# Remote Controlled Turret

Senior Design Project Spring 2008

RIT Computer Engineering Department
Josh Bookout – jmb2371@rit.edu
Kyle Howarth – kth7485@rit.edu
Mike Romero – mrr2732@rit.edu
Instructor: Dr. Roy Czernikowski

Submitted: May 5, 2008

## Table of Contents



## Table of Figures



## Table of Tables



## Appendices

# Project Description

The Remote Controlled Turret is a “foam” dart gun, capable of holding 6 rounds, controlled via the internet with a live webcam feed for aiming. Darts are propelled by bursts of air from a pneumatic firing system. The user must sign into a password protected web page to control the device. The user will be able to aim and fire foam darts from the turret. A webcam will allow the user to see where the turret is aimed. The user is also given the option of turning on super bright LEDs in low light conditions. The device uses an Arduino microcontroller to handle the motions of the servos and the firing system electronics. A “host” PC serves as a web server that will host the user interface webpage, broadcast streaming video from the webcam and interpret commands from the interface to relay them to the microcontroller. Figure 1 shows the physical construction of the turret.


**Figure 1: Completed Remote Controlled Turret**

# User Interface

## Client

The client is the main user interface for users. Upon accessing the client webpage, the user is presented with a login screen. The user needs to give a proper username and password to continue. Figure 2 shows the login screen.

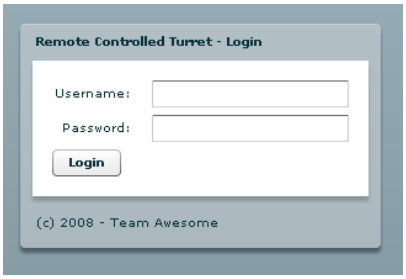


**Figure 2: Login Screen**

Upon providing the proper credentials, the user is logged in. The screen transitions to show controls for moving and firing the turret, see Figure 3.

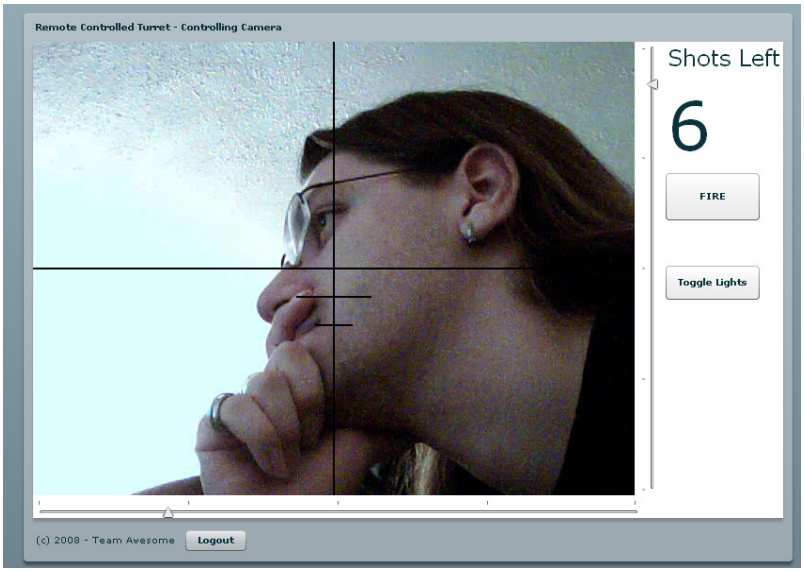


**Figure 3: Enabled User Interface**

As shown in Figure 3 the user can see where the turret is pointed with the live webcam feed. From within the 'enabled' login, or one with full controls, the user can use horizontal and vertical sliders to control the pan and tilt, respectively. The user can also click the 'Fire' button to fire a dart from the turret. The 'Shots Left' counter decrements after each shot. The toggle lights button turns the turret lights on and off. Finally, when a user is finished they can click the 'Logout' button to return to the login screen.

If another user is already logged in, the user is presented with ‘disabled’ interface that only displays the webcam feed as shown in Figure 4.

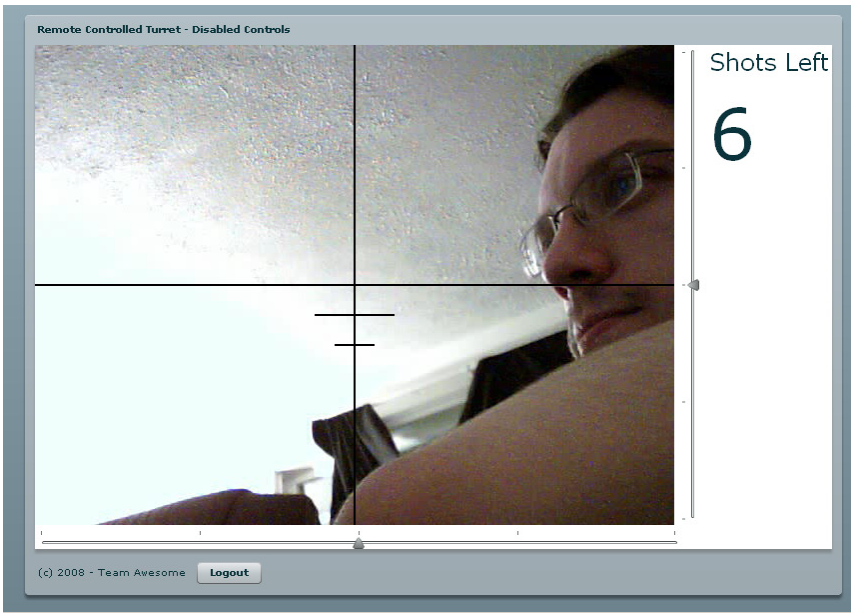


**Figure 4: Disabled User Interface**

.

Finally, when 5 minutes have passed, the user needs to confirm he or she is still active. An alert dialog will display with an ‘Ok’ button, if one minute passes without confirmation, the user is automatically logged out.

