

# Automated Targeting Proximity Turret

Hector Colon, Adam Horton, Kyle Steighner,  
Nicholas Yielding

School of Electrical Engineering and Computer  
Science, University of Central Florida, Orlando,  
Florida, 32816-2450

**Abstract** — The application of vision acquisition and response is essential in every aspect of life. What our group will be utilizing this application for is just that. Our system will utilize multiple visual inputs and respond accordingly to how we have programed the microcontroller. Our system is focused around military application with an airsoft rifle as the simulated weapon. The way in which our system will acquire the target(s) and respond accordingly was a difficult task. We will have digital webcams that will capture what is in front of our system send that data to computer for processing and from there the data will go through the microcontroller to maneuver the servos that we have attached to our mechanical base.

**Index Terms** — Automation, computer vision, laser radar, military, network servers, object detection, Robots.

## I. INTRODUCTION

The system that we have designed and implemented has been funded solely on our own group member's budgets. The system that we have designed will first acknowledge that a subject has entered the field of view (FOV) of the web cameras that we have implemented into our design. After a subject has entered the FOV the system turret will continuously be monitoring the distance and location of the subject from the base. Once the subject has come within a distance of the turret the audible alarm that we have implemented will sound warning the subject that they will be fired upon if they do not immediately exit the area. Every account that the turret comes into contact with a subject will be documented on our server. Also with the design of our system we have implemented a manual control operation if or when a user would want to use the turret for their own operation.

## II. MECHANICAL DESIGN

The mechanical design and layout of any system is incredibly important. With our system there is no exception. We have constructed our turret's base out of wood for the simplicity of corrections and alterations. This might have not been as lightweight as aluminum or other metals but our budget was very tight for this component. We have taken two square pieces of wood and one will serve as the mounting into the ground with stakes. The other has been trimmed down and altered to serve as the mounting of the base arms and ball bearing turntable. The two wood arms that are coming up from the base have had a hole machined through them to allow for an aluminum tube to be slid in. This will serve as the connection of the airsoft rifle to maneuvering servo. The trimmed down piece of wood that was spoken about earlier will serve as the connection to the other servo that is on the other side of the wood.

The airsoft rifle is attached by a 5/8" Aluminum tube and 1/4" threaded rod that was modified to form a similar image to a U clamp. The modified threaded rod was fed through nylon clamps that were placed around the aluminum tube. The gearing ratio for the tilt motion is 2:1 which in turn will produce more torque put less range of motion. We incorporated stainless steel machined bushings on the tilt motion of which the aluminum tubing will maneuver on so that we have a tight fit and avoid aluminum to aluminum contact.



Fig. 1. Airsoft gun mounted on to our rotating base. The rod going through the top allows the gun to also tilt upward and downward.

### III. ELECTRICAL HARDWARE COMPONENTS

The electrical hardware will serve the purpose of translating the software into signals for the motors and sensors which will in turn become mechanical movement of the turret.

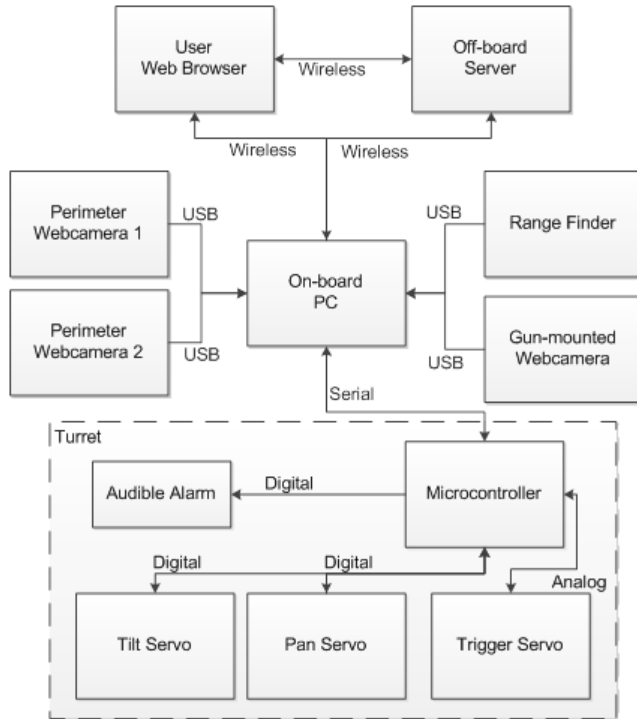


FIG. 2. The turret block diagram shows the relationship between the offboard server, the onboard PC, and the electrical components of the auto-turret.

