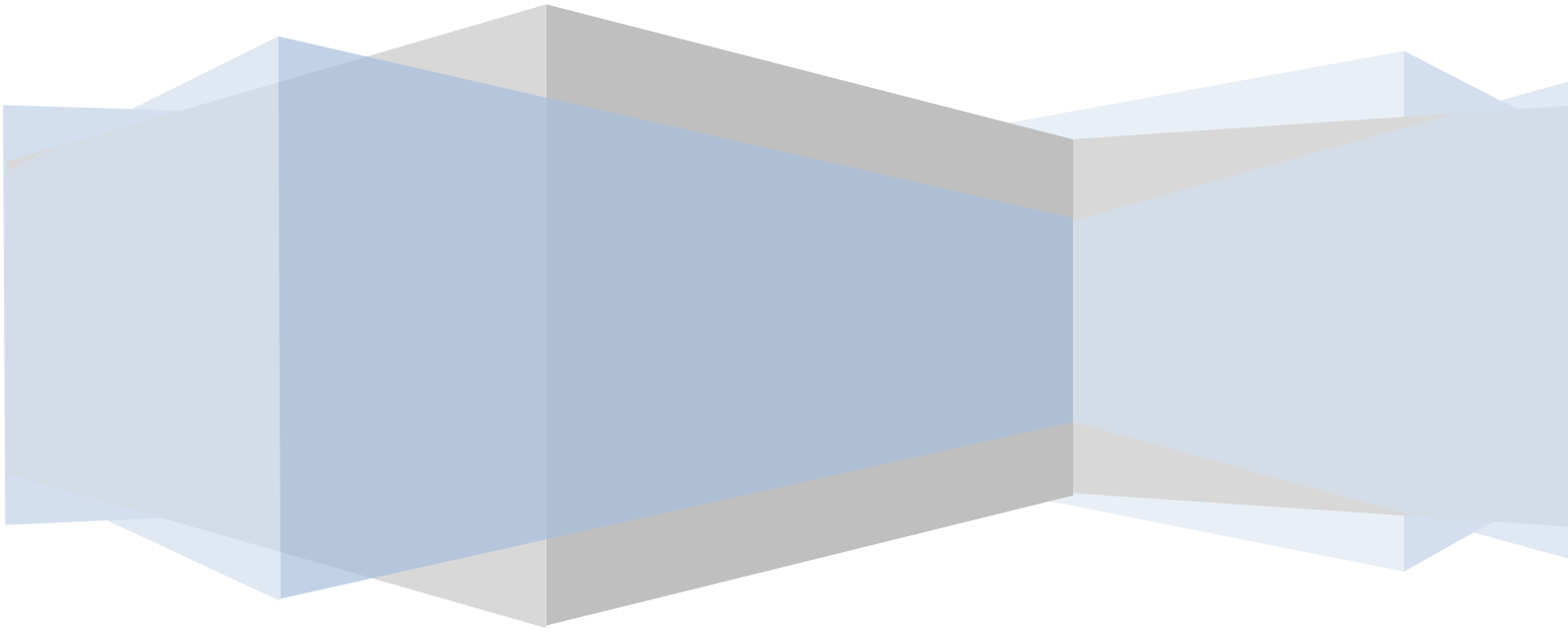
University of Central Florida

Remote Touchscreen-Controlled Defense Turret

Senior Design Documentation

Courtney Mann, Brad Clymer, Szu-yu Huang
Group 11



Fall 2011-Spring 2012

Table of Contents

Table of Figures

Table of Tables

1 EXECUTIVE SUMMARY

The remote defense turret is a platform for defending a sensitive area with human control, but without risk to the defender, or a need for such a defender to possess technical defensive skills. The turret monitors a field of defense with a wireless camera – to which it is physically attached – via wireless-n protocol, and automatically acquires any moving targets evident in this field. The targets are displayed to the user through an Open-CV-based user interface on a touch-screen tablet, which highlights the acquired targets via a color-coded outline. The user selects their target-of-choice – which will be tracked by the system as it moves, and updated constantly – by simply pressing the correspondingly-colored target button at the bottom of the screen. The system then calculates the centroid of the target, and relays the information to an Arduino microcontroller, at which point the Arduino controls the servo motors so as to appropriately point toward the target, and fires. In the prototype presented in Senior Design, the firing mechanism will simply be a laser, but attention was paid in hardware selection to allow for the firing device to be scaled up to a paintball gun, long-range taser, or potentially a traditional powder-bullet weapon, though this was not the group's primary concern; the system was designed to neutralize threats, rather than be an offense platform.

The simple nature of the user interface was intentionally made to not resemble tests of coordination such as those presented in first-person-shooter video games; the goal was a very "plug and play" type of interface that required no training. However, completely defaulting aim to the control of the system left out the ability to fire upon stationary targets, or targets of greater choice than those which the system might automatically select based upon size and speed. Thus, an additional mode is available via the multi-touch feature of the user interface tablet: a desired target may be selected by the placement of the user's finger on the screen, and simultaneously pressing the manual fire button at the bottom of the screen, which is also indicated by a dedicated outline color.

To allow for the desired firing-mechanism scalability and interchangeability, the hardware of the turret was selected to over-perform in comparison to the lightweight laser-pointer in the prototype; it can readily be refitted with heavier devices. The digital servos are capable of traversing the entire field-of-fire in about a fifth of a second when un-loaded, and will slow down proportionally with heavier loads due to different firing devices. Fortunately, common servos from servocity.com were selected; thus, simple modular servo replacement - in the event that a retrofit of this system with a heavier firing device is desired – is easily accomplished.

The challenge of constructing the system lied not only in the control of the individual elements – OpenCV, the Arduino, and the User Interface, among others – but in at least equal proportion in coordinating these systems effectively.

2 PROJECT DESCRIPTION

2.1 Project Motivation and Goals

The motivation for this project was multi-faceted, consisting of civic, functional, academic, and logistic elements. Young engineers often become acutely aware of their ability to affect change in the world in ways that students of most other disciplines cannot; the mission of many undergraduate engineers, upon realizing this potential, is not to simply make the personal profit of which they are often so capable. Rather, visions of what can be created with the toolkit presented by postsecondary education spin in their imaginations and take the shape of responsibility. For Group 11, this manifested in civic responsibility. That was, the group was equipped to make a defense platform which could be the nucleus of a system that would allow defense of not just an area, but of people. The chance to take steps toward responsible engineering early in the careers of the group members was not to be missed.

The group‘s personal goals were not limited simply to the civic responsibility, however. They included a desire to work on a project that the group members found interesting, and which would marry feasibility and challenge. The fusing of programming that had never been attempted by any members, as well as pulling together information from all of the disciplines that group members had studied thus far clearly met these goals. Additionally, the task of management of the project presented a challenge to the group members, none of whom had ever served in a project management capacity.

Functionally, the motivation was to create a human-selective defense platform that does not expose the operator to direct risk, while minimizing training time and the need for physical skill in the mounting of a working defense in a tactically important or personnel-sensitive area.

Logistically, the group expected that this project would fall under the guidelines laid out by Workforce Central Florida to be eligible to receive their funding, thereby enhancing the potential breadth and depth of the project. If the interest alone had not been, this factor would have been sufficient motivation to proceed with the idea of a user-friendly defense platform.

As were alluded to several times before, the goals for this project were to make a system that is easy to use, readily installed, and intuitive. It would not be extraordinarily low-cost, because of the touch-screen interface and control hardware were known to incur a relatively high minimum expense, but if a user (or other engineering team) desired to make a large-scale system, it would be low-cost in comparison to both the original system, and to the human cost associated with putting a live person in a defense situation.

2.2 Objectives

In technical terms, the objective list was not short. The camera system would accurately represent the field of fire, with precision alignment to minimize the aiming error and need for correction in control systems. It would also need a high enough resolution to be accurate at a distance, but not so high that it would bog down the central processing portion of the system (in the user interface) with too much data. The frame rate of the camera would need to be fast enough that it could track targets which moved quickly, such as an erratic attacker, which would be realistic if such an attacker had military training in evasive maneuvers. The transmission range of the camera would be representative of a defender in a room adjacent to the defense area, in an area protected enough for the user to be safe, thereby likely creating a difficult medium through which to propagate signals.

The user interface tablet would then need to process this visual information rapidly and present it to the user in a meaningful way, all while taking in the user's touch-screen input quickly and accurately, minimizing latency between target acquisition and firing. Here, coding would need to be lean and efficient to further minimize strain on the central processor of the system, and the user interface would need to be robust enough to tolerate a range of non-ideal or unexpected inputs from untrained users.