## Electrical and Mechanical Components

As shown in Figure 1 the Remote Controlled Turret consists of two major components, the turret itself and the air tank, more specifically the pneumatic firing system. The turret and all of its components are mounted on or in an aluminum base. These components are as follows.

The barrel and the servo to rotate it 360° are mounted on the pan and tilt system. Above the barrel and attached to it are the LEDs and the webcam see Figure 5.

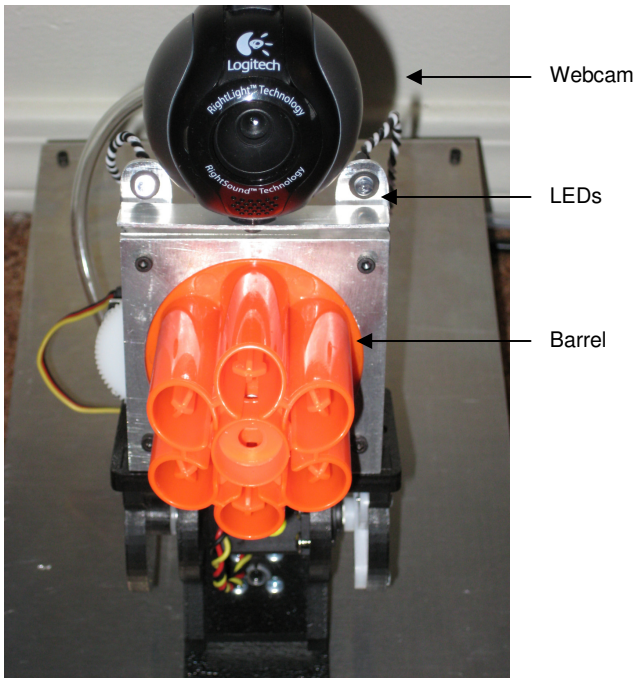


**Figure 5: Barrel, Webcam and Lights**

The pan and tilt system is mounted to the top of the base of the turret and is driven by two high torque servos, see Figure 6.

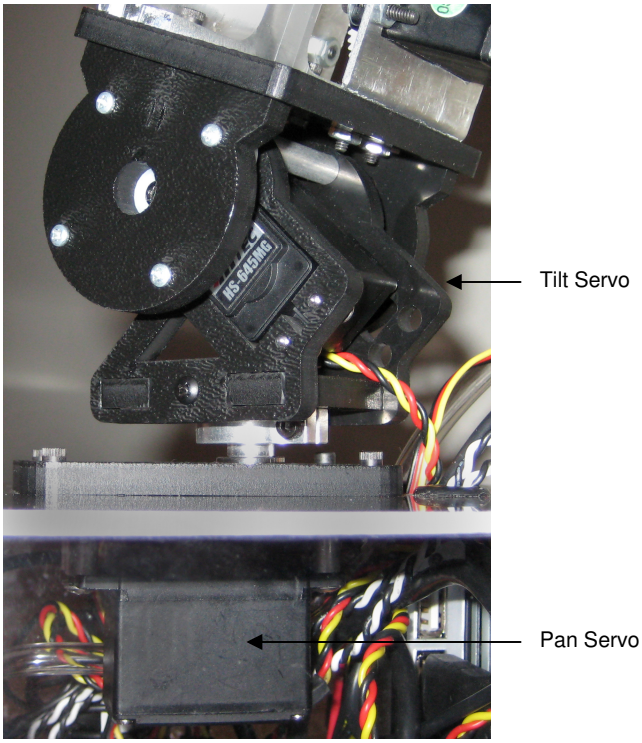


**Figure 6: Pan and Tilt System**

The darts are fired by a pulse of air from the air tank. The air supplied to the system passes from the tank through a pressure regulator and then to a solenoid. The Arduino microcontroller is used

to control both the solenoid and the servos. The solenoid and Arduino are mounted to the lower plate of the base of the turret. Also mounted to the lower plate are the power supply, a USB hub, and a circuit board. Figure 7 shows all of these components.

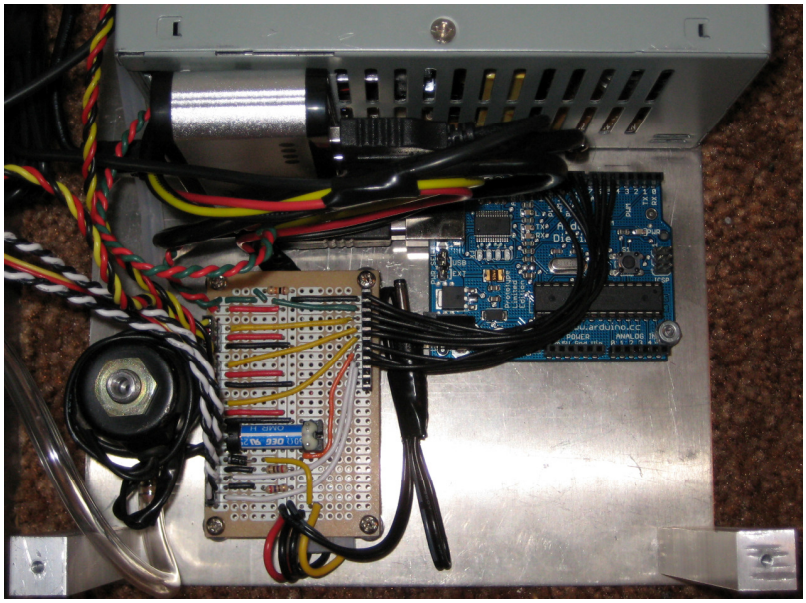

**Figure 7: Components mounted to Lower Base Plate**

The Arduino and the webcam are plugged into the USB hub, allowing a single USB cable (rather than two), the air supply tube and a power cord to be the only external connections of the device. The USB cable plugs into the host PC. On the back of the power supply, a power switch and a reload override switch have been added, Figure 8.