#### A. Motors

The servos that we are using on our turret are the HS-5745MG Digital ¼ Scale and the HS-485HB Analog servo. The HS-485HB will be used for the firing of the trigger and the HS-5745MG Digital ¼ Scale will be used for the Pan and Tilt. The HS-485HB produces 83.3 oz-in of torque and has an angle movement of 90 degrees. This motor requires 6 volts of power and has a current draw of 180 mA. The HS-5745MG Digital ¼ scale produces 250 oz-in of torque and has an angle movement of 180 degrees. This motor requires 6 volts of power and has a current draw of 840 mA.

#### B. Rangefinder

Rangefinder selection presented a unique problem for a component in our project. We required a rangefinder that could give continuous measurement accurately for a range of at least 75 feet. Also, it must be inexpensive to fit with

our limited, non-sponsored budget. Usual means such as IR sensors are inexpensive but greatly lack the range we require, and LIDAR sensors meet all of our requirements, but are far too expensive. As a middle ground, we chose a single point laser rangefinder, the Fluke 411D. Below are the specifications.

The Fluke 411D is a construction tool, so there needs to be a way to interface it to the computer. For this we found a control board that is designed especially for use with the 411D. It solders in place of the control pad and the LCD screen, and allows continuous measurement mode to be used, and all of the information to be sent to the computer. The device can be attached through USB to the PC, and interacts as a human interface device (HID).

Technical specifications	Fluke 411D
Range (for extended distances, use a target plate)	0.1 m to 30 m (0.33 ft to 100 ft)
Measuring accuracy**	± 3 mm (0.118 in)
Units displayed	00.000 m, 000 ft 00 in 1/8, 000.00 ft
Laser class	II
Laser type	635 nm, < 1 mW
Automatic power off	after 180 seconds
Continuous measurement	•
Addition/subtraction	•
Battery life	up to 3,000 measurements
LCD illumination	–
Data locations	–
Min/Max	–
Audible feedback	–
Pythagoras (Indirect measurement)	Simple
Ingress protection	IP40
Dimensions	123 mm x 50 mm x 26 mm (4.84 in x 1.97 in x 1.02 in)
Weight	150 g (5.29 oz)
Temperature range	
Storage	-25 °C to 70 °C (-13 °F to 158 °F)
Operation	0 °C to 40 °C (32 °F to 104 °F)
Operating altitude (ISO 9002)	up to 3500 m
Storage humidity (at 35 °C)	maximum 85 % for 24 h
Batteries	AAA (2)

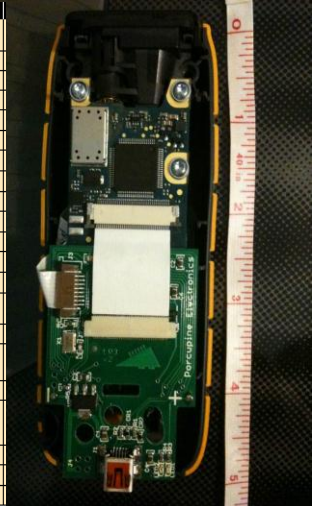


FIG. 3. The rangefinder spec. sheet can be seen on the left. On the right is a photo of the rangefinder control board installed.

#### C. Audible Warning Alarm

For the warning alarm we have used the Floyd Bell Audiolarm II, XC-09-330-S model. This alarm operates as a 2900Hz frequency, with a volume of 82dB at 2ft. The operating voltage is 3-30Vdc which allows us to control it using the digital output pin on the microcontroller, which outputs a range of 4-5Vdc. The alarm produces a continuous sound, but can be changed to a siren effect with control from the microcontroller.

#### D. Cameras

For the cameras, 3 commercial off the shelf webcams will be used that have the least grain possible, and are the least costly or already in possession of the group members.

