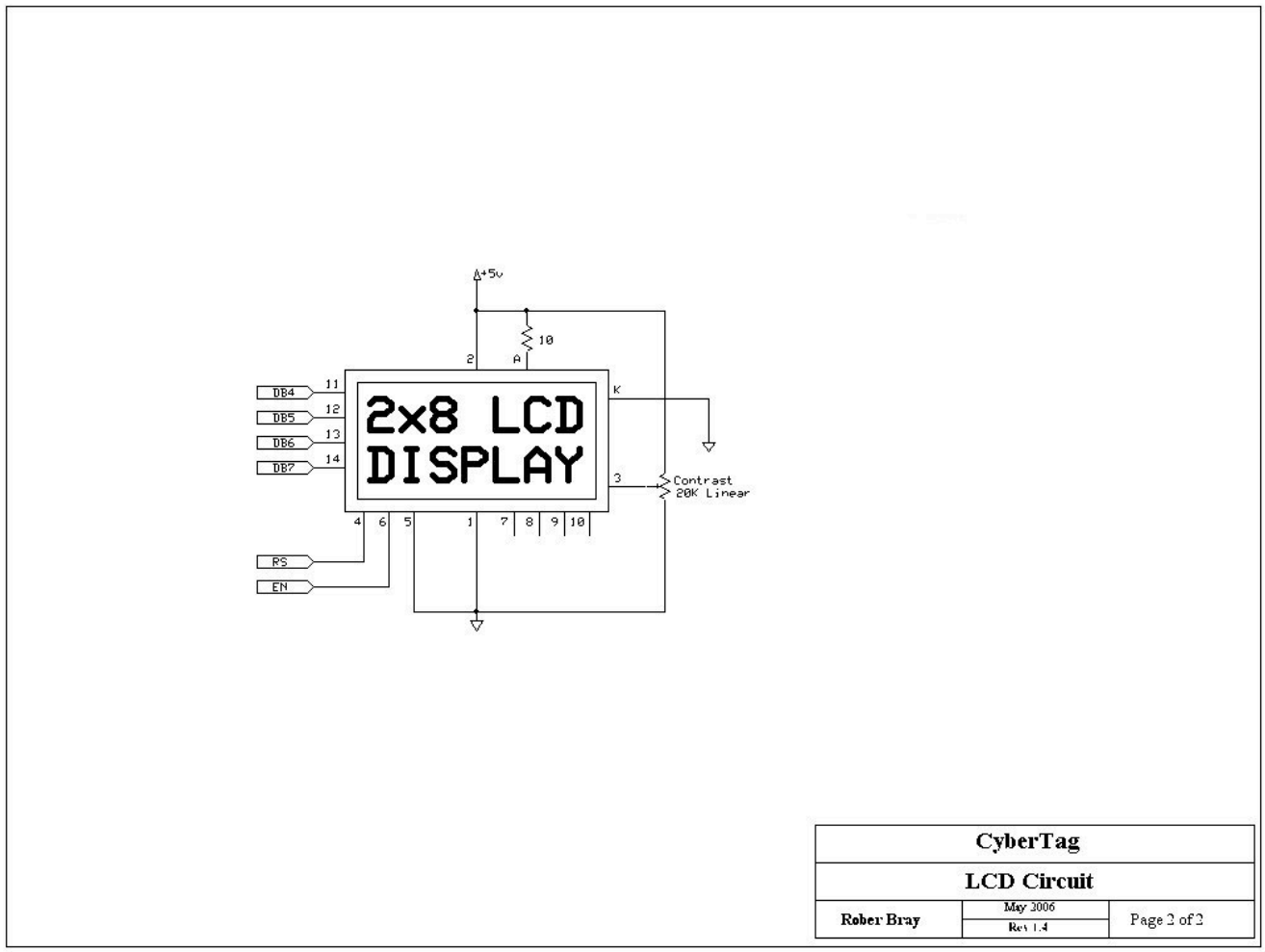
The IR in, trigger, reload, and menu lines were all attached to pins that could be configured as external interrupts. This allowed the processor the ability to sleep most of the time and save battery life.

All code was written from scratch except for the LCD controller. I used C (AVRGCC) to write all code. The code consisted of high precision time delay routines which were used throughout the program. The code also consisted of an encoder/decoder routine that stuffed the player's team ID, player ID, and weapon damage count (pistol, shotgun, missle, etc...) into a 2byte packet. I created an IR sender which sent the packet bit by bit with a header and footer in the front and back.

I used time modulation to send the signal where the standard time unit was 600 us. Therefore, to send a logic level 0, I modulated the IR LED at 40 kHZ for 600 us. To send a logic level 1, I modulated the IR LED at 40 kHz for 1200 us. The header was 2400 us, and the footer was 1800 us. Between each carrier on time, there was a 600 us pause where the IR LED was off. This signal format closely matches the Sony standard, and I was even able to use a Sony remote control at times when a shooter was not available to me.

