Laser Tag

Tong Chang, Steve Mott, Marshall Schiring, John Tooker

University of Nebraska – Lincoln

ELEC 494 – Group 34

Spring 2009

Abstract

Laser tag is a game involving infrared (IR) beams volleyed from a transmitter-gun at receivers on opponents' vests. Implementation and development involves multiple areas of electrical engineering from communication theory to microprocessor programming. Background of the game is covered, and then the purpose and design of this implementation is discussed.

Introduction

The purpose of this paper is to describe the implementation of the laser tag senior design project. First, some background and history of laser tag will be discussed followed by a discussion of the group's motivations and novel ideas. The document then will go into detail on the design of the project and specifically how the infrared (IR) transmitter and receivers were designed.

Background

Laser tag is a game that was created in 1979. It can be played as an individual or with a team. First used for combat training by the US Army, it is meant to simulate combat and can be used in a competitive setting. Laser tag is very similar to other sports like paintball, Nerf, and airsoft.

Using the term laser is misleading since IR light emitting diodes (LEDs) are typically used to shoot other players wearing IR receivers instead of "light amplification by stimulated emission of radiation", or, LASERs. This IR technology is actually the same used in a TV remote control.

The game requires durable electronics to take the wear and tear of running players and occasional, accidental contact. Laser tag also requires unique software to tabulate scores and handle other features of the game.

Motivation

Current implementations of the laser tag game allow for exploitation of the game that this project intends to correct. The way players reactivate after being shot by enemies turn the game into a petty chasing game instead of a combat simulation. Another issue is that players can run around madly pressing their trigger without caring to aim. Lastly, when playing with a team, there is very little motivation to work with the team toward a common goal.

Being killed can either be an advantage or a disadvantage in current systems. While dead, the player is temporarily invulnerable to attack. He can then run behind enemy lines, or use this invulnerability period to get an advantageous position he normally would not have been able to get into. The disadvantage is that, after being killed, it is possible to be "camped", or have the player who killed you keep firing at your sensors, until you "respawn" (come back to life) and immediately die again. Players on both sides often try to exploit this issue which turns the game into a frenzy of players standing only feet apart from each other pulling their triggers as fast as they can in the hope that their shot will register on the other player's vest before the other player's shot registers on their own. Obviously this is not fun for any of the players.

To fix this problem there will be locations in the arena in which players can respawn. This area can be team controlled or neutral. This would mean dead players will no longer be able to take advantage of their invulnerability period, however, they can still be camped. This will have to be resolved either by allowing a team capture spawn points (adding to team synergy) or somehow make it neutral and allow anyone to spawn, but disable all guns near the spawn location.

Another feature that will be added is ammunition. This will also add a reload feature to the gun, requiring a time penalty to reload the gun. This modification will stop players from being able to spray the arena with their laser. They will, instead, have to be more careful with their aim and be responsible about finding and using cover effectively. This will discourage living players from camping dead players since there is a chance that the dead player will respawn as the living player reloads. Finally, addition of these features will require more team communication and strategy.

One final feature implemented added variety to the game play. Too often a game can get monotonous and lose its replay value. In order to make the game more interesting and complicated, two different gun types were introduced in the game. There is both a pistol and machine gun. The pistol is semi-automatic (single shot each trigger pull, but multiple shots per reload) with quick fire rate and is very easy to aim and reload. The machine gun has a much larger magazine size (amount of shots that can be fired before requiring a reload), but since it is fully automatic (holding the trigger down continues firing) the ammo is used much more quickly. Due to the nature of the gun and position of reload buttons this gun is harder to aim and reload. Finally, addition of these features will require more team communication and strategy.

Design

Concept

There are three main components to the laser tag system: vests, guns, respawn points, and a PC application. Each of these components is made up of many intricate parts.

The Vest.

The vests are made from a hunter's vest. Sensors are attached at multiple points of the vest in order to provide a target for other players. There is one senor on the chest, one on the back, and three on each shoulder. The three sensors on each shoulder are mounted in three different directions so that an opponent can register a tag on any side of the vest.