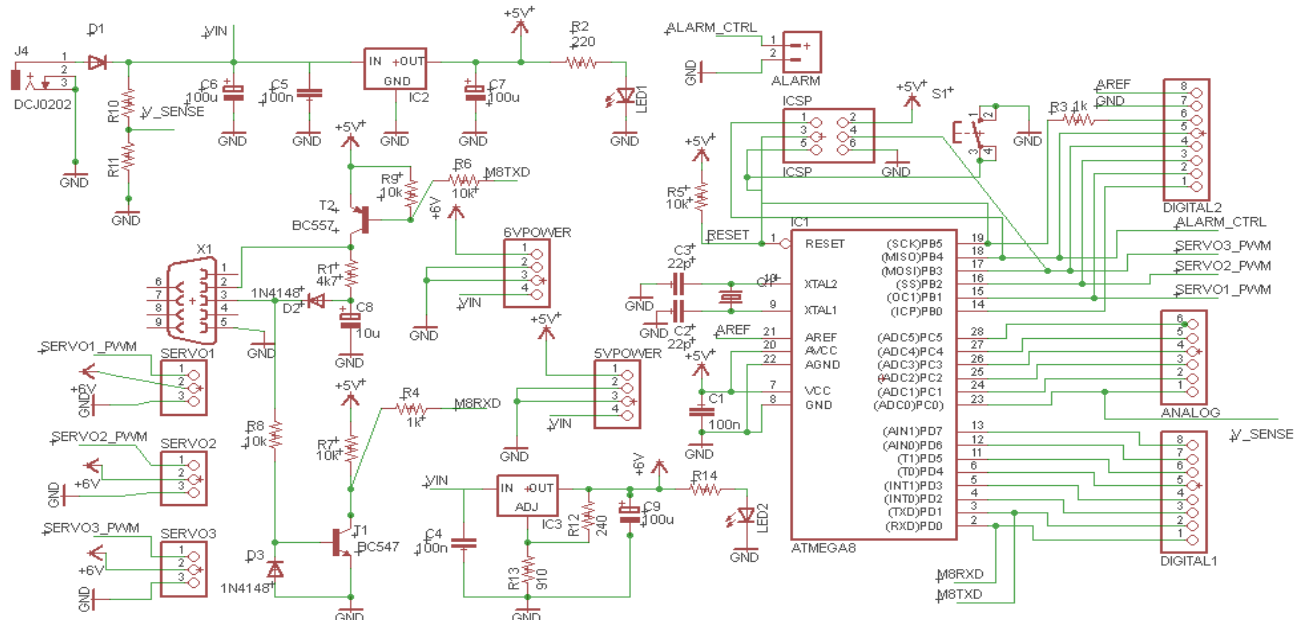


FIG. 4. The final design schematic of the PCB

### E. Computer

For the computers we are designing the software to run on the minimal spec possible. The goal is to run on a netbook, but it may require a full sized laptop to run the software smoothly. Any computer will work with it.

### F. Microcontroller

In order to control the servo motors and other sensors we will use a microcontroller. For our microcontroller chip, we chose to use an Atmel Atmega328. The 328 is an 8 bit microcontroller that features 32Kbytes of programmable flash, 1Kbytes of EEPROM, and 2Kbytes of SRAM. There are 6 analog input lines, as well as 14 digital I/O lines. 6 of these digital lines can be used for PWM for use with out servo motors. The Atmel chip can be programmed with the Arduino boot loader, making it easier to interface the software with the hardware. [2]

## IV. PRINTED CIRCUIT BOARD DESIGN

The printed circuit board design is important because a premade microcontroller circuit will not handle the currents from the sensors and motors we are using. The servo motors run at optimal torque and speed at 6 volts. Also, the motors have a current draw of 800 mA each even when not under load, so a higher current draw will be needed than the 1.5A line that is used on an Arduino.

Also, we wanted the motors and sensors to cleanly attach to the board, with connectors, instead of just wires connecting them in an unorganized fashion.

### A. Schematic

To meet these specifications, there are separate power traces for the board power (5 volts) and the motor power (6 volts). The board power line can draw 1.5A while the motor power line can draw 5A. Also, breakout headers are used for the three servo motors and the alarm. The microcontroller chip communicates with the computer via a DB9 serial connector. Below is the schematic for the printed circuit board.