Following this in the system loop, the control system would also have to be able to receive signals across distances and through media similar to those of the front-end camera. The final firing control would be one of the most difficult elements: it would need to be calibrated to accurately match the visual field of the system as presented to the user and the user-interface tablet, and would also control aim via a PID controller without any further feedback into the system.

Around all of this would need to be an enclosure visible to professors (or any other evaluators who might be interested in the inner workings of the system), and accessible to the frequent changes the group knew would probably be required in a prototype system. This casing would need to be light, and have wheels or castors for easy transport, but would need also to be sturdy enough to tolerate multiple teardowns and rebuilds. The control armature would need to be sturdy enough to accept heavier firing systems while still being nimble enough not to make the system sluggish or overloaded.

2.3 Project Requirements and Specifications

In this section, the requirements and specifications for the main systems that were initially used in the project are detailed. These requirements were used as a basis for determining which specific components to select for more thorough analysis. This process was explored in the next section, titled Research Related to Project Definition.

2.3.1 User Interface

It was decided that the user interface for the project must be intuitive, with the aim the user should be comfortable firing the system upon first use without extensive instructions – though a brief on-screen note would instruct him or her – and completely seamless and integrated with the user's thought process after only a few firings.

Due to the fact the system might need to be utilized by personnel that were untrained in projectile combat and likely may simply lack the modern "video game dexterity" necessary to operate a system that was based upon manual aim, it was decided early on that the system would barter some control for ease of operation; the aiming would primarily be handled by the system, with the user operating primarily in a target selection fashion.

Noting the expertise of Apple in designing intuitive interfaces, the group took a cue from their design and began with a design similar to the one shown in Figure 1. The potential targets were to be outlined via blob-detection software, in a high-contrast and distinct color, readily indicating to the user which target corresponds to which colored button on the bottom the screen. When the user was to have chosen a desired auto-tracked target, they would need to press the button of corresponding color at the bottom of the screen and the system would then automatically calculate the centroid of that target and fire upon that location. Alternatively, there would be one manual-firing mode in which the user might simply have selected a target on the screen by placing their finger in that location; at that time, the location would be outlined by a color which does not correspond to the colors of the automatically selected targets, and with a second touch, the user might fire on this manually selected location with long press on the target. At that time, the system would initiate the firing sequence and the target would be fired upon.

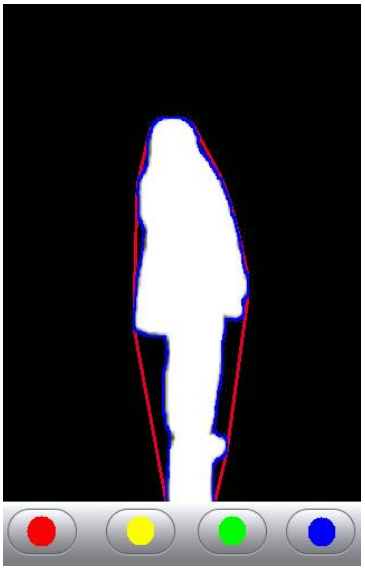


Figure 1: Apple-inspired user interface showing buttons and target outline in red

2.3.2 Tablet

The tablet is the main computational source for the image processing. It would initially receive input from the camera in the form of captured frames. It then analyzes the images and performs the programmed operations to successfully detect and track multiple moving targets. In addition, it has the capability to recognize manually chosen stationary targets. The specifics of this process are detailed further in section 4.5.2, titled Target Acquisition. In order to accomplish these tasks, the tablet has to meet certain requirements. First of all, it must have the capability to interface wirelessly with both the camera and the system processor, where it sends the objects' locations so the servos can be properly oriented. The tablet must also be easily programmable, since an application needs to be created to serve as the user interface to select the targets. Another factor was the necessity of a touch screen interface, since one of the primary project goals was to create a touch-operated system. Additional considerations included computing power and processing speed, which were important for processing the images with minimal delay.

2.3.3 Targeting Control

2.3.3.1 Pan-Tilt Motor control

The motor system is controlled by the Arduino single signals board, where the commands are sent from the user interface and signals are delivered to the motor. To aim at a stationary point that is chosen by the user, the coordination data is sent from the user interface to the Arduino board, and the motor moves to the specified position and a fire command is sent as followed. To track a moving designated target, user first selects a target. The coordinates of the target are continuously sent to microcontroller to update the position of the servos. The servo system then keeps tracking the moving target until it moves out of the range. It is easier to calculate the path from the current position to the next position every time. Also, it is better for a smooth servo movement. To track down a moving target, the motor requires at least 15ms delay to position itself. When the motor tracks down the target, it keeps aiming and move along with the target until the target moves out of the shooting range. Servos then remain in the previous coordinates that are last received from microcontroller. The moving angle of the motor is 45 degrees horizontally and 85 degrees vertically.The motors are driven by digital pins of the microcontroller, which uses PWM to control the servos. The assumption was made that targets do not move faster than 9 feet per second.

2.3.3.2 System Processor

The system processer is the main controller for the positioning of the servo motors and also tells the firing device when to fire. Additionally, it acts as the main onboard control center for connecting all the separate components and allowing them to communicate with each other. One of its main tasks is

interpreting the wirelessly transmitted locations of targeted objects into commands for the motors.

For a system comprised of servo motors, the microcontroller has to provide pulse width modulation for at least three motors, two for positioning the turret for proper aim and one to fire the paintball gun, if it is implemented. In addition, the microcontroller also has to have ports suitable for communicating with the servo motors and firing mechanism. Because it has to have wireless communication with the tablet, it must include this capability for the chosen wireless method, whether it is Bluetooth, Wireless USB, Zigbee, or some other alternative. Since it was important that the delay between choosing a target and firing upon it be as small as possible, the system processor should have a high clock speed, at least 16MHz. Also, the memory must be a sufficient size, at minimum 16kB. Finally, because of the desirability of a portable system, size constraints must be factored in as well, so a specification is set that the microcontroller be smaller than 8x8x2.

2.3.3.3 PCB Design

Three subsystems' components were implemented on PCB: the core of the Arduino microcontroller, Atmel ATmega328, Two voltage regulators, and wireless communication modules. It was measured that the servos drew close to 1A each, and the current drew by other components was less than 1A. This gave the rated current close to 2A and the thickness close to 1mm. After calculating the trace width for printed circuit board based on a curve fit to IPC-2221, the required trace width was determined to be 0.0712mm, resistance 6.31 mΩ, voltage drop 12.6mV, and power loss 25mW for internal layers. For external layer in air, the trace width was determined to be 0.0274mm, resistance 16.4 mΩ, voltage drop 32.8mV, and power loss 65.6mW.

2.3.4 Firing Control

2.3.4.1 Tablet/Microcontroller Interface

The interface between the tablet and the microcontroller needed first to be wireless, since a primary function of the system is that it is remotely controlled. While it could have been accomplished in a more complicated way – by establishing a wired connection between the UI tablet and the microcontroller, and then separately adding a standalone receiver board onto the firing part of the system, and utilizing it to control the firing – this, in the estimation of the group, would be needlessly involved. The simplified block diagram is shown in Figure 2; note that it shows the peripheral device attachments coming from the microcontroller, and their final control points.

On a technical level, the wireless connection needed strength at range; a reasonable estimation of defense distance was 30m indoors, so a spec was put into place dictating a 40m range necessary for the system, which immediately

excluded IR and Bluetooth. A decision was then made that the method of communication between the tablet and microcontroller should be wireless-n, because the IEEE standard for that protocol is 70m indoors, which accounts for walls.

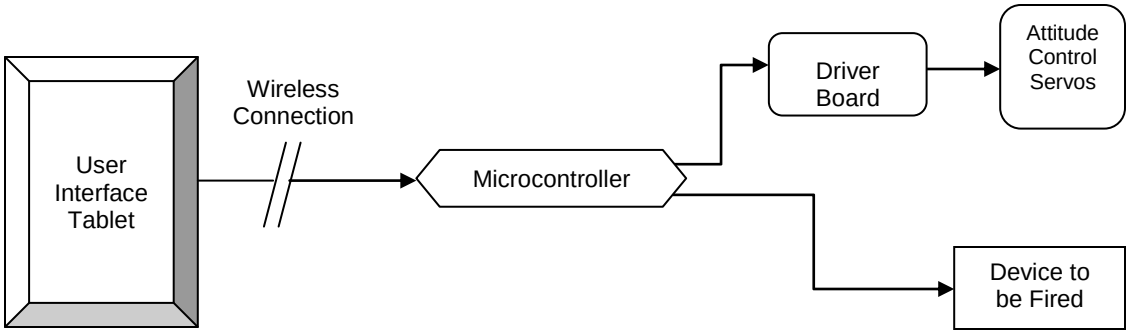


Figure 2: Block diagram for the interface between the tablet and microcontroller, showing also the eventual termination of the controller's outputs.