The vest also holds the microcontroller. It was found that the Arduino Duemilanove was the best microcontroller for this project. The Arduino is a bootloader and programming systems based on the ATMega microcontroller series. The ATMega microcontroller offers 14 digital input/output pins and six analog inputs. The Duemilanove is a development board designed by the Arduino organization, and provided the basis of the printed circuit board (PCB) used to interface all of the parts of the project.

The Gun.

The gun is the primary way the user interfaces with the system. There are two active interfaces with the user and two passive interfaces. The active interfaces are the trigger and a button mounted on the side of the gun to be used for reloading. The passive interfaces are a transmitter and receiver mounted to the gun.

When the user pulls the trigger on the gun, a signal is sent back to the microcontroller in the vest. The microcontroller then relays a signal to the transmitter on the gun that is received by a sensor on another player's vest or gun.

The reload button sends a signal back to the microcontroller to initiate the reload sequence. When the user runs out of ammunition, the microcontroller will not relay the signal to the transmitter when the trigger is pulled. Once the reload sequence is completed, the microcontroller will resume relaying signals to the transmitter upon a trigger pull.

The Respawn Point.

As discussed previously, the respawn point is an innovative addition to the game in order to prevent cheating. It is not a perfect solution, though. Unless deployed correctly, living players can exploit the respawn points by picking on players freshly respawned.

The respawn point is a transmitter, much like the transmitter on the gun, that is constantly transmitting a special code

The PC Application.

When a game is complete, the PC application tabulates the data of the game in order to decipher which team won. There is a USB port connected to the PCB that allows communication between the microcontroller and the PC application.

Integrating the Parts.

Figure 1 shows an overview of how all of the pieces fit together.

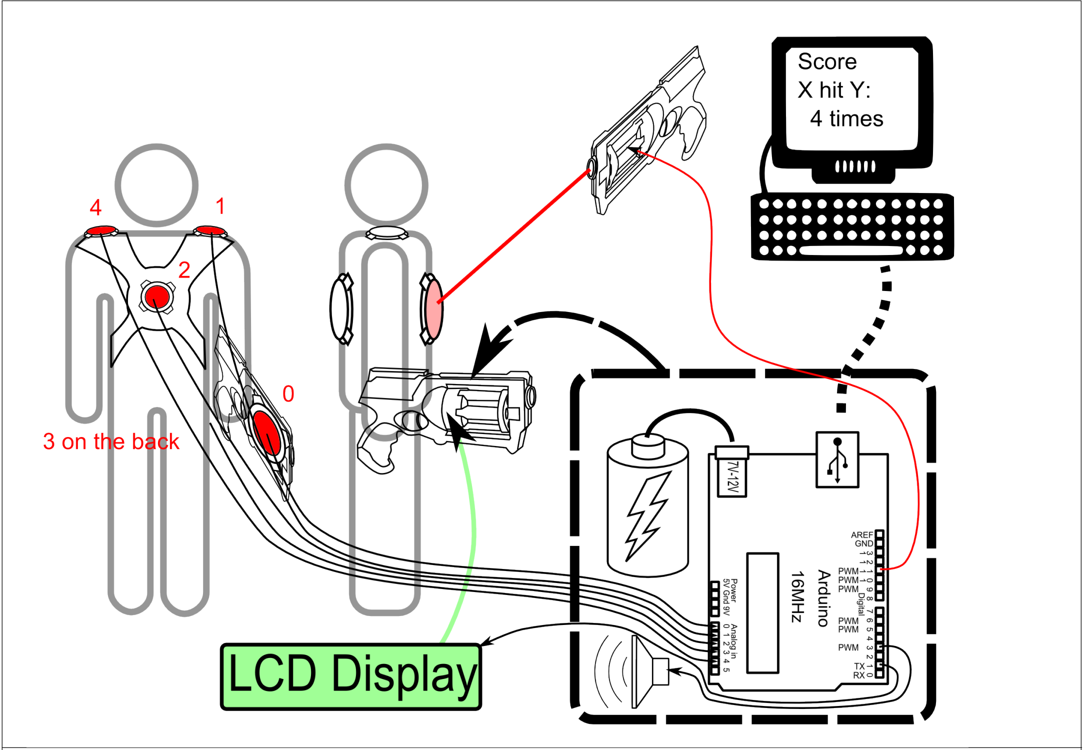


Figure 1: A diagram of the different parts of the laser tag project and how they work together