### B. Layout

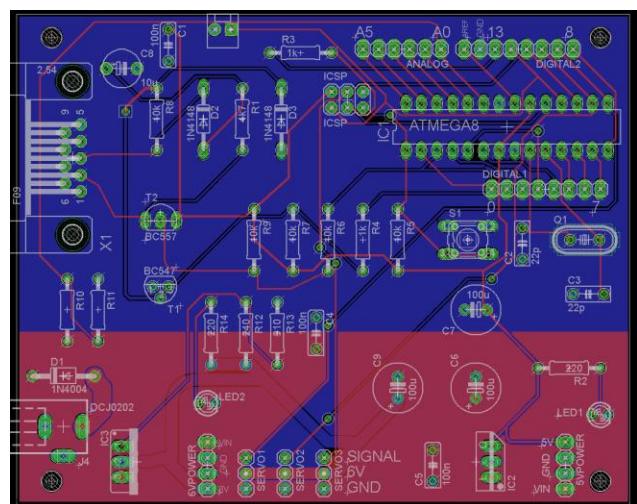


FIG. 5. Layout view of our PCB showing both layers.

For the layout of the printed circuit board, we used the free version of Eagle. Because of this, the size of our

board was restricted to 4 x 3.2 inches. We found this to be plenty of space for our schematic design, and chose to use all through hole parts so that the board could be assembled by hand quickly. Because of the possibility of 5-7A current in certain traces on the board, space became a problem in certain sections. To remedy this, a power plane was used, as well as a ground plane. For developmental purposes, all of the microcontroller pins have pin outs, as well as the pin outs for the ICSP if programming via the DB9 port fails. Eagle's auto-route function was used for most of the traces, with the remaining traces routed manually. Below is the layout of the board.

## V. ONBOARD SOFTWARE

The onboard software is responsible for motion detection, target tracking, and controlling the microcontroller. The software was written in C# using Visual Studio as our IDE. The software allows for onboard manual control as well as automatic controls.

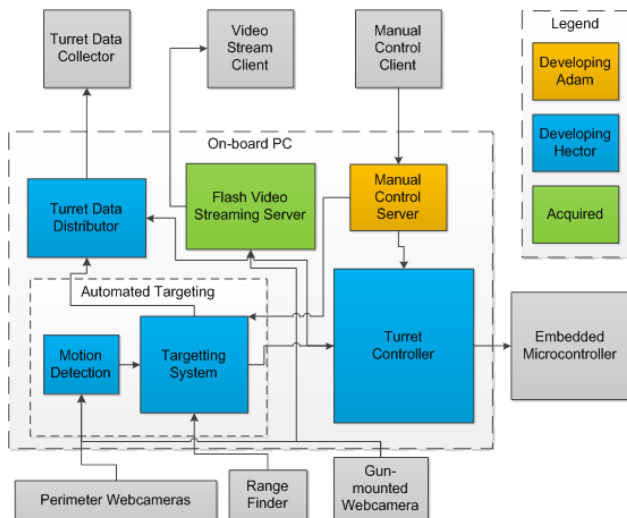


FIG. 6. Software block diagram

As you can see in the block diagram above, the software takes input from the webcams and the range finder and processes them in the motion detection and targeting system. This data is fed into the online target data collector as well as the turret controller which moves and fires the turret.

### A. Manual Controls

To use the manual controls, one must first click the Manual button in the main software window. Buttons are available for moving the turret up, down, left, right, and to fire the trigger. The gun mounted camera's vision is

displayed in the main camera window. In this mode, no motion detection or target tracking is performed.

### B. Automatic Controls

For automatic controls, one just has to click the Automatic button below the manual controls. The AForge.NET [1] computer vision library was used to program the motion detection and target tracking of the automatic controls. AForge.NET allows us to easily take the visual input from the cameras and analyze them for motion detection. The software uses a series of initial frames to compare to subsequent frames while slowly updating the initial frames as time goes on. Using the motion detection, we can identify potential targets and pick which target to track. The turret prioritizes targets based on their relative size and distance. With the laser range finder, we can also determine the distance of the target from the turret and when appropriate, fire the weapon.

### C. Turret Controller

A separate static class, TurretController is responsible for sending instructions over the serial port to control the movements of the camera. Its methods are available and used by both the manual and automatic controls. The TurretController class includes methods to pan and tilt the camera a predefined amount (20 degrees), or any value given by the automatic controls. The TurretController class also instructs the trigger servo to fire the mounted airsoft gun. For safety, it also includes a reset and emergency stop methods which respectively return the turret to its original state and position and stop all movement and firing of the turret.