2.3.4.2 Microcontroller-Gun Interface

The interface between the microcontroller and gun – or as it was later determined, the laser – was to be extremely simple. The microcontroller would simply use a comparator or diode to allow completion of the battery circuit, which would then engage the firing portion of the platform. In Figure 3, below, a diode is used for simplicity. A comparator could work as well, but in a slightly more complicated manner.

A similar actuation would work for a paintball or airsoft gun, though the connection would require more disassembly.

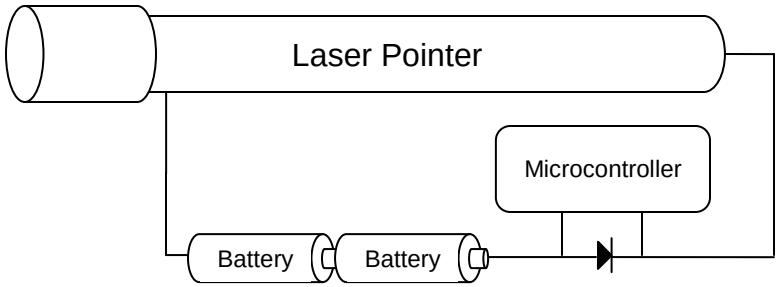


Figure 3: Block diagram for the interface between the microcontroller and laser pointer, showing the microcontroller engaging a diode, and providing power to the laser pointer

2.3.4.3 Paintball Markers

In the project, there were two different approaches to build the turret system. The first approach was to implement a paintball marker to actually fire paintballs and mark the designated target. That way was easier for the user to see if the marker

has aimed at the correct target and accurately shot the target. It was desirable to avoid any mechanical problems that might be encountered while implementing the hardware part of turret system. Therefore, the paintball marker that might potentially be used should be able to be triggered electronically. An additional circuitry should be connected to the trigger of the marker, which is controlled by the Arduino microcontroller.

The weight of the marker was crucial to determine which type of servo motors that was going to be used since the load weight and torque were important factors that must be taken into account in order to avoid overshoot and burnout. Ideally, the marker should be around 5 to 10 pounds. It was desired that the designated target would be fired upon nearly instantaneously as it was tracked to avoid missing as it moves away. This specified a need for a high speed paintball marker. 20 balls per second would be ideal for the project. However, this criterion put the group in the high end of paintball markers. The sufficient speed range would be 15 balls per second to 20 balls per second.

2.3.4.4 Laser Pointer

Another approach was to use a laser pointer in the turret system for practical demonstration purposes. Unlike a paintball marker, which involves calculating the gravitational effect on the shooting angle, a laser pointer simply illuminated the designated target by a bright spot of light. The requirements for a laser pointer were simple. The range of the laser pointer had to be at least 40 meters and the brightness had to be noticeable and easy for the user to see even in day time. However, it should not be too powerful for implementation into the turret system. The group did not intend to damage any object or run the risk that someone could possible lose their vision while constructing the project.

2.3.5 Image Processing

2.3.5.1 Camera Hardware

The camera for the system needs to offer sufficiently high frame-rate to allow for tracking of quickly-moving objects. The desired frame rate for continuity was calculated as follows: assuming that the processor needs a 10% overlap from frame to frame to track, a very thin person (lateral thickness of 0.02m) would need to be captured visually having traveled no more than .018m from frame to frame. Further assuming this individual is running at a *very* high sprint pace of 40 km/h, they will traverse this distance in

$$\frac{.016m}{44.72\frac{km}{hr}} = .0162s \tag{1}$$

which means that they would need to be photographed in periods no longer than .0162s. Inverting that number gives a minimum desired frame rate of 62 fps (frames-per-second).

For resolution considerations, it should be observed that using old 16 bit color depth on with a 1024x768 camera, sampling at 62 fps results in 93MB/s data throughput, which could significantly slow down a system. As a result, the group had focused primarily on lower resolutions that would accomplish the same task, primarily 640x480. At the same frame rate, this lower resolution requires a significantly reduced 36MB/s data throughput, enhancing the system's ability to process the data in a timely manner.

The camera needs to either have built-in wireless capability, or be a USB device which can plug into a secure wireless USB hub.

2.3.5.2 Target Acquisition

Targeting acquisition involved the visual processing of the images obtained from the camera to determine the existence and the location of a target. The software had to be capable of tracking at least one individual moving target at one time. First it has to recognize that a new designated target that has chosen by the user based on color. Then it has to find the location of the target, so the motors can be directed where to aim.

Object detection can be implemented with the use of a background subtraction technique. This requires that the system capture and store background frames to be the reference against which new frames would be compared. Due to inevitable sporadic background movement, for example a car being parked or a trashcan being moved from one side of the frame to the other, the reference image has to be periodically updated so that the newly positioned objects are not read as targets. Current frames have to be acquired and stored at a relatively high rate, so that new targets can be detected quickly.

After the current frames are compared with the reference, the target is identified. The center of the target can be found using a centroid calculation. By looping through this process, an object can be tracked by the changes in their centroid locations. This informs the microcontroller how much it needs to move the servos to maintain its aim with the target. If the gun is in its reset position facing straight ahead, the program must compare the target's position with this default location in order to orient the gun correctly.

Another feature of the system is the ability to select stationary targets. When a user has selected an object that is contained in the background layer, the system recognizes the specified point as the new target. As with the moving targets, the location is then be calculated, however since the object does not move, the 'tracking' is unnecessary, so this calculation only needs to be done once.

2.3.6 Wireless Communication

It was necessary to implement wireless technology in the project. To avoid any possible danger or risk users might encounter while monitoring potential targets, wireless communication between the user interface and both camera system and microcontroller was necessary and requires the range of transmission to be at least 10 meters. The two communications had different requirements due to different kinds of data that was transmitted through the protocols, the size of the packets, and the speed of the transmission. Because the system from the camera to the user interface had video streaming transmission in a real time manner, massive data was transmitted through the protocols. Data rate and network acquisition time became crucial for the project to be completely successful. Ideally, it was assumed that the wireless communication module between the camera and the user interface had capability of transmitting at least mega bits per second with acquisition time in 2 milliseconds. The requirements are summarized in Table 1, below.

Table 1: Camera-the UI Wireless Communication Requirement

Data Rate	Transmission Range	Network Acquisition Time
>15Mbps	>10m	<2ms

Data rate in the communication module between microcontroller and the user interface did not need to be as high as it is for the camera system. In the project, only small packets of data were transmitted from the user interface to microcontroller. That was, only simple pulse signals that have been converted from analog to digital in the image processing module on the tablet was sent through the protocol. Data rate anywhere between hundred bits per second to kilo bits per second would be sufficient. Network acquisition time was still important in this module, since the whole project is operated in real time, so 2 milliseconds would be adequate. The requirements are summarized in Table 2, below.

Table 2: Microcontroller- the UI Wireless Communication Requirement

Data Rate	Transmission Range	Network Acquisition Time
>15kbps	>10m	<2ms

2.3.7 Range Calculation

2.3.7.1 Rangefinder Hardware

In the scenario where a paintball gun was used as the firing mechanism, a rangefinder would be necessary to properly orient the gun to counteract gravitational forces. The range of the turret system was determined to be an approximately 9 feet arc spanning an 45 degree angle. This set the requirement for the maximum distance from the turret that the rangefinder must be able to read at 9 feet, as illustrated in Figure 4 given below. Because the rangefinder most likely operates at a single point at any given time, it must also have the capability to change position, so that it could effectively mirror the target movement. This could be easily accomplished by securing the rangefinder to the gun.

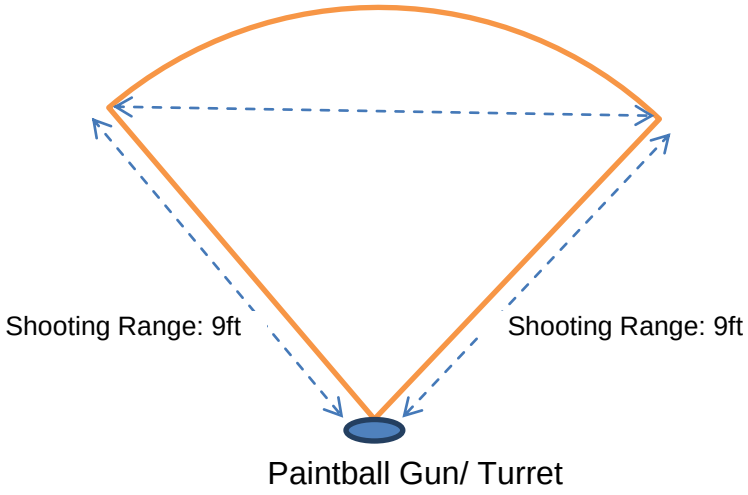


Figure 4: Illustration of the range of the turret

2.3.7.2 Software

Another consideration was that the rangefinder must interface with the system processer, so that the gathered data could be translated into commands for the orientation of the turret. This would most likely involve a USB connection to the processor, and might incur some additional programming as well, in which case OpenCV were utilized.