IR Data Transmission

Infrared communication is the facilitator of laser tag. It allows data to be transmitted from one player's gun to another player's vest. To receive data, the Sharp IR detecting unit, GP1UX51RK, was used. This part specifies that data must be modulated over a 38kHz signal, shown in Figure 2. The duty cycle, D_t, of the data must be below 40%. The data rate (bits per second) itself is flexible, but, as shown in Table 1, 500Hz was found, through experimentation, to be the most accurate rate. The microcontroller will use its timer interrupts to send data at a fraction of its clock speed to a 555 timer chip oscillating at 38kHz.

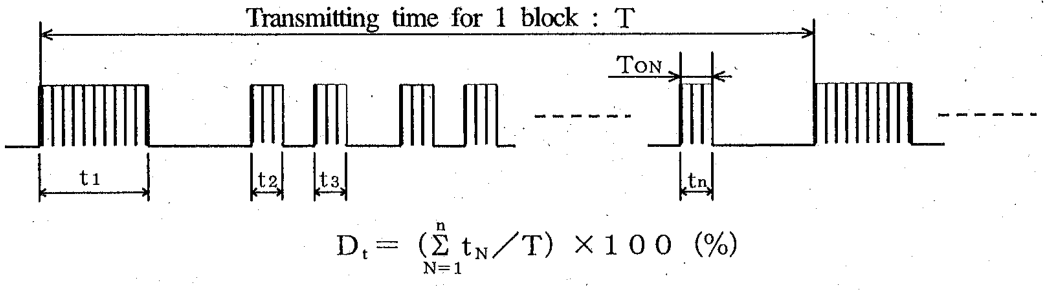


Figure 2: A visual representation of the timing scheme for the Sharp GP1UX51RK IR detecting unit

The detecting unit inverts the values it receives. For example, unit A sends 0010 and unit B receives 1101. This will be ignored in this report for sake of simplicity.

Most existing data packet protocols assume that a connection will stay open for as long as needed once it is obtained. Laser tag is a much more volatile environment. Data needs to be transferred in the flash of a light, so the communication protocol needs to be simple and robust. The packet scheme designed for this project is outlined in Figure 3. In order to increase confidence of receiving an entire signal, the data will be transmitted multiple times each time the player pulls the trigger. There is a total of 20 bits per packet, where eight bits are reserved for data. The start bit is zero, but the string of leading 1s also signifies the start of data. A parity bit is implemented to ensure some accuracy, but if more accuracy is needed, the receiver will wait for multiple transmissions of the same data. Eight bits is more than enough room to transmit who is shooting whom during the game. A transmission would be considered successful if the leading ones are all picked up, parity is met and leading ones of the next packet are identified; if this is not enough to ensure accuracy, the next packet will be read as a whole. There is also the option of additional error correction, but this would take more computation power and will only be used if repetition of data does not produce desired results.

$$\text{packet}(d) = 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ d_7\ d_6\ d_5\ d_4\ d_3\ d_2\ d_1\ d_0\ p$$

Figure 3: The data packet scheme to be transmitted from the transmitter

To keep the 40% overall duty cycle requirement, each bit is separated into three parts. The first part is high for a one, low for a zero; the second part is low for a one and high for a zero; the third part is always zero, see Figure 4. It can be observed that the overall duty cycle is always 33%, meeting the less than the 40% requirement.