### D. Microcontroller

The microcontroller code is separate from the rest of the onboard software. We are using an Arduino microcontroller and its software was written in the Arduino IDE using its native language. The microcontroller code takes input via the serial port and translates it into commands for the turret. All communication is done by 5 packets of 1 byte each. Each packet is read by the microcontroller in series and can contain an instruction for the turret. Once the instruction has been translated, the microcontroller sends the appropriate command to the turret. The microcontroller also keeps track of the turret's current status and position.



### E. Client-Side Software

In addition to sending data to the microcontroller, the turret software is responsible for relaying data back to the server. This is done using UDP and TCP socket connections to the server. A TCP socket connection is made with a TCP server running on the server computer, where all initial set-up data including the initial camera screenshot are sent across followed by any engagements that occur afterwards. After connection with the TCP server is obtained the turret starts to broadcast a UDP socket transmission of the current turret state including the pan and tilt of the turret and any targets that it is currently tracking. This data is used both by the server's data collection and the manual control page for displaying up-to-date radar.

## VI. OFFBOARD SOFTWARE

The off-board server is designed to collect data from the turrets and supply a web-application called the Turret Command Center (TCC) for end-users. The web-application includes a collection of pages for viewing the connected turrets, the targets that were engaged by each turret, and the option of taking manual control of a turret with a video stream from the gun-mounted camera.

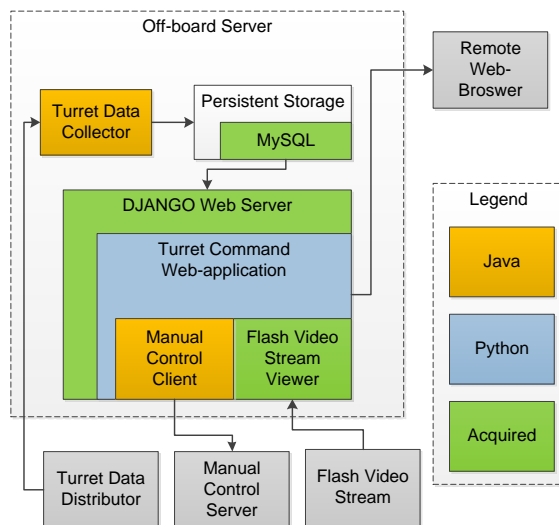


FIG. 7 Web server block diagram

### A. Turret Command Center

The TCC is responsible for providing up-to-date statuses of all connected turrets and historical data to the end-user. The first screen users will view is the home screen page.

On the home screen the user will be presented a grid view of all the currently connected turrets. For each turret, a camera snapshot of the latest target engaged will be displayed. If there has not been any targets engaged by a turret then its initial set-up snapshot will be displayed instead.

From the menu available at the top of every page is a link bar that takes the user to the Turrets page and the Engagement History page. On the Engagement History page, all the targets that were engaged by a turret are displayed in a list format on the left pane, including the turret name and the time of engagement. Clicking on an item will populate detailed information on the right pane. This update handled by AJAX, using a JSON-serialized object that is handled by JavaScript in order to update static elements in the right pane. Some of the information included is the turret relevant information to recreate a radar representation of the turret. The radar representation uses a blue triangle to represent the turret and which way it was facing. The red and yellow semi-circles indicate the warning and firing ranges of the turret. The small orange arc indicates the target of the engagement. Along with the detailed information of the engagement and radar, a snapshot of the target at the time of the engagement is displayed.