2.3.8 Power Supply

The system was to have a wide range of components. Table 3 summarizes the expected peak power requirements for the turret portion of the system, since the charging of the user interface was not relevant during times of operation.

Table 3: Power Requirements of Individual Components

Part	Peak Voltage	Peak Current	Quantity	Peak Power
Motor Control Servos	6.00 V	500.00 mA	2	6.00 W
Arduino	9.00 V	50.00 mA	1	0.45 W
Laser Pointer	3.00 V	33.33 mA	1	0.10 W
Totals	18.00 V	1.08 A		6.55 W

Since these components required large amounts of current (and therefore power) at any given moment, the system therefore required either a large battery (which inhibited transport) or the use of AC from a generator or wall outlet (which limits installation location and makes the project susceptible to interruptions of service from the power grid).

The Arduino is normally powered by an AC adaptor which outputs 9V DC. Laser pointers – most notably the model which the group initially selected – are typically powered by two AA batteries, which are 1.5V cells in series. Thus, a 3V DC adaptor, properly wired, could serve as a suitable power supply for this device. Additionally, the servos are controlled by the Arduino, but this device does not supply nearly enough current to operate them, so they needed some form of driver board to act as a buffer between the Arduino and the servos. This board could easily run on an 18v, 2A AC adaptor. The above specifications strongly suggested that AC should be the power source of choice for the project.

2.3.9 Hardware Housing

The housing for the project was constrained by requirements of durability, portability, accessibility, and visibility. In terms of durability, it needed to be able to withstand the torque applied by the servos to the armature, as well as frequent transport and disassembly. Since the project would be carried from workspace to workspace, portability would be a high priority. Since very few engineering projects work on their first attempt, the device needed to be easily accessible to alteration. Also, the group was advised early on that during the final evaluation of the project, the parties performing the evaluation would be very interested in seeing the inner workings of the device, so it should be visible.

To address the durability requirement, a material with a strong tensile strength but low brittleness was required. This material would require tolerance of .121 kg-m torque resulting from the operation of the servos rotating at maximum speed, as well as the weight of the armature, which the group estimated would be in the range of 3 kg. For scalability, it was determined that, in case there needed to be a larger firing device – such as a heavy paintball gun – added later, there should be higher weight and torque tolerances. Also, this material would need to

balance the attributes of having enough weight to counterweight the motion of the turret rotation, but not too much to interfere with portability.

This portability would be achieved by a marriage between the use of a lightweight-enough material, and casters that allowed it to be rolled. The casters would be mounted through the base, and they would need to be high enough quality to handle repeated placements on the ground; this was a concern because members of the group had previously worked with music equipment, and noticed that casters for such equipment wear out prior to any other elements of the design.

Accessibility was of primary concern both because of the necessity for the project to be worked on after it would be initially constructed, and because the evaluators might desire to inspect the internal components more closely than a cursory visual exam could prove. The reviewer may want to inspect the robustness of the internal components beyond a visual inspection. However, in this respect, the visibility component of the project – both for inspection by mentors and outside parties, as well as for ease of troubleshooting before project completion – would need to be addressed via the use of a transparent material.

3 RESEARCH RELATED TO PROJECT DEFINITION

3.1 Division of Labor

The project had been split up according to Table 4 given below. The tasks were divided according to estimated difficulty, with each group member taking responsibility for one of the larger tasks (User Interface, Motor Control, and Image processing), and a few of the smaller tasks. Brad was in charge of power, motor control, hardware housing. Fairen took on the tasks of wireless communication and PCB design, and Courtney was responsible for the image processing and camera.

Table 4: Division of Labor

Brad	**Fairen**	**Courtney**
Hardware Housing	Wireless Communication	Image Processing
Power	PCB Design	Camera
Motor Control		

3.2 Existing Similar Projects

The concept of a turret paintball gun, in various forms, is quite a popular one. Through the research numerous examples of this design were discovered, ranging in scope from hobbyist creations to professionally built machines. There were many variations on the basic idea of a paintball turret, based in large part on the desired application of the project, whether it was for recreational or security purposes, but the group was able to sort out multiple projects that share many of the same features that it wanted to incorporate into the turret gun. Because the designs and components used varied widely from project to project, a comparison of the different options for each part of the system and the level of success attained by each aided in forming the decision for which method to choose. This proved beneficial in helping to avoid “reinventing the wheel” by building on the previous experience gained by predecessors and saving both time and expenses incurred by avoidable mistakes. In addition, it provided the group with a way to narrow down its design options by giving helpful suggestions for which components to use.

Among the multitude of designs available, there were quite a few from the UCF’s Electrical Engineering Senior Design classes from previous years. Three in particular stood out as the most similar to the Remote Touch-Controlled Defense Turret: the Motion-Tracking Sentry Gun, the Paintball Targeting System, and the Automated Targeting Proximity Turret. By reviewing the design and construction

processes of these groups, the team members were able to expand their own knowledge on the subject, which gave them a basis on which to make informed decisions about the direction that they wanted to take the project.

The Motion-Tracking Sentry Gun was a turret paintball gun that autonomously detected and tracked motion, and fired upon any moving targets it found. The group used a Xilinx XC3S200 FPGA for the image processing because it fit their specified requirements, which was to be portable and stand-alone. In order to receive camera inputs and control the servomotors through the output, the FPGA needed to be mounted to a circuit board; they decided on the NEXYS board by Digilent Inc, due to its available inputs for expansion boards. A CCD camera was connected to a video decoder board, which captured the analog video signal and converted it to a digital output. This was then sent to the FPGA for processing the images. The FPGA also acted as the servo controller by means of a PModCON3 Servo Connector board, which connected the FPGA with the three servo motors. Of the three, two were used to operate the motion of the turret -one for the up and down positioning, one for the left and right movement by means of a turn-table- while the other was used to pull the sentry gun's trigger.

The image processing was broken down into three basic steps: first detecting an object, then representing it for ease of calculation, and finally tracking the object's movement. In order to detect an object, the CCD camera was used to capture frames, which were stored in the NEXYS onboard memory, then a background subtraction technique was implemented wherein the background frame was compared to each new incoming frame, and any differences in pixels that were detected between them was determined to be an object. This was then represented by a rectangle, which was approximated by the object's outermost pixels. From there it was a simple geometric process to calculate the centroid of the rectangle. Once that was accomplished, the process of background subtraction was again utilized to track the movement of the rectangularly-represented object. For a moving target, the rectangle positions differed from frame to frame as the location of the object changed. The distance between two centroids from consecutive frames was calculated, and a signal, was sent through the Servo Connector board that was based on that distance. This technique, known as Pulse Width Modulation, was the basis for the targeting control. It works by setting the servo position depending on the width of the signal pulse. For example, a width of 1.5 ms would position the servo to be pointing straight ahead, while a width of 2.0 ms would turn it towards the left and 1.0 ms width would turn it towards the right.

For the power supply, a standard United States 120V AC outlet was employed, which was then converted to the DC voltage necessary to power the individual components. The power was run through a step-down transformer to reduce the output voltage, then a bridge-diode rectifier in parallel with a capacitor to convert the waveform to a constant DC value. This voltage was sent through a linear voltage regulator, which had an output of 6V. This is enough to power both the

NEXYS board and the servomotors, but the camera required an input of 12V. This problem was solved by connecting a switched-mode power supply, also known as a boost converter, which effectively outputted the necessary 12V from the 6V input that was the output of the linear regulator.

For the assembly of the turret, a bracket was designed and constructed to contain the paintball gun. As mentioned above, three servomotors were used for manipulation of the turret. The first controlled the pitch movement by means of a shaft connected to the bracket, the second was connected to the base, whose purpose was to support the weight of the turret and house all electrical components. This was affixed to a lazy-susan to allow for the right and left motion due to the rotation of the bracket housing the gun. The third and final servo was set up for pulling the trigger of the paintball gun by means of a linear actuator.