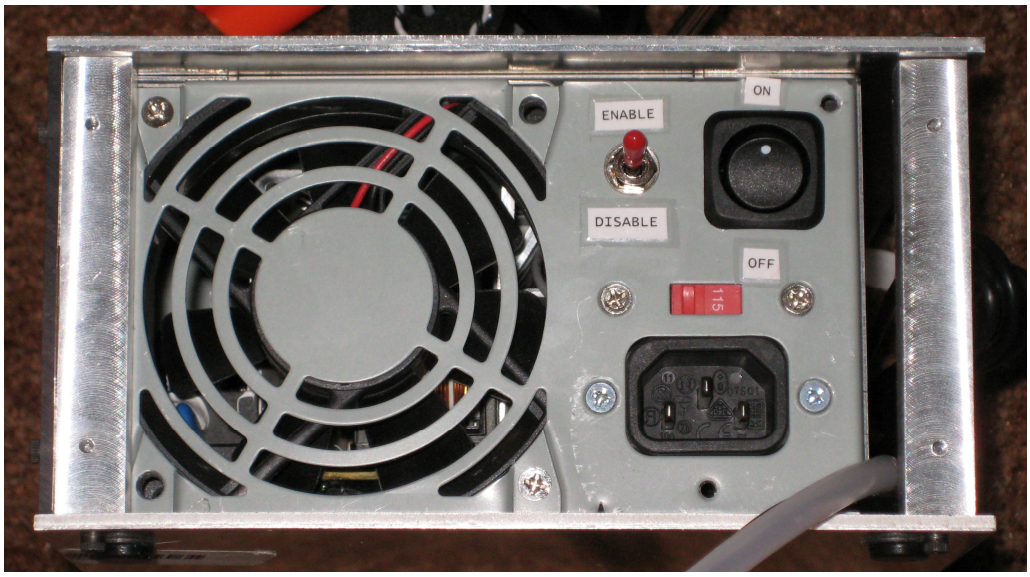


**Figure 8: Back of Turret, Power and Reload Switch**

## Microcontroller, Host and Client PC

An Arduino microcontroller is used to control the pan and tilt servos for positioning of the turret, the barrel rotation servo for the six-shooter barrel, and the relay for opening the solenoid valve used to fire a dart. The microcontroller also powers the LEDs used for lighting, and detects when overrides are set by the toggle switch on the back of the device. The microcontroller communicates serially over a USB port to the host PC. The host PC runs a serial proxy server to allow communication with the microcontroller over a TCP socket connection. The host PC also runs a media server to serve a webcam feed, and a web server to distribute the flash application and interpret commands used to control the device. The client PC can be any computer with an internet connection and a flash media player. The client may use a web browser to log into the host PC and control the positioning of the turret, view the webcam feed, toggle the lights on the system, and fire a dart.

# System Specifications

### Component Specifications

Most of the specifications for components in the system are related to the pneumatic firing system components and the servos. All of the pneumatic components used are rated for at least 125 PSI. The advised operating pressure for the turret is approximately 80 PSI. The system has not been test beyond 85 PSI and it is therefore unsafe to operate it at a higher pressure.
The servos used have two specifications (different for each servo) that must be met for the movement of the system to operate properly. The first specification to be met is operating angle. Each servo purchased was factory modified to have an operating angle of 180°. The tilt servo is software limited to a 90° range, to protect the hardware from contact wear. The other consideration taken for each servo is the operating torque. The pan and especially the tilt servos are rated at much higher torques than standard hobby servos. This extra torque is necessary to allow the servos to move the heavy load applied to the pan and tilt system. The pan servo was rated for 76 oz-in of torque and the tilt servo was rated for 133 oz-in.

### Testing, Performance, Analysis

The completed system has been tested for several performance metrics, number shots at operating pressure, firing range at operating pressure, max firing speed, and shot spread. The system was tested for number of shots and firing range in the following way. The turret was set on the ground, leaving the barrel about 8 inches above ground level. First several volleys of six shots were fired to determine the flat 0° range. This range averaged 6’ at 80 PSI. Second the turret was aimed straight forward (pan centered) and tilted 45° from horizontal. More than 30 volleys were fired in this position, see Figure 9.

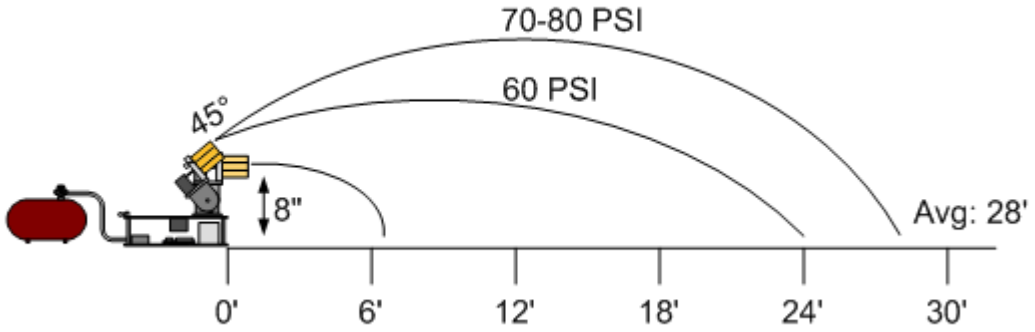


**Figure 9: Range Testing Setup**

The data collected is shown in Table 1.

| Max Distances per Volley | 80-70 PSI | | | |
|---|---|---|---|---|
| 1 | 38 | | 16 | 28 |
| 2 | 26 | | 17 | 30 |
| 3 | 24 | | 18 | 29 |
| 4 | 28 | | 19 | 28 |
| 5 | 28 | | 20 | 28 |
| 6 | 28 | | 21 | 29 |
| 7 | 29 | | 22 | 27 |
| 8 | 26 | | 23 | 27 |
| 9 | 25 | | 24 | 28 |
| 10 | 29 | | 25 | 28 |
| 11 | 29 | | 26 | 30 |
| 12 | 32 | | 27 | 26 |
| 13 | 29 | | 28 | 28 |
| 14 | 30 | | 29 | 26 |
| 15 | 28 | Average Distance: | 30 | 27 |
| | | | | 28.26667 |

**Table 1: Range Test Data**

This data shows that to discharge pressure from 80 PSI down to 70 PSI, 180 darts must be fired. The average maximum distance in this test was 28.26 feet. The same test setup was used with an operating pressure of 60 PSI, yielding an average maximum distance of 24 feet.