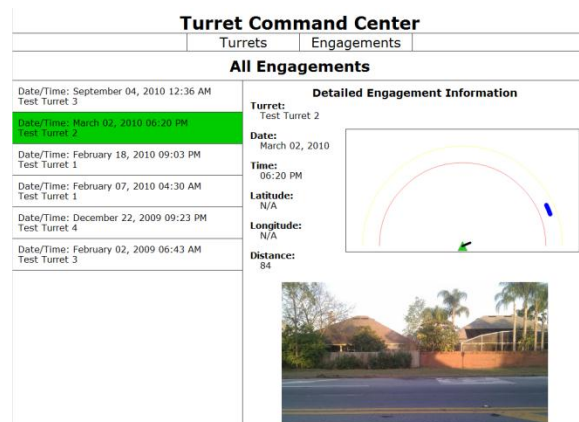


FIG. 8 Turret Command Center view of the online web application

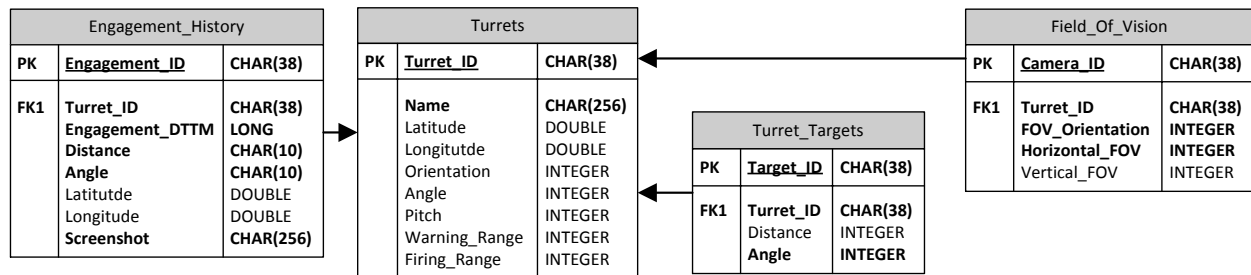


FIG. 9 Class diagrams showing the relationship between the classes used in the online web app.

The Turrets page provides a grid representation similar to the home screen for viewing all turrets, connected or not. If a turret is connected then the turret name is a link that redirects the user to its manual control page. Along with the manual control page link, a link is provided to redirect to the engagement history page, however filtered for that individual turret. In addition to the links there are checkboxes for each connected turret that allows the user to view all the specified turret's respective radars at once. On that radar page, it will display a auto-updating radar representation of each turret. Due to the latency of data collection, and nature of the update, these radars only update once a second.

The last available screen for users to browse is the manual control page. On the manual control page is a live-video stream of the gun-mounted camera, and a java applet to take manual control of the turret. Once manual control is requested, the automated targeting system will disengage and the user will have complete control of the turret. The user is able to then pan and tilt the turret, fire the weapon, and sound the alarm. The manual control can either be disengaged manually or will automatically disengage on either a timeout, or loss of connection, whichever comes first.

To provide the TCC, the web-application was developed using the Django web-framework. Django was selected for use because of its easy to use template system for generating the final html, and its ORM support. The ORM was mapped to our database table structures, this allowed easy to use filtering on the database to select the persisted objects needed to display on a page. Django provided an easy to use template system that included inheritance based on the concept of blocks. Using blocks, a base web-template was created that included all common elements for every page like the title and link bar. For each specific page, the main content was populated after performing logic on the objects passed into the template during rendering. For handling the AJAX requests, jQuery was used to simplify the JavaScript calls and handle the different logic for each internet browser.

## B. Database:

A MySQL database was used to store persistent static and dynamic data for the TCC. The turrets table is the base table that specifies all the turrets that exist in the system. A primary id is assigned to each table to be used for all different mappings to other tables. The field\_of\_vision table specifies the camera configurations of each turret. The table stores the field of vision angles in both axes and their relative positions to the center of the turret. The Turret\_Targets table holds the current targets for each turret, along with the pan angle and distance of the target from the turret. The Engagement\_History table holds information for when a target is either fired upon or warned using the alarm. These tables are populated using the a data collection process, and read by the TCC to generate the web-pages.

## C. Data Collection:

In order to populate the tables, a background java process runs TCP and UDP socket servers to handle data collection from turrets. A UDP socket server exists to collect constant turret status updates of the turret's pan and tilt along with any turret targets it is actively tracking. A TCP socket server is set up for receiving engagement data and screenshots.

To populate the database, once a turret connects to the TCP socket server, it will send over all static data of the turret to store the data needed for the turrets table and field\_of\_vision table. Along with the static data, the turret will send over the initial set-up snapshot of the area the turret covers. After all set-up data has been received, the TCP socket will wait for engagement data and store it in the database along with the screenshot after each transmission is started. Along with the TCP socket, a UDP socket will listen once the TCP socket is connected to receive the current status of the turret including the pan and tilt angles along with the turret targets. This data will be parsed out from the status packets and stored into the database to display on the radars.

## VII. CONCLUSION

The original goal of our group was to create some sort of automated robotic device. Before we had chosen and a

project, we knew that we wanted to create something that merged our respective expertise in electrical engineering and computer programming. The Automated Targeting Proximity Turret has succeeded tremendously in this regard. The multiple facets of the Auto-Turret not only gives it impressive functionality, but allows us to a method to present our knowledge and skills in a real world example.