From this project, the group was able to learn a number of useful things that were relevant to the RCPT. One of the first things that was noticed was the image processing procedure used, which was appealing both in the simplicity of its steps, as well as the effectiveness of its methods. An alternative to the rectangular representation that was considered was an outline conforming to the curves of the object; this would be aesthetically smoother looking but with the tradeoff of more complex programming. Servo motors also seemed to be an appealing choice. They worked well within their project, and would fit the requirements of the RCPT. The choice of the FPGA for the system processor was still uncertain, due to the large number of parts needed to connect all the components and the difficulty of the programming language. For the power source, the driving force was again simplicity, as the MTSG relied on a straightforward AC-DC converter powered from an AC wall outlet. This defeated the need for batteries and multiple power sources, although it was somewhat limiting to the portability factor. However, it should be sufficient for the RCPT's power.

The Automated Targeting Proximity Turret was another recent project that reasonably matched the design specifications. This system was automated as well, but proved more highly advanced than the MTSG in the amount of features included. When a subject initially entered the field of view of the monitoring camera, the turret calculated their distance from the base; at the point where the subject came within the turret's range, an alarm would sound warning them to immediately exit the area or they would be fired upon. In addition, the onboard software documented each case of the turret coming into contact with a subject, which was collected by an off-board server and placed in the Turret Command Center, a web application that displayed the engagement history of the turret. A manual mode was also available, where the user could control the system through the PC connected to the turret.

The image processing portion of the system was handled by a computer and 3 webcams, one high definition and two low resolution. The group relied on the AForge.NET computer vision library to aid with the motion detection and target tracking programs. The HD camera was mounted to the barrel of the gun for precise target acquisition. The two LD cameras were connected to the base and remained stationary. Their purpose was to each focus on a specific direction that the target could be moving in, one on yaw and the other on pitch, so that the difference in consecutive images could be calculated to position the turret in the proper direction. Another component used for aiming was the rangefinder. Because of cost constraints and range requirements, the group decided that a single point laser range finder, the Fluke 411D, would best suit their needs. In order to send the range information to the PC, a Porcupine Electronics control board, which was designed specifically for the Fluke to interface with a computer through USB, was purchased and installed on the rangefinder.

The APTP employed a number of different batteries to power their project. The computer and the rangefinder both included a built-in battery, and the computer supplied the power to the cameras as well via USB connection. Similarly, the alarm, servo motors, and microcontroller were all powered through the control board. After the computations were made based on required current and voltage for each component, a 12V lead-acid battery was chosen as the power supply for the control board.

The turret itself was assembled from wood, with the base acting as a turn-table. For the ATPT, the group opted to use an airsoft rifle as a replacement for a paintball gun. The airsoft gun was suspended from an aluminum tube supported by two wooden arms. As in the MTSG project, there were three servo motors: one to rotate the gun-supporting rod, which controls the up and down movement of the gun, one to spin the turn-table, which changes the yaw position, and finally a third to control the trigger.

The application of their image processing seemed slightly more complex in nature than the previous group’s, involving three cameras instead of one and using binocular vision to track the object’s location. However, their idea of using a prebuilt library for ease of programming was appealing, and could be helpful for the RCPT processing. Also, the use of a rangefinder was critical in determining distance so the gun could be properly aimed to account for gravitational forces. Because of the number of different components, multiple batteries had to be used to accommodate the varying power requirements, which seemed messier than a single AC power source. The amount of features offered by the ATPT was impressive, but for the purposes of the RCPT, the group decided to keep it simple, due to the complexity of the touch screen user interface. This is comparable to the ATPT’s web application that monitored all the tracking activity, combined with the manual mode that could be activated through the onboard computer.

A third project from the EE Senior Design class, the Paintball Targeting System, offered yet another method for implementing the desired goals of the system. The completed project gave results similar to the two mentioned previously, which was a machine that automatically detected and fired upon a targeted moving object, but used an algorithm that determined the target based on a specified color. Rather than military or defense application, this device was created to be used during paintball competitions. Manual control was also available, if the user desired to override the automated commands and take control of the turret.

For the central processing of the PTS, it was necessary to find a system processing board that could not only handle the interactions of the individual components and send the commands to the DC motors, but also take in the visual inputs and perform the computations for the image processing needed to properly orient the paintball gun. In order to fit these requirements, the VIA® EPIA Pico-ITX was chosen due to its 1GHz processor and ability to support up to 1GB of RAM. This was connected to a camera, which provided the visual inputs necessary to track the target. Through the use of the Open Source Computer Vision Library, which was developed by Intel Corporation, the Pico-ITX board captured the frames from the camera and used a color detection algorithm to look for a specific color in the frame. If the color was found, it was recognized as an object and the system control module was alerted to the target's location through a centroid calculation. The color detection method was decided to be the best course of action because it did not rely on comparisons between two frames, which would have been slower, but instead on probability distributions. It also would have been troublesome to calculate the difference in frames with a moving camera, since the frame of reference is constantly changing as the motors move the camera. Once a target is detected, the Pico-ITX sends rate commands to a motor control board, the Mini SSC II, which converted them into pulse width modulation waveforms that specify the magnitude and direction of the motor movement. The PWM signal was finally fed into and interpreted by speed controllers, which directly controlled the movement of the two motors involved in aiming the paintball gun. To control the firing of the gun, the PWM signal was also sent to a relay that was connected to the trigger. When the waveform was long enough, the relay closed the circuit, causing the gun to fire. This was directly followed by a shortened signal to open the circuit and release the gun's trigger.

To power the Pico-ITX and the relays, an Advanced Technology eXtended, or ATX, was used. This also provided power for the camera, which was connected to the Pico-ITX through USB connection. A separate 12V, 4A power supply was connected to the DC motors through the speed controllers, which also aided in powering the relay, since it was in parallel to the speed controllers. Additionally, both the paintball gun and the motor control board were connected to 9V batteries.

For the mechanical portion of the turret, two DC motors were used. As was common to the previous projects, one was connected vertically to control the pitch axis, while the other was attached to the base to account for the yaw motion, with a third connected to the trigger. The compressed air tank needed for the firing of the gun and a box containing all the different controls and power supplies were mounted outside of the moving surface to lessen the load on the motors.

The image processing done by this system was handled differently than either of the previously mentioned projects, using a color tracking technique to identify its targets. The benefits of this method are the increased speed when compared to, for example, background subtraction, but the drawbacks are the increased limitations of the target identification process. Since the RTCDT is meant to target any large moving objects, regardless of color, this method was determined to be less than ideal. Similar to the ATPT, the TPS utilized a prebuilt library of functions to aid in the coding, although they used OpenCV rather than the AForge.NET library. The specifics of these libraries will have to be looked into further to determine the strengths and weaknesses of each. For the power requirements of the project, multiple batteries were connected, again bringing up the issue of portability versus cleaner design. Finally, the motors used were DC motors, rather than the servos employed by the other two projects. DC motors use voltage lines to tell them at what speed to move, with a higher voltage indicating a quicker speed. Servos, on the other hand, receive a pulse that indicates the angle it will turn by the width of the signal, which is dependent on the duration of time. Further consideration will be given to determine the merits of each type of motor.

From these three projects, the group was able make comparisons not only theoretically, but based on actual experience with the systems. By weighing each subsystem of each project against the others, the group observed which operated the most successfully in practice, as well as which one best matched the specifications. This helped to inform the decisions about which possibilities could be thrown away outright, and which ones the group wanted to devote more extensive research to. This is detailed further in the next section, Relevant Technologies.

3.3 Relevant Technologies

There were many considerations that needed to be made in terms of determining the subsystems that would make up the project. For each procedure, there was an extensive range of options from which to choose, each with similar results but widely divergent processes for accomplishing these results. Based on the research as well as the previous paintball turret projects examined above, a selection of options for each part of the project were assessed, and a comparison of these choices revealed the best solution. Some of the main factors taken into

consideration included cost, size, power consumption, and how closely each conformed to the desired specifications.

3.3.1 Rangefinders

In order to properly track the location of the target, it was essential for the control system to know not only the horizontal and vertical distance in a given frame but the depth as well, so that the angle at which to fire the gun could be correctly computed to reach the target of choice. Since the camera would supply only a two dimensional rendering of the target field, the system would need an additional component to fill in the missing depth information. A rangefinder was exactly suited to this purpose; the next question became which type of rangefinder would best match the specific requirements of the project. The appropriate solution had to have a range at minimum equivalent to the gun, which was estimated to be approximately 9 feet. In addition, it must have relatively good accuracy, in order to effectively aim the gun. After a variety of different types of rangefinders were researched, the top selections were infrared rangefinders, ultrasonic rangefinders and a laser pointer in conjunction with an image sensor.

IR rangefinders use triangulation to calculate target distance, where a pulse of light is sent out and reflected off an object. The angle it returns is proportional to the distance, which can then be easily calculated. IR rangefinders offer fairly good immunity to interference from any ambient light, as well as indifference to target color, and their simplicity, low power requirements and small size make them popular in many robot designs. Their disadvantages lie in their small detection range and the thinness of the beam width, which means that if the object is not directly in front of the beam, it would not be detected. Another problem with this option is that due to the triangulation process, there also exists a minimum range, meaning there would be errors in detecting any objects that are closer than this.