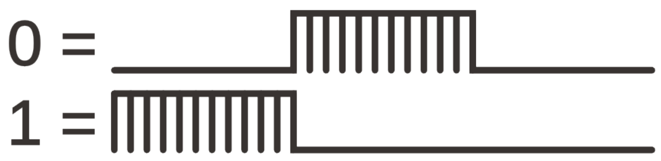


Figure 4: The composition of the bits in the transmission protocol

The leading ones give the micro a higher chance of sensing incoming data, but this comes with a cost: to keep the 40% duty cycle limit, a data packet of 0xFF is not allowed. As of now, byte transmission will be done with the ASCII data set, so 0xFF is not needed.

Duty Cycle	On time (μs)	Off time (μs)	Notes	Average On Voltage (V)	Average Off Voltage (V)
30	600	1400	reflectable signal	3.33	4.52
30	6,000	14,000	reflectable signal	3.18	4.52
30	60,000	140,000	reflectable signal	2.06	4.52
30	600,000	1,400,000	not reflectable	1.19	4.52
15	30,000	170,000	not reflectable	4.52 (off)	4.52
40	80,000	120,000	not reflectable	4.52 (off)	4.52

Table 1: Data showing different rates of data transmission. A larger difference between on and off voltage is better (at equal duty cycles)

Optics

A typical IR LED has a viewing angle of 20˚. When a signal is transmitted through an LED with this kind of viewing angle, the area that it covers increases as the distance from the LED grows. For the application as a transmitter for a laser tag system, this isn't acceptable. If an unmodified LED were used in a laser tag system, a target only a few feet away would be very easy to tag since the beam spreads so wide. Another problem with this spread is that the intensity of the LED drops off as the distance from the LED increases. Range is very important in a laser tag game. Targets range from only a few feet away to more than 15 yards away, and it is important that the signal is able to travel the required distances.

A simple, elegant solution to this problem is to use a double convex lens (also known as a converging lens) at the end of a tube to focus the light emitted from the LED. This solution is illustrated in Figure 4. The lens diameter and focal length need to be optimized to capture as much of the light emitted from the LED as possible. The ratio of focal length to diameter can be obtained with a trigonometric model of the system. Ultimately, the following ratio is found.

$$D = 0.353 * FL$$

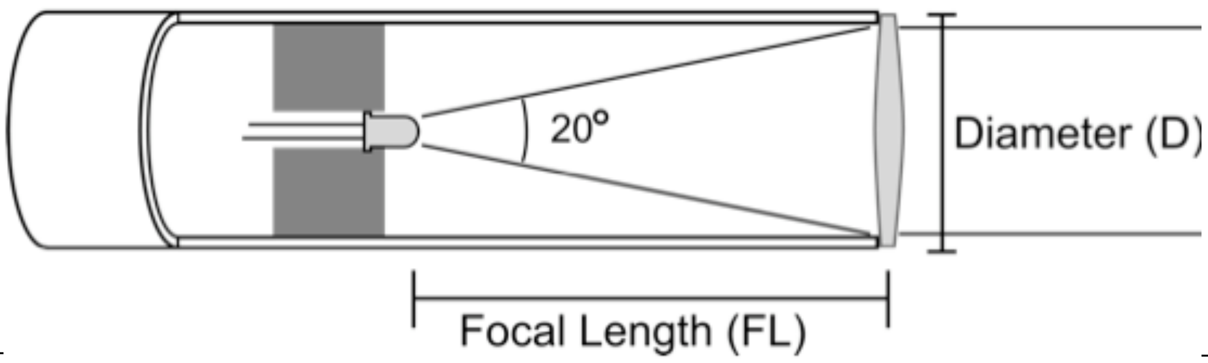


Figure 4: Diagram of transmitter tube optics

The website surplusshed.com was found to be a good source for optics. Some of the lenses offered by Surplus Shed are shown in Table 2. Another factor in choosing the proper lens size is the size of the tube encasing the transmitter. For durability and cheapness, the tube is a section of PVC pipe. The sizes of PVC pipe are very limited, and the lens must be able to easily fit into one of the sizes. In order to fit, the lens diameter must be smaller than the inner diameter of the pipe, but not by too much so that it is easy to affix the lens to the end of the tube. The size of PVC pipe chosen to house the transmitter has an inner diameter of 0.75 in (or 19.05 mm). The lens with a 19 mm diameter and 57 mm focal length was chosen to focus the light from the LED.