When sitting on a desk, the system averages a 1-2’ spread accuracy at 10’ distance. This accuracy depends greatly upon the system pressure. As the pressure in the system decreases the accuracy similarly decreases. Range and spread can also vary greatly based on which barrel is being fired and the particular dart fired. During testing it was plain that the 6$^{th}$ barrel had a much better range and spread accuracy, while other barrels were consistently shorter ranged or consistently outliers in the spread. This is due to the fact that the barrel is plastic (an inexact material) and the minor variations in alignment of barrel and air tube.

The minimum time to fire all 6 darts was 8.75 seconds, though this can vary based on network delay. Average response time of video and command latency also varies based on the network connection. On a LAN connection there is very little response delay. However over long distance 'high speed' internet connections one second and longer delays have been experienced.

# Implementation Details

Figure 10 shows the entire system overview. These components are described in detail in the following sections.

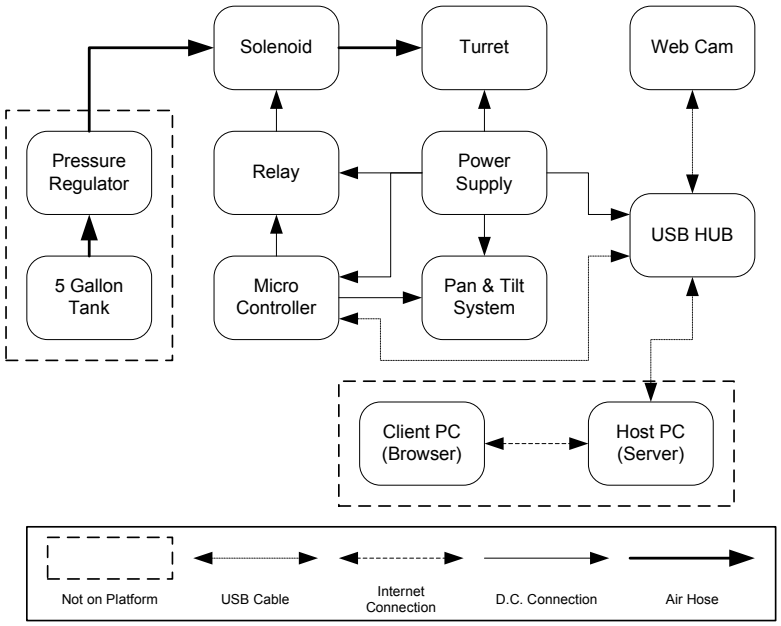


**Figure 10: System Overview**

## Software Implementation

### Red5

Red 5 is a media server that is used to broadcast the webcam feed, making it available to the flash based user interface. Red5 is only concerned with webcam broadcasting and streaming and has no bearing on any of the other implementation details.

### Flex

Flex is a SDK for developing Flash applications. Flex code can be either MXML (XML-based markup) or ActionScript. It compiles into a Flash SWF file. Flex is object-orientated so different views and objects can be reused and included in larger, more elaborate setups. The Flex is used for both the client and broadcaster applications. Implementing the webcam feed required the creation of a Broadcaster. The Broadcaster utilizes the NetStream and NetConnection objects to establish a connection to the server and start publishing a live stream.

In the client application there is a VideoPlayer component which is created upon login and finds the server IP address and connects to the live stream. Logins and commands are sent to services which are handled by a Ruby On Rails web server. To implement the login feature, the Flex

application takes the values from the text inputs and submits them to the AccountService on the web server and then waits for a response. If the response is NULL, then the user isn't authenticated. If the response is "disabled" then the user is in a disabled mode. If the response is the login name, then that user is logged in.
The rest of the functionality is handled with the TurretService. The TurretService deals with all things relating to the turret, such as moving and firing. Implementing the fire feature has a two-way communication link that goes all the way to the microcontroller, but the Flex application only worries about the response. The button disables itself until it gets a response back from the server, and the response is the number of shots left. This is then updated to the user interface. The moving and light toggling commands are always sent to the server, and the Flex application does not wait for a response. Overrides on the server side can make these commands.

### Ruby On Rails

The Ruby On Rails server is used to handle the communication between the microcontroller and the Flex application. There is software called WebORB that wraps Ruby and Rails objects to communicate easily with the Flex application. The server also uses a Ruby-to-Serial bridge called Serial Proxy to communicate with the microcontroller. WebORB uses two services to communicate with the Flex application. The Account service handles all things related to logging in and out of the system. The Turret service handles all things related to controlling the turret.

The Account service has two methods, login and logout. The service is responsible for keeping track of who is logged in and out of the system and determining whether the user interface provided to a logged in user has controls enabled.

The Turret a service has multiple methods dealing with moving, firing, getting the status of the microcontroller, and toggling lights. The firing method talks to the microcontroller to send a command and then receives the shots remaining. The status command works in the same fashion. The move command just sends to the microcontroller the direction it wishes it to move, it does not return anything.

### User Authentication

User authentication is handled inside of Ruby on Rails through a configuration file. The configuration file is laid out in username password pairs. The server, on startup, will load the file into memory and authenticate users based on what resides in memory. This means that the server must be restarted if the configuration changes. Authenticated logins will be logged into a database. Upon a successful logout, the logins for that person are removed from the database.

### Timeouts

Timeouts are handled in Flex to notify the client if they are about to be logged out. The ActionScript timer is set for 5 minutes and is reset after the user presses 'Ok'. On the Rails side, it will disable another username from getting control unless the last login occurred longer than 5 minutes. This is to prevent people that close the browser window without logging out from locking out other users.