Another alternative, the ultrasonic rangefinder, operates on the same basic principle as the IR rangefinder but with sound instead of light. The radar emits a mostly inaudible sound, and then waits for the return echo that bounces off the object. The time taken between transmission and reception could be used to calculate the distance of the object. This method is relatively inexpensive… Also, the echo can be distorted easily by factors such as angle of the object relative to the rangefinder and material properties, which could potentially give erroneous results.

A third option was to use a simple laser pointer, such as the kind used as a presentation tool, in combination with an image sensor. The laser is offset from the sensor a known distance, with their axes lined up parallel to each other, as illustrated in the figure below. The sensor uses an algorithm to detect the brightest pixels in the image, which is the point where the laser beam is reflecting

off of an object. The object's distance can then be computed by simple geometry, based on Equation 2 given below. Figure 5 depicts the setup of the sensor and the laser pointer. While the expense for this system exceeds the options mentioned previously, the range is also greater, which is a primary factor in the decision. The other main requirement, which was the attainment of a high level of accuracy, can be achieved with a high resolution image sensor, such as a 1024x1 image sensor from Panasonic.

$$\boldsymbol{D = \frac{h}{tan\theta}} \tag{2}$$

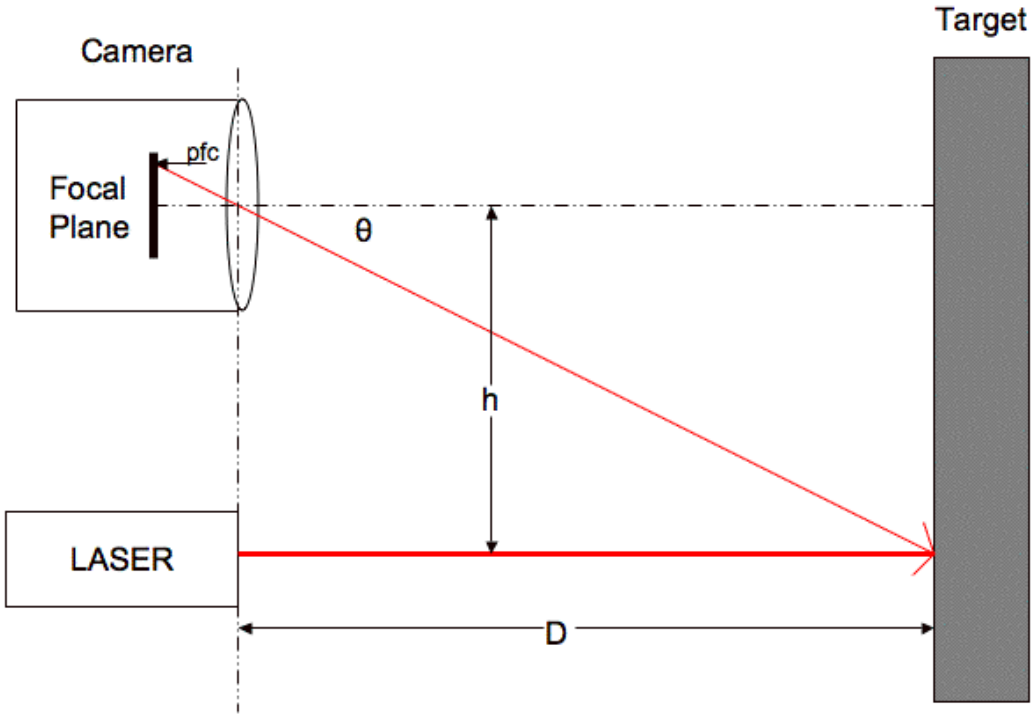


Figure 5: Setup of Laser Pointer and Image Sensor for Distance Calculation

3.3.2 System Processer Board

The system processer board acts as the control unit for the entire turret. Its function is to communicate with the individual components and integrate them into a cohesive whole. This involves taking in the captured images and processing them to recognize targets and determine their location. It then converts this information into commands which are sent to the motors to effectively track and shoot the object. Since the group decided from the beginning that a tablet interface would be included for user interaction, it was logical to let the tablet handle the image processing as well. This left the tasks of motor control and component integration. Among the options at the group's

disposal, there were two main categories of processors that were considered, which were FPGAs and microcontrollers.

3.3.2.1 FPGA

One of the processors under consideration was the Field-Programmable Gate Array, or FPGA. This structure contains logic blocks that can be configured to perform combinational logic or mathematical calculations. Since the FPGA would have needed to be connected to the motors, a circuit development board would also be required, with inputs for expansion. FPGAs execute their code in parallel, which makes them a good solution for problems with repetitive procedures, for example image processing or radar range. This was one of the reasons it was chosen and successfully implemented for the Motion-Tracking Sentry Gun discussed previously. For the RTCDT, however, since it was already decided to leave the image processing portion to the Android tablet, this was not a deciding factor.

One of the downsides to this option is the large amount of power consumed compared to a microcontroller. Additionally, FPGAs are more expensive than their counterparts, due to their greater complexity. Another concern was the intricacy of the programming itself; FPGAs need to be programmed in a hardware descriptive code, such as VHDL or Verilog, which is notoriously difficult to program in.

3.3.2.2 Microcontrollers

The other choice for the processing unit was microcontrollers, which, unlike FPGAs, already have their circuitry and instruction set preconfigured. While more limited in that aspect, they also have smaller power consumption and are less expensive. They are also easier to program, since the code can usually be written in a high-level language such as C or C++. Because the device would be used for relatively simple tasks, it was determined that the benefits of a microcontroller outweighed those of an FPGA, and it was therefore selected for the project.

The next step was to choose a specific model. Among the many viable options in this field, the group narrowed the choices down to Texas Instruments' MSP430 and the Arduino Uno, which contains the Atmel ATmega328 as its core processor. The MSP430 was a strong contender, mainly due to the fact that the group was in possession of a few already, but in the end the Arduino was chosen for a number of reasons. First of all was its 'ease of use' factor. Besides the fact that its programming environment is beginner-friendly, the software and hardware are both well-documented, and there exist numerous pre-built libraries that would greatly help in the coding process. Of all the boards in the Arduino family, the Arduino Uno was singled out because it contained all of the features that were needed without too many extraneous ones. According to the datasheet,

it has 6 analog inputs and 6 digital input/output pins, which can be used to connect the servo motors, in addition to a USB connection. For memory storage, it includes 2 KB of SRAM, 1 KB of EEPROM, and 32 kb of flash memory, although of that 0.5 kB are used by the bootloader to upload programs onto the board. Power is supplied through the USB connection; alternatively, an external supply is also acceptable, in the form of either batteries or an AC to DC adapter for use with a standard wall outlet. The allowed range of input voltage for the board to function correctly is 6 to 20V, although 7 to 12V was recommended for better results. A resettable polyfuse provides protection from shorts or too much current to the computer connected through the USB. Table 5 given below summarizes the main features of the board. Another key factor in the decision was the necessity of wireless communication between the board and the tablet. The Xbee shield for Arduino was designed to interface well with the Arduino Uno, but is not compatible with many of the other Arduino boards, eliminating them as possible options.

Table 5: Arduino Uno specs from Atmel

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

3.3.3 Remote Computation

One of the biggest challenges of the project was the integration of the tablet into the system. While many versions of a basic turret paintball gun already exist, even some with remote control available as a feature, there were no individual cases found that utilized a touch interface tablet. Additionally, most of the previous projects used an onboard system processor of some sort, be it FPGA or microcontroller or even a computer, to handle the image processing computations. Because of the complex nature of these calculations, a large amount of processing power was needed, and since the tablet was already

available for the user interface, it was decided that the task of target acquisition and tracking could be assigned to it as well. This allowed the group to devote the onboard system processor to the simpler tasks of motor control and firing commands.

The chronological order of events is proceed as follows: First the camera captures the images, then the images are transmitted wirelessly to the tablet. The images then are processed as explained in section 4.5.2 Target Acquisition. A user application is open to display the processed images to the user, which consist of the incoming frames with the enhancements added to them by the program. When the user has selected their targets, the targets are outlined by rectangles. The program then calculates its position based on the location of its centroid, and sends this information wirelessly to the microcontroller, which then interprets the signal into commands for the movement of the servos.