Diameter(mm)	Focal Length(mm)	Error(%)
19	57	5.6
19	58	7.2
19.1	66	18
20	54	-4.9

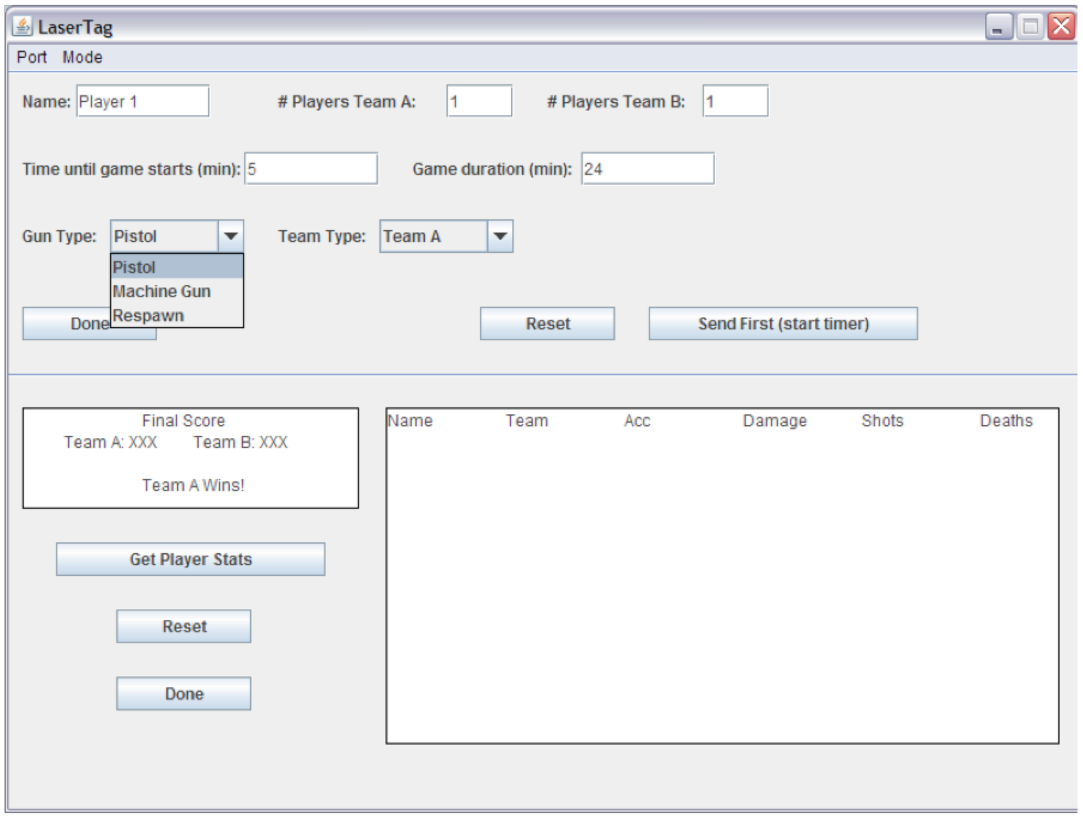
Table 2: List of available lenses.

PC Application

The PC application is the method to communicate to the microcontroller in each player's vest before and after the games. Figure 5 shows a screenshot of the PC application to be used by the moderator of the game. All pregame information is above the horizontal border, and the after game statistics are below. The PC application utilizes USB serial communication (Virtual COM Port) and follows our team's designed protocol: See Appendix A. This Protocol was designed and modified to fit our needs to relay the information needed to start the game and collect statistics afterwards.

Before a game, the PC needs to gather information as to how many players there will be and then send startup information including how long till the game starts, how long the game will last, the gun type and ID, and which team (and list of opponents) the player is on. The time until the game starts is important because once the first data has been sent, all players need to set up before this time expires in order to ensure all players start the game at the same time.