## Microcontroller Implementation

The Arduino development platform was used for the microcontroller portion of the device. The microcontroller on the Arduino is the Atmel ATMega168, and is programmed using the Arduino programming language (based on C/C++). The Arduino was chosen because of its low cost, flexible power requirements, small size and ease of development. More information on the Arduino can be obtained from the official website: arduino.cc.

### PWM and Servos:

The Arduino controls the pan and tilt servos for positioning the turret, and the barrel servo for rotating the six-shooter barrel to the next dart. Although the Arduino has 6 PWM ports, the frequency of these ports by default is set to 30 kHz. This is unacceptable for servos which operate with a PWM frequency of approximately 50 Hz. Fortunately, there are servo libraries for the Arduino that are capable of providing the proper PWM signal to a servo on any of the 14 digital I/O pins available. In this project, two such libraries were used: the “Servo” library, and the “ServoTimer1” library. Both libraries implement the same methods and can be executed in the same fashion. A minimum pulse and a maximum pulse are set in microseconds for each pin to be used to control a servo. A value from 0 to 180 representing degrees can then be written to those pins, and the servo libraries will interpolate the angle to the proper PWM pulse. The key difference between the two libraries used is that the “Servo” library is capable of controlling all digital I/O pins, while the “ServoTimer1” library provides more accurate positioning using one of the onboard timers (pins 9 and 10 only). The “Servo” library is used to control the barrel servo, and the “ServoTimer1” library is used to control the pan and tilt servos.

### Safety and Overrides:

Safety of the system was carefully evaluated, since the device is designed to target and fire projectiles. To adhere to expected behavior of the system when it is initially powered up, the pan and tilt servo will center their positions, and the barrel servo will rotate the barrel to the starting position. The device will begin with the LED lights turned off, and the number of remaining shots will be reset to six (fully loaded). To avoid accidental firing, an override switch has been provided to allow a user to disable remote control of the device. The override switch is a red toggle switch located on the back of the device to prevent accidental toggling. When the system enters the override state, the servos adjust to center the pan and tilt system and rotate the barrel to its starting position. The white LED lights, if on, are automatically toggled off. This override switch is also used as a reload switch. When the device requires a reload, it is intended that the user toggle the override switch (into the ‘off’ position) so the user may safely reload darts into the six shooter barrel. Once the override switch is toggled to the ‘off’ position, the device will reset the number of available shots to six (fully reloaded). When the override switch is toggled to the ‘on’ position, normal operation of the turret resumes. From the client perspective, if the system is in override state, all commands such as ‘fire’ and ‘move’ from the client are ignored. When the system is taken out of override state, commands received from that point on shall be performed.

### Serial Protocol:

The Arduino communicates with the host PC via serial communication. The standard Arduino library provides the faculties for serial communication. The Arduino connects using a USB port, and an IC from FTDI allows the microcontroller to be accessed as if it were on a

COM port. All commands are communicated between the microcontroller and the host PC as ASCII characters. The host PC uses a program called serproxy which creates a server that allows communication with serial ports by opening corresponding TCP socket connections. This serial connection operates at a 19200 baud rate. The port number used to connect to serproxy may be specified in the user. This parameter is located in the "<rails_app>/config/configs.yml" file. Table 2 details the commands used in the system.

| Command | Description |
|---|---|
| "$MX_1X_2X_3Y_1Y_2Y_3$" | This command will be used to move the pan and tilt system to a new location. The $X_1X_2$ $X_3$ and $Y_1Y_2Y_3$ values are pairs of three ASCII characters that represent the absolute positions of the pan and tilt system in degrees, with the X parameter controlling the pan and the Y parameter controlling the tilt. Both the pan and the tilt values range between "000" and "180". The X and Y values are obtained from the slider bars on the user interface, which slide between "000" and "180". Note that since the tilt has only a 90 degree range of motion, the actual angle is half the $Y_1Y_2Y_3$ value. |
| "F" | This command will be used to indicate that the user wishes to fire a dart. No parameter will be provided. This command initiates the firing mechanism of the shooter, and once a shot is fired, rotate the six shooter barrel to the next dart tube. Once the microcontroller has fired, it will respond to the server with the number of shots remaining, passed as an ASCII number between 0 and 6. |
| "L" | This command is used to toggle the lights on the turret on and off. No additional parameters are provided, and the microcontroller makes no response to the host PC. |
| "S" | This command requests the number of shots remaining in the system. This is called when a user attempts to log into the host PC, and the number of available shots on the client must be updated. When the microcontroller receives this command, it will respond with number of shots remaining as an ASCII number between 0 and 6. |

**Table 2: Serial communication commands**

## Error Tolerance:

Due to the multiple network communication stages, the system is designed to be reasonably fault-tolerant. Occasional improperly formatted messages may be encountered due to network error. In this event, both the microcontroller and the serial interface on the host PC are designed to check the validity of the input, and gracefully disregard any garbage. Specifically, any commands received by the microcontroller are checked to determine if they are any of the 4 accepted commands from the above Table 2. On the host PC side, the "F" and "S" commands both expect return values from the microcontroller. If these return values are not within the range of expected values (numerical ASCII values between 0 and 6), then the number 0 is returned to allow operation to continue. If retrieving the remaining number of shots fails for some reason,

the remaining number of shots will be set to 0. However, the next call to retrieve the remaining number of shots will update the value correctly if a valid value is received.

### Task Control Block:

The following code describes the task control block and how each value in the TCB is used.

```
struct tcb {  // Structure for the Task Control Blocks
  byte firePin;              // Pin used to fire dart
  byte errorPin;             // Pin number of error LED
  byte overridePin;          // Pin numbers used to read override commands
  byte lightPinA;            // Pin used to toggle on and off LED lights
  byte lightPinB;            // Pin used to toggle on and off LED lights
  int round;                 // Current dart chamber (1-6)
  int barrelPos[6];          // Positioning for the barrel
  char status;               // Current status of the system (I, M, F, O)
  byte command;              // Command received from the host PC
  int panAngle;              // The numerical angle in degrees of the pan
  int tiltAngle;             // The numerical angle in degrees of the tilt
  int firePulse;             // Duration of pulse to open solenoid
  boolean error;             // Error status
  boolean light;             // Boolean indicating if the lights are on or off
};
```

## Mechanical Implementation

### Schematics and Drawings

For each of the custom designed components used for this device AutoCAD was used to design and render schematics with exact measurements. The components, the schematics developed, and the construction of the turret itself are discussed in this section.