On the electrical engineering side, the printed circuit board not only contains a micro controller with the Arduino boot loader, but it also features all of our signal lines for the various servos and alarms that are a part of our system, along with the proper power for each. Our design provides a compact, robust, and easy to use interface between the software and hardware of our project.

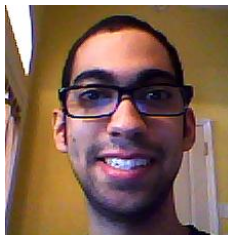
The software the Auto-Turret runs is equally important. The automated controls provided a real challenge between complexity and performance. Able to detect motion and identify targets, the automatic controls make our turret a serious deterrent even when unmanned. Online data collection and control set this project apart from previous Senior Design attempts at creating similar sentry guns. Online access means that users can be anywhere in the world and still have access to our security system. The web interface was also designed with multiple turrets in mind which make this project easily scalable and provides a clear path for future enhancements.

This project also took us outside our comfort zones in designing and implementing the base of our turret. Not being mechanical engineers, we were unsure of the best designs and practices in creating our mount for the airsoft-gun. Coupled with our budgetary constraints, it took a bit of ingenuity and trial and error to come up with our final design.

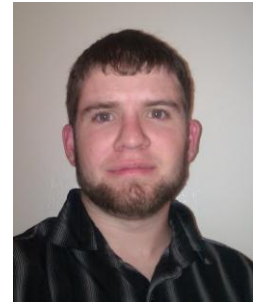
In the end, the Automated Targeting Proximity Turret provided a challenging and rewarding experience. It was the first time we had the opportunity to have complete control over an engineering project of this magnitude and length and see it to completion from its nascent stages.

#### THE ENGINEERS

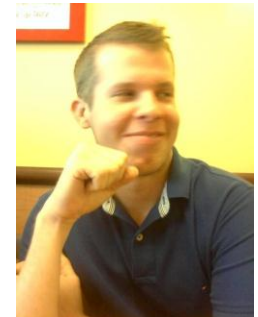
**Hector Colon** is a 24 year old senior. He will be graduating with a B.S. in Computer Engineering from the University of Central Florida. He currently works at HostDime, one of the top 50 largest web hosting companies in the world.



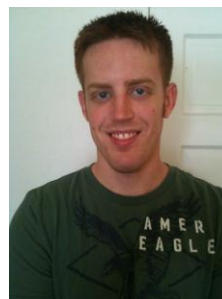
**Adam Horton** is 23 years old and is graduating with a Bachelor of Science in Computer Engineering within the College of Engineering and Computer Science at Univeristy of Central Florida. He has been working as a software engineering intern through the UCF College Work Experience Program for Lockheed Martin for three years on a program with a large portion involving web-application development. He is also the webmaster for the UCF Outlanders club as well a member of Tau Beta Pi.



**Kyle Steighner** graduated from Rockledge High School in 2005 heavily involved in the RHS Robotics club and worked out at NASA on volunteer time. Was dual enrolled as a high school and college student at Brevard Community College. Was awarded scholarships and grants for 2 years of college. Continued education at Brevard Community College while working for Harris Corporation on a 2 year internship. Received AA degree from BCC in 2007. Transferred to the University of Central Florida in fall of 2007 and continued the rest of my education to receive a Bachelors in Electrical Engineering. Received a one year contract internship with NAVAIR on research parkway from summer of 2008 to the summer of 2009 while at UCF. Will be graduating in the Spring of 2011 with a BSEE and has a career lined up with the Department of Defense.



**Nicholas Yielding** is a 22 year old graduating with a Bachelor of Science in Electrical Engineering from the University of Central Florida. He is a member of the UCF Robotics Club. He is currently enlisted in the U.S Air Force through the TDSP program, and will be attending OTS after graduating. Upon graduating from OTS he will be a 2<sup>nd</sup> lieutenant and developmental engineer in the Air Force.



## REFERENCES

- [1] AForge.NET COMPUTER VISION FRAMEWORK  
[HTTP://CODE.GOOGLE.COM/P/AFORGE/](http://code.google.com/p/aforge/)
- [2] ATMEL PRODUCTS – AVR SOLUTIONS – ATMEGA168PA”  
[HTTP://WWW.ATMEL.COM/DYN/PRODUCTS/PRODUCT\\_CARD.ASP?PN=ATMEGA168P](http://www.atmel.com/dyn/products/product_card.asp?PN=ATMEGA168P)