The best part of laser tag is comparing statistics at the end of the game with friends. This is the post game task of the PC application. Each player must again hook up via USB to communicate to the PC. The moderator simply selects the COM port from the drop down menu and presses the button to 'Get Player Stats'. The information that is then sent to the PC includes the gun ID of who owns the following stats, an array of who hit which sensor and the frequency of hits, and finally total shots fired and deaths during the game. Since the only way to store the damage on the microcontroller is to keep track of which player has hit one of its own sensors (as opposed to which sensors it has hit) the statistics will be continually updated as more players' statistics are collected. This means that the final winner, accuracy, and damage done statistics will not be determined until all players' stats have been collected.



Name: Player 1 # Players Team A: 1 # Players Team B: 1

Time until game starts (min): 5 Game duration (min): 24

Gun Type: Pistol Team Type: Team A

Final Score
Team A: XXX Team B: XXX

Team A Wins!

Name	Team	Acc	Damage	Shots	Deaths

Figure 5: Screenshot of PC Application

Final Product

Much of the design was created during the first semester. Getting the microcontroller programmed was the first milestone met. The optics tube was then completed. At the end of the first semester two microprocessors were able to send data from one to another and the overall system plan had been created.

The project was completed during the second semester. A PCB was created and manufactured early second semester based off of the Arduino Duemilanove development board with the project's needed additions, including a 555 timer, pull up resistors and ports to connect auxiliary equipment (gun and vest). The PC application was also written during the second semester. By the 12th week one unit (gun, vest, microprocessor) had been completed. The next couple weeks were used to replicate this package so that an actual game could be played between two people.

Overall, the project specifications were met and games of laser tag were played. Figure 6 shows one unit of the final product: vest, box with LCD screen facing the camera and machine gun.

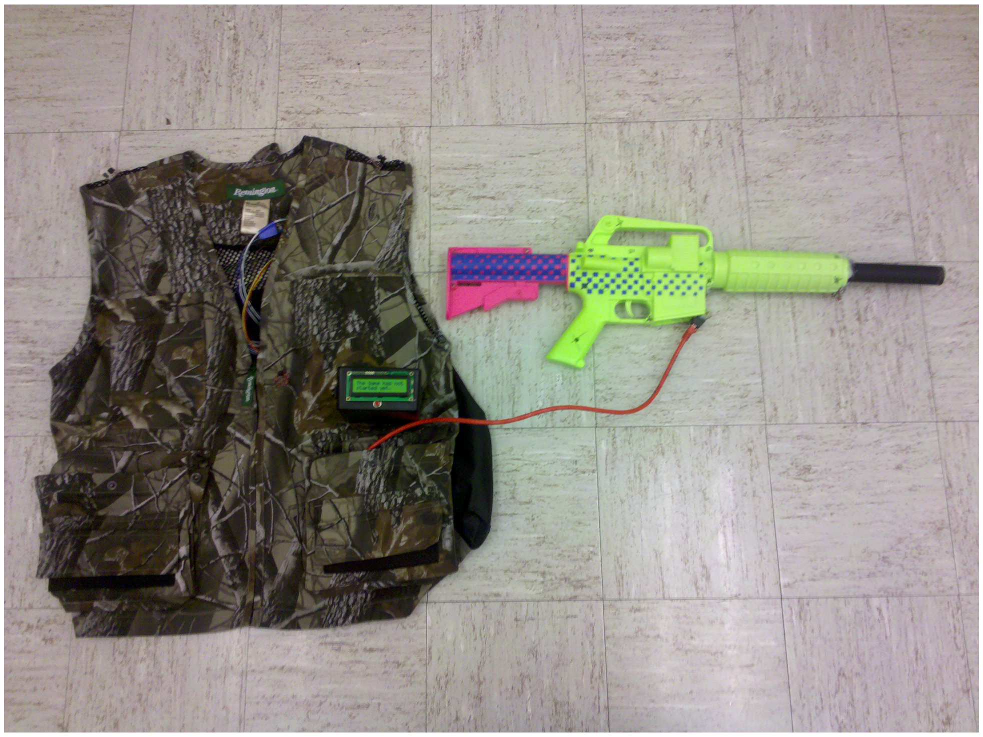


Figure 6: One complete user set

Step-by-Step Usage

Pre-Game.