Based on the previous projects, it was observed that supplementing the necessary program with a prebuilt library, such as Intel's Open Computer Vision or Aforge.NET library, would greatly reduce the amount of code needed and simplify a number of intricate programming details that the group would like to avoid, since none of the members, all electrical engineering majors, were knowledgeable about computer vision or greatly experienced with programming in general. Extensive research was done to measure the capabilities of each library, and determine which would be the best fit for the RTCDT. AForge.NET is based in C#, and has a lot of filters, which make it an excellent for features such as edge detection and thresholding. It is also widely held to be the easier of the two to use. Unfortunately, it was unclear how exactly this could be implemented on an Android system. OpenCV, on the other hand, is geared towards C/C++ developers, and is useful for more sophisticated image manipulations, such as facial recognition system, gesture recognition, object identification, and motion tracking. It also includes wrappers for languages such as C#, and Java, which will be useful for the transition to the Android platform. One of the other benefits of using OpenCV is the existence of a large support base due to its popularity among developers, where the group can troubleshoot problems and find solutions for common issues.

There were a variety of tablets and tablet systems that were looked into for use. Since it would be involved in both the user interface and the image processing computations, concern for the requirements of both of these processes had to be taken into account. This means a high quality display for viewing of the video streaming from the camera, as well as adequate memory and computing power for processing the images.

The systems brought into considerations were Apple, Android, and Windows 7. Among Apple tablets, the only competition was between the first generation iPad and the second, while Windows had a few different options from which to choose, and Android had the most. Apple uses Objective-C language to write

programs in. Windows can be programmed in C++, which would be compatible with the OpenCV functions. Android is mostly Java based, however it is also possible to run programs written in C or C++. This comparison eliminated the Apple iPad from consideration, leaving the choice between the two remaining options. The Windows tablet had the advantage in that the group would be using OpenCV on a Windows-based PC before moving it to the tablet, so having the same operating system on both would greatly reduce the changes that would need to be made. On the other hand, the Android operating system is much more popular than Windows as a tablet platform, and the large amount of support information available to developers make it an appealing choice for beginning app builders.

The specifications for 3 tablets- Motorola Xoom running Android, Apple iPad 2 running iOS 4.3, and ExoPC Slate running Windows- are shown in Table 6 below for purposes of comparison. On the hardware level, the tablets are all roughly similar in size, with the ExoPC, the Windows tablet, being the bulkiest as well as the heaviest. The tradeoff is a larger display screen, with greater resolution as well. As far as processing speed, the ExoPC leads the Xoom and iPad 2 in the size of its RAM, 2GB versus 1GB and 256MB, respectively. Each of the tablets is Wifi capable.

Table 6: Comparison of Tablets

	Motorola Xoom	**iPad 2**	**ExoPC Slate**
OS	Android	iOS 4.3	Windows 7
Price	$500	$500-$700	$550
Dimensions	9.8x6.6x0.5 in	9.50x7.31x0.34 in	11.6x7.68x0.55 in
Weight	1.6 lbs	1.33 lbs	2.09 lbs
Processor	NCIDIA Tegra 2 Dual Core 1 GHz	A5 Dual Core 1GHz	Intel Atom Single-Core 1.6 GHz
Display Size	10.1 in	9.7 in	11.6 in
Display Resolution	1280x800	1024x768	1366x768
RAM memory	1GB	256 MB	2 GB
Size	32GB	16/32/64	64 GB
Wifi	802.11n	802.11a/b/g/n	802.11 b/ g/ n

3.3.4 Paintball Guns

Paintball guns are designed to use compressed air to expand in the barrel and force paintballs out the end. For the project, the group looked into two types of paintball guns: mechanical paintball guns and electronic paintball guns. Mechanical paintball guns are mechanically activated while electronic paintball guns are battery powered and use a circuit board to fire.

3.3.4.1 Mechanical Trigger Actuation

Mechanical paintball markers fire when the trigger is pulled manually to release a bolt, which is propelled by a spring and forces a paintball in the barrel. Then compressed air expands the barrel and fires the ball out the end, blowing the bolt back to its original position. Mechanical paintball markers need to be cleaned and oiled after every couple uses. Generally, mechanical paintball guns are not as accurate, fast, and consistent as electronic paintball markers. Also, they need to be cocked before they can fire every time, and need high air pressure. Therefore, the same volume of compressed air can fire more shot using electronic paintball guns than mechanical ones. However, mechanical paintball markers are less expensive and easy to set up compared to electronic paintball guns. For the project, since the turret system is controlled by the user wirelessly, mechanical paintball guns are not ideal. In order to fire the gun without any physical interaction, hardware and software implementation would be required. To avoid mechanical related technical issues that might be encountered, the team decided to exclude this option.

3.3.4.2 Electronic trigger Actuation

Electronic paintball markers' triggers are controlled by battery powered circuit boards. The firing mechanism of electronic paintball markers is that, when the trigger is pulled, the voltage applied to the internal circuit board drops to zero, which will release an electromechanical solenoid that releases the trigger. An external circuitry can be used to correct the voltage applied to the internal circuit board to control the firing. Compared to mechanical paintball markers, electronic paintball markers are easier to be implemented in the turret system and less hardware is required. The system could manage the airflow which determines the speed of firing since the circuit board controlling the solenoid is programmable. Electronic paintball markers have a wide price range, from less than fifty dollars to thousands of dollars based on their features. Ideally, the shooting range is assumed to be within 40 meters and firing rate is 15 to 20 balls per second. Also, the group looked at the weight range from 2 pounds to 6 pounds. The weight of the marker was an important factor that had to be considered while designing the turret. Those requirements put the group in the price range from $800 to $1500. Since low-end blow-back markers, which are operated at 150 PSI, do not have internal springs that are hit by a heavy metal hammer to reach 20 balls per second , the group would not be able to program the circuit board, controlling airflow, to reach the speed. This only left the option to purchase a high-end electro mechanical marker.

3.3.5 Airsoft Gun

There are several different types of airsoft guns categorized by their firing mechanisms: spring-powered, gas-powered, and electric. Generally, airsoft guns propel plastic pellets by a piston that compresses a pocket of air and forces a pellet out of the barrel. Spring-powered airsoft guns use potential energy stored in a compressed spring to propel a pellet. There are several advantages of spring-powered guns. Unlike electric guns, spring-powered ones are not affected by weather condition. Since airsoft guns are all mechanical, no gas tank or battery is needed, which decreases the project's expanse. Compared to other types of airsoft guns, spring-powered ones are much cheaper and lighter. However, the biggest concern the team had about using spring-powered ones was that, they are not accurate and not powerful enough for long range shooting. Instead of using potential energy stored in a compressed spring, gas-powered guns store gas under pressure in a liquid form. When the valve opens up, gas is released from the chamber and transforms from liquid to gas, which forces the pellet out of the barrel. Both springers and gas guns are mechanical, which means it would be more complicated to design the firing control if the group decided to use either one of them in the project. In order to pull the trigger, the group might encounter some mechanical technical problem since none of the team members have a thorough knowledge of mechanics.

Automatic electric guns, or AGEs, use a battery to drive electric spring systems in an enclosed box. The gears driven by the motor have a specific torque and speed ratio corresponding to the spring tension. AGEs are much more powerful than the other two types of airsoft guns. At a distance of 15 to 30 feet, they could sting if fired at human skin. At a distance of 10 feet or under, they could cause a slight welt. The firing mechanism of AEGs is similar to electronic paintball markers. Thus, an external circuitry could be used to correct the voltage applied to the motor to control firing. However, those small pellets do not cause noticeable impact on targets like paintball markers do. It would be difficult to see if a target has been marked.

3.3.6 Laser Pointer

Another alternative method for marking chosen targets was laser pointers. Instead of causing any physical damage on targets, laser pointers would simply show if targets have been aimed at accurately. Laser pointers are small portable devices that are intended to be used to highlight something of interest by illuminating it with a small bright spot of colored light. Laser pointers do not require high power consumption. Power is only up to 1000mW. They make a potent signal tool even in daylight and have long range. A laser pointer is much cheaper than a paintball gun. It is also easy to implement in the turret system. However, laser pointers have some potential hazard so the group had to be careful while implementing them. Laser pointers can cause eye injuries directly if it is aimed at eyes. According to U.S. Food and Drug Administration (FDA)

regulations, more powerful lasers may not be sold or promoted as laser pointers, that is, the output beam power must be less than 5 mW.

3.3.7 Wireless Technologies

In this project, two different point-to point wireless communication systems were needed. One was between the turret system and the user interface, and the other one was between the camera and the user interface. Range of transmission, data rate and the network acquisition time were all taken into account while determining which protocols would be most sufficient for the project. This project was meant to be operated in real-time manner. Therefore, network acquisition time became crucial. Ideally, users are able to control the turret from 10 meters away in all directions and respond to the immediate danger in 2ms. Since only small packets of data are transmitted from the user interface to the Arduino board, low data rate transmission is sufficient for the communication between the turret and the user. Massive image data information would be sent from the camera to the user interface, high data rate was essential to keep the system operated in real-time manner. The group looked into the most commonly used wireless protocols in other similar projects and other potential methods for the projects. Based on the complexity of implementation and project requirements, all the possible solutions were narrowed down to Bluetooth, ZigBee, and wireless USB, which are compared in Table 7 below.