The orange plastic barrel was taken from a spring powered foam dart gun. It has been modified to attach to an axel and gear. The barrel is mounted inside the barrel bracket, with the axel and gear protruding out the back. Protecting the axel and bracket from excessive wear is a small bearing, fitted to the bracket. All of these components, the barrel, axel, gear, bearing and bracket are shown in the CAD plots in Figure 11.

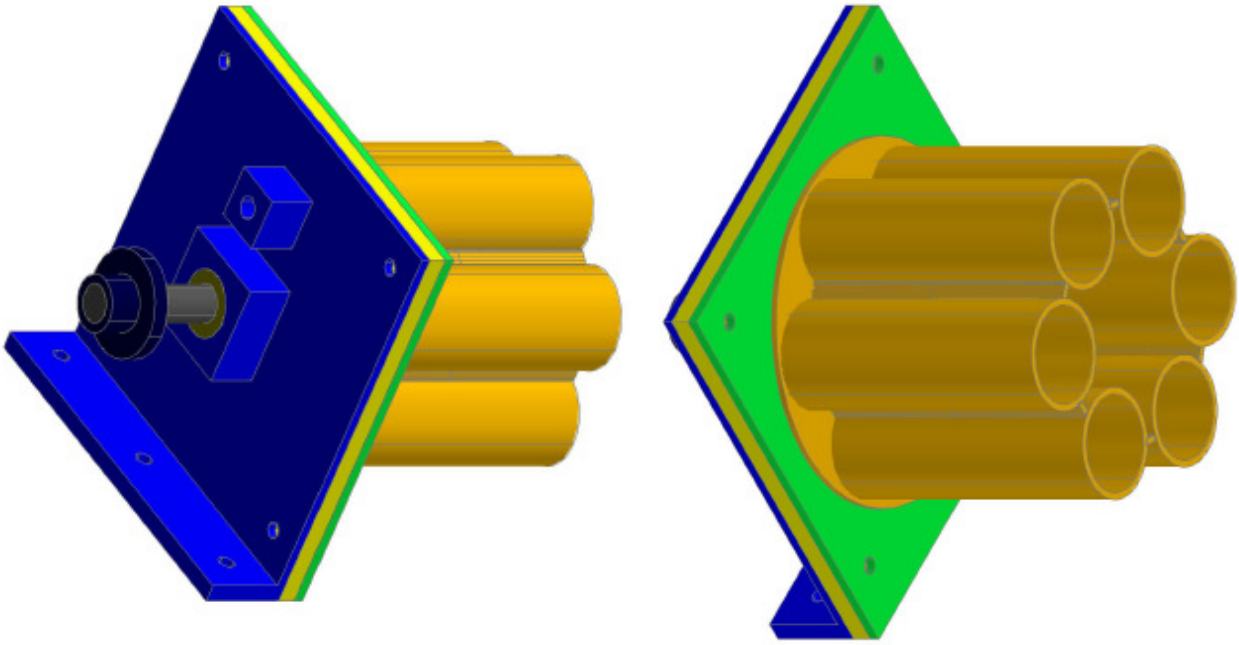

**Figure 11: Barrel, Bracket, Bearing, Axel and Gear CAD Renderings**

Figure 12 shows the actual machined parts shown in the CAD Drawings.

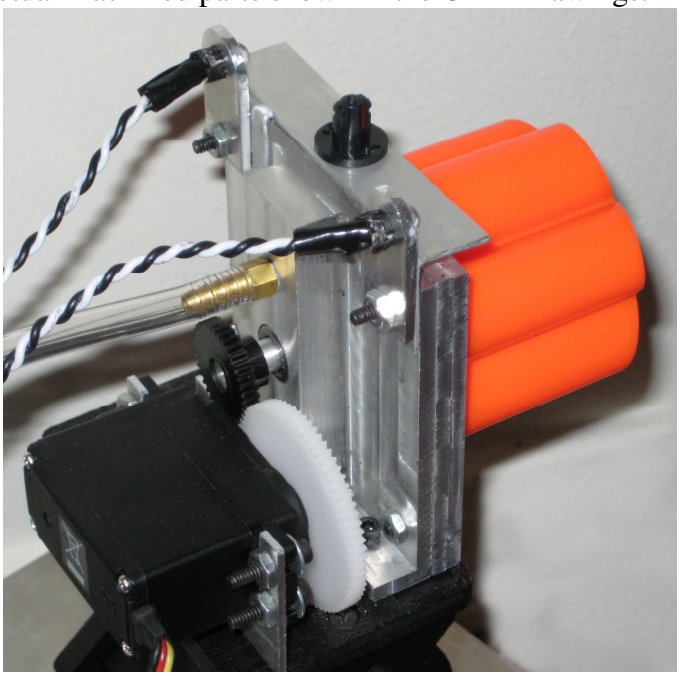

**Figure 12: Barrel, Bracket, Bearing, Axel, and Gear**

Also visible in Figure 12 are the brackets for the LEDs and the bracket for the Webcam. These simpler parts are constructed from aluminum, no CAD drawings were created. To rotate the barrel, a servo was mounted behind the barrel bracket, and a matching gear was fitted to the servo shaft. The gear on the servo is twice as large as the smaller one attached to the axel. This 2:1 ratio allows the 180° servo to act as a 360° servo so that the barrel can be rotated such that each of the six barrels lines up with the air tube (also visible in Figure 12).

The pan and tilt system (shown previously in Figure 6) was purchased from ServoCity.com dimensions of the system are shown in Figure 13.