Player A's and B's name are entered into the PC along with which weapon (pistol or automatic) they will use. Time until the game starts is specified, as well as how long the game will last. When the first player's unit is programmed, the countdown until game time starts and all other players' units must be programmed before the countdown reaches zero. This programming gives each player's microcontroller its own ID (to shoot) and a list of opponent IDs (so it is not possible for a player to be damaged by the player's own gun or team); the game time is also sent to the microcontrollers. Each of the players' LCD displays indicates how many seconds are left until the game starts.

In-Game.

Player A and B try to shoot the eachother's sensors to 'tag' them and score points. Each player's gun shoots a unique code, known by the other players. This way IR noise is ignored. When a beam falls on the receiver, the beam is stronger than any other IR noise in the area (except at extreme distances). A player's health and ammunition count are displayed on the LCD.

Every successful shot takes away a player's health, until there is no health left and that player dies. Each sensor has its own value of damage done. Thus getting shot in the gun may not hurt a player as much as getting shot in the chest. This difference is reflected in health lost, as well as damage points after the game. A speaker on each person buzzes a low tone when they are dead and the LCD display prompts them to find the respawn station.

When a player dies, they must find the respawn point. In a large game, this would allow this player to get away from the action for a bit and rejoin later without getting hit right away after coming back to life.

Every shot fired subtracts from a player's total ammunition. The process of reloading involves hitting the button on the gun (drop the empty magazine), hitting the button on the box (pick up the next magazine from your pocket/belt) and then pressing the reload button on the gun again (put the new magazine into the gun), in that order. These commands are listed on the LCD so players new to the game know what to do. This addition keeps players from continuously shooting and helps provide a more realistic game experience. Every shot fired also sounds the buzzer; a higher pitched beep is heard.

The game continues until the counters on the microprocessors reach zero (specified by the PC when they were programmed) and the LCD tells each player the game has ended.

Post-Game.

After the game, each microcontroller is connected to the central PC and the in-game statistics are uploaded. Since there is no wireless data transmission, it is impossible for a player's microcontroller to be aware when it has a successful hit, only the microcontroller belonging to the player who is hit knows about it. The PC puts all of this hit data together and computes the score for each player. The number of shots is also returned and accuracy is calculated. Individual deaths are recorded and one team is pronounced the winner.

Transmission Angle and Distance Data

The IR beam spreads over a distance, and because it also loses intensity, it is useful to know at which distance does a player have the highest chance of hitting his or her target. A string with distances marked attached to a protractor on a tripod was used to test this range using the pistol hardware. The pistol continuously shot the respawn code. A second microprocessor was programmed to light an LED whenever this code was seen (and to turn off the light if the code was not present). The data collected from this experiment is shown in Table 3. Figure 7 shows what this looks like in 2-D space.

The measurements show the trend that was expected. The beam started off very small (at the barrel) but then spread quickly. After about an arm's distance (three feet) the angle reduced significantly. Though the signal traveled over 20 feet from the transmitter, it was hard to measure the angles at that distance. The way the optics were constructed introduced a lot of potential error in the system, and more power could have been realized if the optics were more precise.

Distance (feet)	Angle (degrees)
2	8
3	8
4.5	7
6.75	5.5
10.13	3.5
15.19	2
22.78	Not detectable