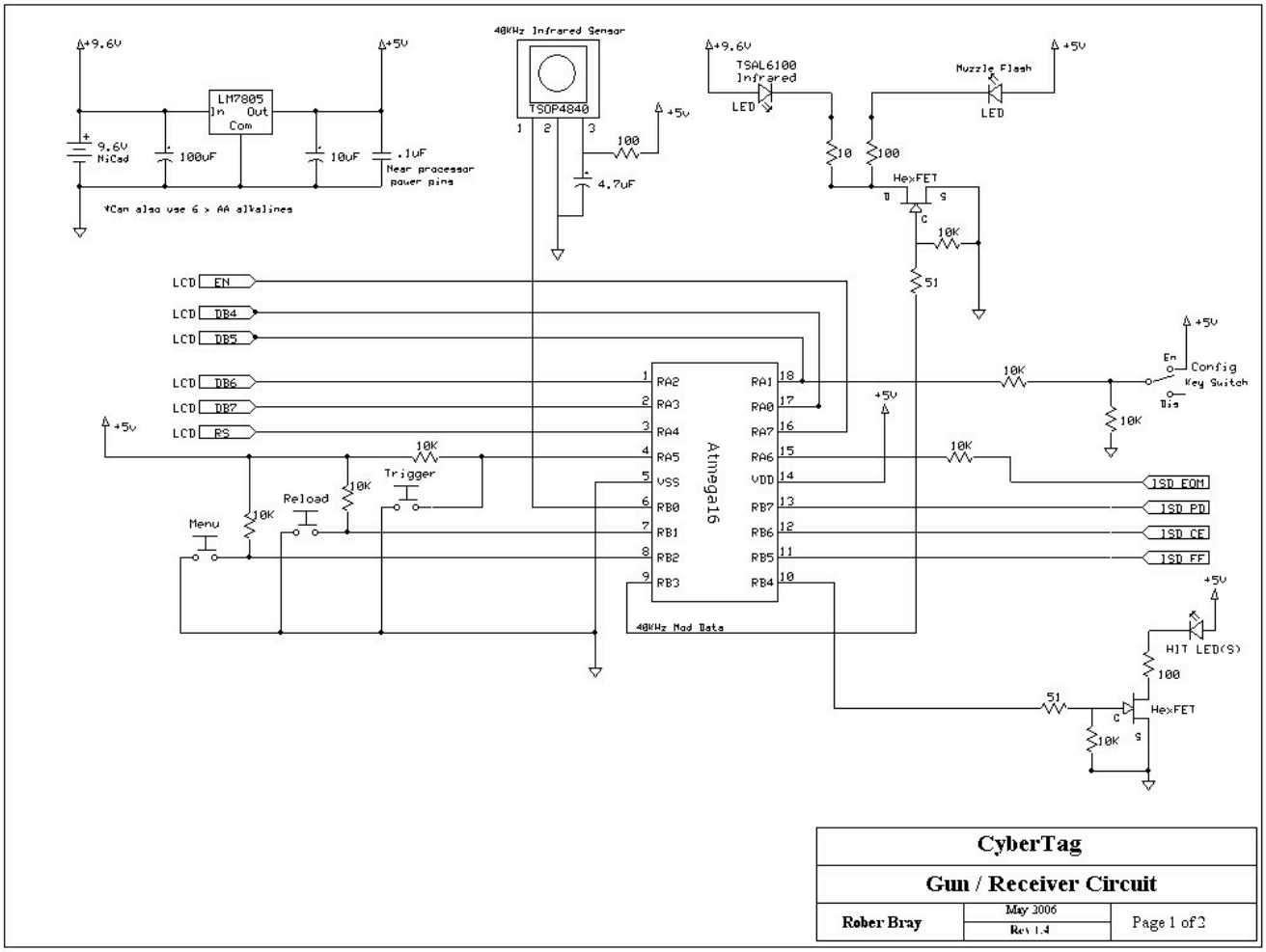
Receiving the IR signal was made much easier by using a demodulating sensor tuned to 40 kHz. In this way, I did not have to add code to demodulate the signal. The sensor tuned out all other IR signals that did not match a 40 kHz carrier frequency, thus eliminating any IR noise from the environment. The line from the signal was an active low signal. Therefore, I configured my Atmega to generate an interrupt when it saw a low signal from the sensor. The ISR consisted of waiting for the line to go back high, then computing how long it was low. It also computed how long the line was high. In this way, I was able to keep track of both the carrier on and off times. I added timeout logic and error handling as well. I placed the sensors in three locations on a baseball cap.

CyberTag LCD Schematic



CyberTag		
LCD Circuit		
Rober Bray	May 2006 Rev 1.4	Page 2 of 2

CyberTag Gun Schematic



CyberTag		
Gun / Receiver Circuit		
Rober Bray	May 2006 Rev 1.4	Page 1 of 2