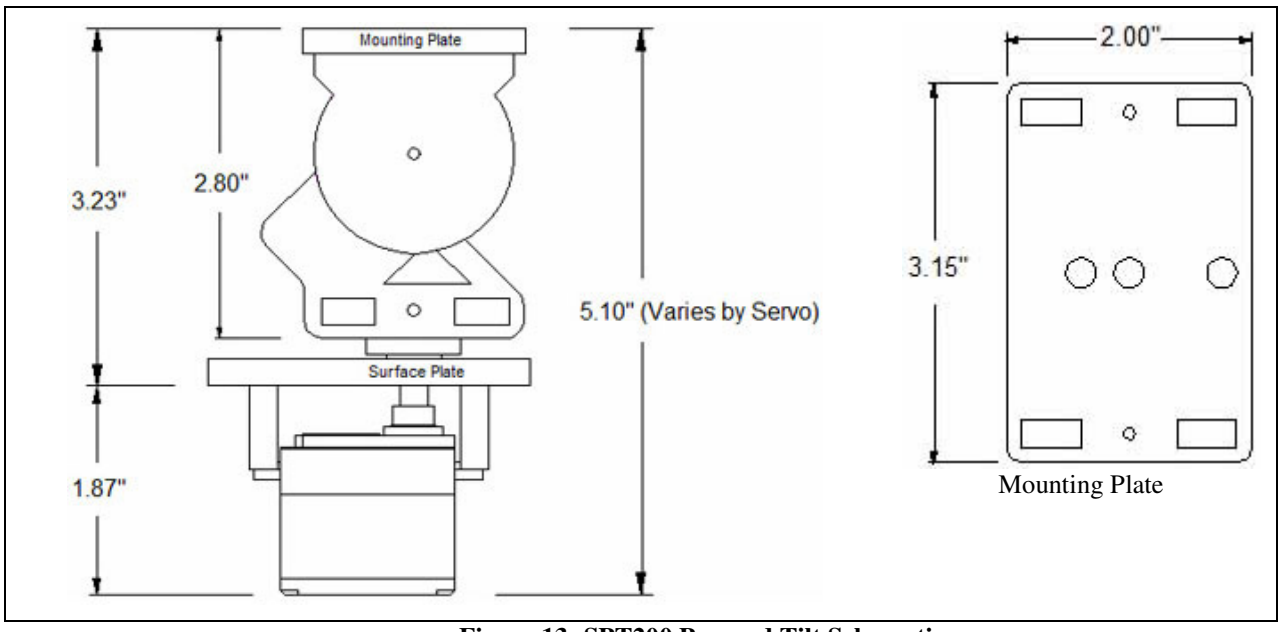


**Figure 13: SPT200 Pan and Tilt Schematic**

The base of the turret was designed in AutoCAD. The internal components, including the Arduino, solenoid power supply, and circuit board were modeled as well, to allow possible layouts to be shown before implementation. The final models and layout is shown in Figure 14.

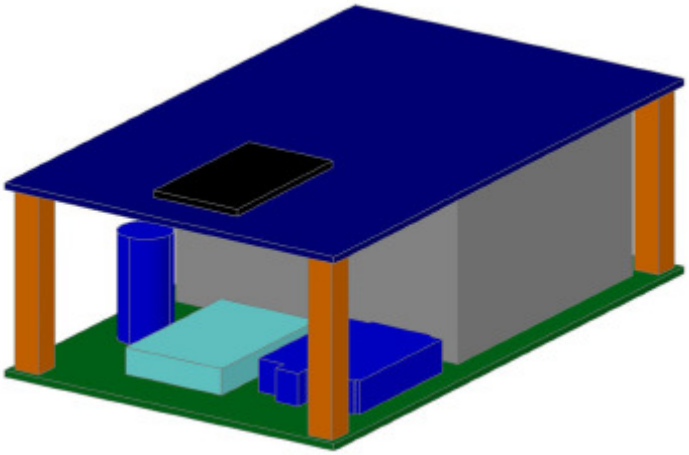

**Figure 14: Turret Base CAD Rendering**

The upper and lower plates, as well as the columns in the CAD plot were machined from aluminum. Figure 1 and Figure 7 on previous pages show the base and the lower plate with the components mounted to it. Exact dimensions of these components and those machined for the barrel bracket are available in Appendix III.

The electrical portion of the design is described by the circuit diagram in Figure 15.

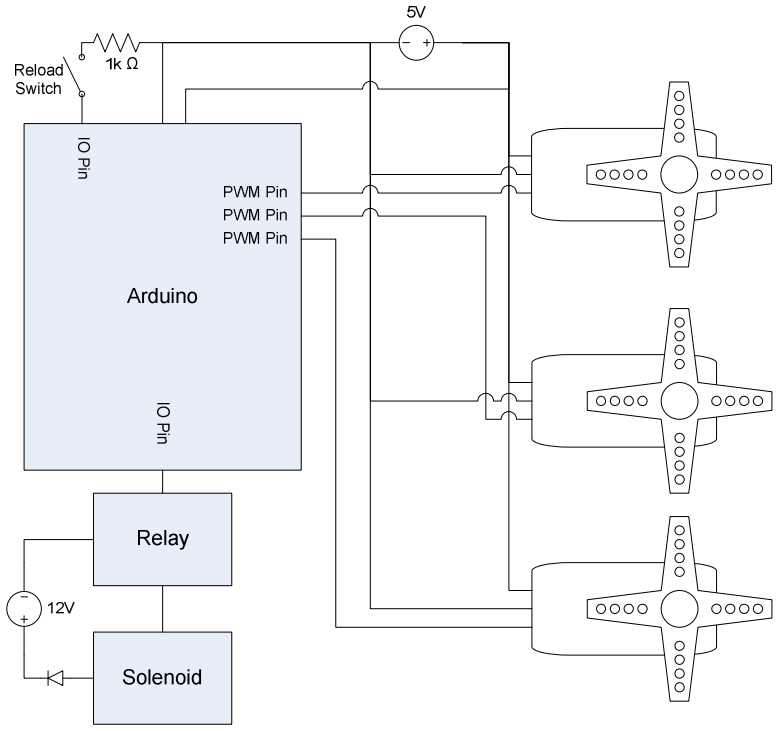


**Figure 15: Circuit Diagram**

# Manufacturability

## Software

All of the software used to develop and run the software for the turret is freely available online, and provided on the project CD submitted with this report. For specific requirements and the installation and execution procedure see the Installation and Execution section. Presumably in a commercial version of this project, custom software would be needed, for ease of design the prototype used convenient free software packages.