Table 3: Data collected of distance versus angle

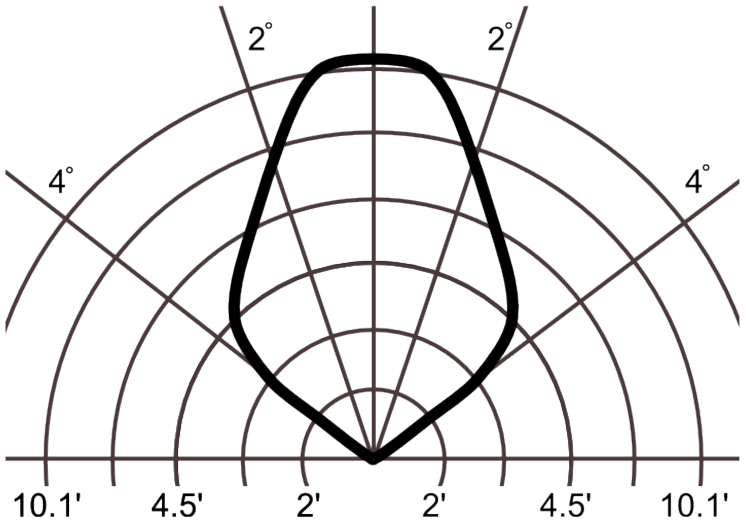


Figure 7: A 2-D model of the beam emitted from the transmission tube

Conclusion

As a game and engineering project, laser tag offered many options and areas for development and creativity. Further game elaboration leads to an increase of technical challenges to overcome while also yielding a more robust game style. Certain precision is required in both the field of optics and data communication. This project's distinctive data transmission needs produced a unique data packet scheme that used short bursts of unique patterns, followed by equally short slices of data, all repeated in quick succession. Experiments with optics were performed in order to reach a more accurate way of detecting a 'hit' in the course of this fast paced game. In the end it was almost too hard to hit somebody from across the room, not because the signal was too strong, but because the IR data beam was too narrow, but the game itself was very playable.

Rev 4

9 December 2009

Appendix A

Microcontroller to PC Protocol

Will send/receive 'characters' via serial interface (actually USB).

Each packet is 10 (ASCII) characters long and has a *header* and a *message*

Header is the first 4 characters (always starts with a ':')

Message is the last 6 characters

[:, h, h, h, m, m, m, m, m, m]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9] <- index

To start communication the PC will send three chars: 'G' '3' 4' and wait for the board to respond 'G' '3' '4'. NOTE: the board will be sending out chars constantly before the connection is made.

Then the PC will send a message (and wait for a response if one is requested).

Error handling:
NONE (use USB error protection/recovery)
If x amount of seconds passes, assume communication has stopped

NOTE: the micro can only handle 10 messages at a time, so wait X seconds between every 10 packets.

NOTE: most significant digit is leftmost (sent first)

Header	**Message Meaning**	**Message (bytes)**
G34 (without : to begin)	Start communication	(none)
43G	End communicatoin	(any 6 chars)
GUN	Player ID, Type of gun	[4,5] = ID * [6..9] = Type ^
TEM	Player ID on your team	[4,5] = ID
OPP	Player ID on opposing team	[4,5] = ID
Sms	Game starts in this amount of ms	[4..9] = time
Ems	Game length is this amount of ms	[4..9] = time
REQ	Request Data (after game)	(any 6 chars)
HIT	Player who shot you and info. Sent for every combination of enemies and sensors. (e·s times)	[4,5] = ID * [6,7] = Where hit (sensor #) [8,9] = Amount of hits
SHF	Shots fired by gun	[4..9] = amount of shots fired
DED	Deaths	[4..9] = amount of deaths

* Player ID's have to be non zero, zero means empty player slot.
^ Gun types: 'A' = automatic, 'P' = pistol, 'r' =respawn

Example: to say that the game should start in 14 minutes, the computer would send (14 minutes is 840,000 ms = 0xCD140 ms:

```
:Sms0CD140
```

Example: to say that you (micro) got hit 4 times in sensor port 11 by player 55:

```
:HIT370B04
```

Script

Before Game.

Micro sends garbage (actually display on LCD)

PC sends 'G' '3' '4'

Micro sends 'G' '3' '4'

PC waits for that response

PC sends: GUN, TEM, TEM, TEM..., OPP, OPP..., Sms, Ems, '4' '3' 'G' 'x' 'x' 'x' 'x' 'x' 'x'

After Game.

Micro sends garbage (actually display on LCD)

PC sends 'G' '3' '4'

Micro sends 'G' '3' '4'

PC sends REQ

Micro sends GUN (ID), HIT, HIT, HIT..., SHF, DED