## Mechanical Components

Commercial prices and the actual price of the components used in the device are listed in Table 3. Components with higher actual prices contain shipping costs. Actual Prices listed as $0.00 were donated or the materials used were free from various RIT engineering departments.

| Part | Commercial Price | Actual Price |
|---|---|---|
| Toy Dart Gun (Barrel) | $5.00 | $5.00 |
| Pan and Tilt System | $50.00 | $50.00 |
| High Torque Servos | $60.00 | $60.00 |
| Barrel Servos | $20.00 | $20.00 |
| Gears and Bearing | $30.00 | $30.00 |
| Electronic Components | $18.00 | $18.00 |
| Air Tank | $25.00 | $25.00 |
| Pressure Regulator | $10.00 | $10.00 |
| Tubing and Fittings | $45.00 | $45.00 |
| Solenoid and Aluminum Stock | $60.00 | $0.00 |
| Arduino | $35.00 | $35.00 |
| USB Hub | $18.00 | $18.00 |
| Software | $0.00 | $0.00 |
| Webcam | $50.00 | $50.00 |
| Lexan | 10.00 | 10.00 |
| 30 Foam Darts | $8.00 | $8.00 |
| **Total:** | **$444.00** | **$384.00** |

**Table 3: Component Costs**

The aluminum custom components were machined to specification from aluminum stock in the RIT Mechanical Engineering Machine Shop.

While machining aluminum parts works well for a prototype, many improvements can be made to increase the manufacturability of this project. Molded plastic would be used in place of all aluminum components in a commercial version of this device. In a mass production setting, this device would not have servos or a large ATX power supply. These expensive components would be exchanged for cheaper DC motors, which in turn would require less power, possibly staying within USB .5 amp 5V requirements, eliminating the need for a power supply.

Other improvements for marketability include the use of smaller custom CMOS cameras which could be integrated more fully into the structure of the turret. All of the electronics, including the relay, USB Hub, and ATMega168 (Arduino microcontroller) can be placed on a single printed circuit board, reducing the large footprint of the electronics in the base.

If a similar pneumatic system is desired in a commercial product, smaller air canisters rated for higher pressures are easily adapted to this system. Use of a regulator would still be required on a higher pressure system for safety. In the existing commercial foam projectile products small motors are used to drive air pumps and mechanical spring and piston designs are used to propel darts. These designs, while cheaper to mass produce, require higher design complexity than the simpler pneumatic system used in this prototype.

# Installation and Execution

The installation procedures for the remote controlled turret are outlined in a readme file provided with the software package. The readme will be included in this report as an appendix. The system requirements to operate the device are minimal but include the following:

- A PC with internet connection and powered USB 2.0 port
- Java 6 runtime
- Firefox, IE, etc with adobe flash player installed
- Windows Vista/XP SP2 or greater

Note that although the official supported operating system for the host computer is Windows Vista/XP, all software technologies used are cross platform, and with proper configuration it is possible to install the host PC software on non-Windows systems.

The exact installation instructions are covered in the installation readme included as an appendix, and shall not be covered here. The basic installation sequence however is as follows:

1) Install Red5
2) Install Ruby and related software
3) Install drivers for the webcam and microcontroller
4) Install sqlite3
5) Modify configuration files
6) Execute provided batch file
7) Broadcast the webcam feed
8) Log in on client machine

Configuration files have been designed to allow the user a comfortable level of control over the system without exposing too many underlying details. A user will be able to define the IP address of the host machine, the port on which to connect to serproxy (and therefore which COM port to access under Windows) and define various username and password combinations. The usernames may be set in the users.yml file located in the <rails_root>/config/ directory, and the IP address and port number may be specified in the configs.yml file in the same directory.

# Comments and Difficulties

Certain portions of the design and implementation of this project were more problematic than others. One of the major hardware issues faced towards completion of the project was a problem with the barrel rotation. While most of the custom parts built for the turret followed exact specifications, the barrel used was too complex to machine. The plastic barrel from a spring operated dart gun was chosen to alleviate this complexity. The plastic piece however is not an exact piece. Many adjustments to this piece were required to achieve smooth rotation. The smooth rotation was necessary for the low torque barrel servo to be capable of turning the barrel.

There were two major software issues faced, one pertaining to the webcam feed, and the other dealing with serial communication. Several approaches were discovered to achieve serial communication, however after implementing the serial protocol the Arduino would under some conditions face un expected resets, this would happen most often during serial communication from the Arduino to the host PC. The result of the unexpected reset was a stalled receive on the

host PC when the Arduino failed to send terminating characters. The solution was to check individual characters, determining if garbage characters were read and returning from them receive a good result even if bad messages were received. This problem was difficult to diagnose because it was difficult to reproduce.

The webcam feed sometimes stalls on the client application while remaining constant on the server side. This error is also difficult to reproduce, and the reason for it occurring was never fully uncovered. Currently the only fix is to reset the server, rebroadcast the feed, and login again from the client.

## Summary

The result of the project is a working prototype system capable of all the specified operations. The system is capable of firing darts up to 28 feet, with accuracy, at 10 feet, of approximately 1-2 feet. The system can be controlled from any client with access to the host machine via a high speed internet connection. The system displays a live webcam feed from the turret. The user is capable of pan and tilt control, firing, and light control. The system properly distinguishes between approved username and password pairs. The system appropriately allots control of the device on a first come first served basis. The system implements a timeout for inactive users. The system has passed through rigorous testing and documentation of installation and setup procedures have been verified in a clean install environment.