Table 7: Wireless Module Specification

Specification	***Wi-Fi***	***Bluetooth***	***ZigBee***
Data Rate	54 Mbits/s	3 Mbits/s	240 Kbits/s
Range	100m		
Networking Topology	Point to Hub	Ad-hoc	Ad – hoc/ PTP/mesh
Operating Frequency	2.4 GHz		
Complexity	High	High	Low
Power Consumption	High	Medium	Ultra Low
Network Acquisition Time	3 - 5s	< 10s	30ms

3.3.7.1 Bluetooth

Bluetooth uses radio frequency, unlicensed ISM (Industrial, Scientific, and Medical) band (2.4– 24.8 GHz) to send information between two points. Bluetooth is power-class-dependent with short transmission ranges. It is usually

set up for personal area networks between PCs and keyboards or mice, cell phones and headset, or TVs and music players. Bluetooth has low energy consumption and medium speed transmission rate with high levels of security. One advantage of using Bluetooth is that a master Bluetooth device can communicate up to seven other mobile devices with high security. However, Bluetooth is not as reliable as the other two potential protocols that were considered. Bluetooth functions better between two stationary mobile devices with no interrupts that could possibly occur due to open space randomness. For the project, it was expected that moving objects and stationary obstacles would come between the units. Also, Bluetooth is not fully developed for embedded devices. Most importantly, the transmission range did not meet the requirement. Three classes of Bluetooth radio are shown in Table 8, below, for comparison.

Table 8: Bluetooth Classification

Class	***Power Consumption***	***Transmission Range***
Class 3 radio	1 mw	1m
Class 2 radio	2.5 mW	10m
Class 1 radio	100 mW	100m

3.3.7.2 ZigBee

.ZigBee is a protocol that can directly be embedded into various applications at the frequency 2.4GHz. Like Bluetooth, it uses radio frequency and can link with multiple nodes. It is an ideal mesh network for embedded devices. It has low power consumption, low network acquisition time, and low data rate. It is the most suited protocol for monitoring and controlling applications when no massive data information needs to be transmitted. In the project, since image data information are sent directly from the camera to the user interface, the wireless communication system between the user interface and the Arduino board does not require high data rate. ZigBee's low data rate and its capability of transmitting periodic and intermittent data are sufficient for the specific part of the project.

ZigBee protocols support both non-beacon-enabled and beacon-enabled networks. For the case of the RTCDT, a non-beacon-enabled network was more desirable than a beacon-enabled network. Carrier Sense Multiple Access/ Collision Avoidance mechanism was used in this case. The user interface would only send out single signals to the Arduino board when user input is detected. Most of the time, both nodes would remain in sleep mode until acknowledgement is received. After the acknowledgement is detected, nodes will send out commands and returns to sleep mode. ZigBee protocols minimize the time of devices in active mode, thus, reduce the power consumption effectively. Unlike

Bluetooth, ZigBee has such a low network acquisition time 30ms, which is ideal for real-time manner applications. The project required the turret system to be responsive to the user command to fire upon chosen targets.

ZigBee supports three different topologies: point-to multipoint, mesh, and point-to-multipoint or mesh. Table 9 compares the three topologies. For this specific project, two wireless communication systems were considered. However, since ZigBee provides numerous options and features. It might be possible to use one protocol and link to other slave addresses to make the project more efficient and work as a whole complete system. Based on the frequencies, RF line of sight range, transmission power, receiver sensitivity, RF data rate and MSRP, the options were narrowed down to three XBee/RF modules that could be applied to the project. However, all the modules require line of sight. That is, the communication might be blocked by intervening objects. The group had tested the stability and the reliability of transmission by varying the surroundings.

Table 9: ZigBee Wireless Module Specification

Specification	***XBee 802.15.4***	***XBee ZB***	***XBee Digi Mesh 24***
Topology	Point-to-Multipoint	Mesh	Mesh
Frequency	2.4 GHz		
RF line of sight range	90m	120m	90m
Transmission Power	0dBm	3dBm	0dBm
Receiver Sensitivity	-92dBm	-96dBm	-92dBm
RF data rate	250Kbps		
MSRP	$19.00	$17.00	$19.00

3.3.7.3 Wireless USB

Wireless USB is the easiest communication module that could possibly be implemented in the communication system between the camera and the user interface. It provides high data rate and "plug-and play" scenario. No complicated hardware implementation is required. In the project, in order to respond to immediate danger that is monitored by the user through the camera, time delay and data are important factors. Any lags decrease the aiming accuracy of the marker. According to USB Implementers Forum, wireless USB can perform up to 480Mbps within 3 meters transmission range and 110Mbps within 10 meters transmission range(Wireless USB from the USB-IF). To have the camera talk to

the tablet, the camera has to have one or more USB ports. Also, the selected tablet had to have a built-in operating system that can understand USB.

3.3.8 Motors

There are numerous of motors in the markets that could possibly be used for the project. Different motors have different control methods and drivers. To determine what type of motors the turret should use, the group has to understand the mechanism of the control system. An open loop system takes in input data and amplifies the signal, then passes it to the output. The output is not compared with the reference input. Thus, to each reference input there corresponds a fixed operating condition. For example, when the coordinates of a target are sent to the system as an input, the motor will take the amplified output signal and move to the specified position and remain at its position until the next coordinates is sent. Closed loop system takes the output through a specific amplifier back to the system as to reduce the error and bring the output of the system to desired value.

3.3.8.1 Stepper Motors

Stepper motors have no brushes or contact and are operated in open loop system. They use an electromagnetic field to rotate the armature magnet. Instead of rotating continuously like a conventional motor, a stepper motor moves in discrete steps. A rotor only rotates a certain number of degrees at a time. Since a full rotation is divided in to a large number of steps, the accuracy of positioning is relatively high. Stepper motor controllers takes step pulses and direction signals, and deliver the data to drivers. Basically, drivers receive low-level signals from the control signal and convert them into pulses to run a motor. Different types of stepper motors need their own drivers. Even though steppers are said to be a marvel in simplicity, it takes decent knowledge of mechanics to control the movement accurately based on the speed, load, toque and inertia. For this project, closed loop control system would be more sufficient than open loop control system. It was expected that the marker would restore to its center position after every execution. Closed loop system would save some hassle of programming the microcontroller to restore the motor to its center pointer after every execution. This criteria left the group with servo motors, which is easy to implement and easier to program compared to stepper motors.

3.3.8.2 Servo Motors

Servos are controlled by sending them a pulse of variable width within duration in a closed loop system. A servo is essentially a positionable motor, which takes two inputs: the current position and the desired position. The control wire is used to send this pulse, which has three parameters, minimum pulse (ground), maximum pulse (power) and a repetition rate (command). Given the rotation constraint of the servo, neutral is defined to be the position where the servo has

the exact same amount of potential rotation. Different servos have different constraints on their rotations but they all have the same neutral position, which is always around 1.5 milliseconds. Servos are commanded through “Pulse Width Modulation”. Basically, the width of a pulse defines the position. To move to motor 90 degrees and hold in that position, a 1.5 ms pulse is required to be sent to the servo. Also, the command has to be send every 20ms or at frequency 50Hz. It is possible to damage a servo if commands are sent at improper frequency. Unlike stepper motors which can have changing speed by varying the voltage applied to drivers, most of servos have fixed speed rate in degrees/second.

The team decided to use two motors controlled in closed loop, one for horizontal movement and one for vertical movement. As mentioned before, servos are commanded through PWM. Neutral is defined to be the position where the servo has the same amount of rotation in clockwise is as it does in counterclockwise. The angle is determined by the duration of a pulse that is applied to the control wire. Servos are expected to receive a pulse signals at frequency 50Hz. The width of a pulse determines how far the motor will turn. For example, 1.5 ms turns 90 degrees (neutral position). When servos are commanded to move, they move to the position and remain there. If an external force pushes against the servo while the servo is holding its position, the servo will resist from moving out of the position. The maximum amount of force that can be exerted is the torque rating of the servo. Repetitions of the position pulse are needed for the servo to remain at the desired position.

When a pulse less than 1.5ms is sent to a servo, the servo rotates to a position and holds its output shaft some degrees counterclockwise from the center point. When a pulse is wider than 1.5ms, the motor rotates in clockwise direction. The functions of a servo are commanded by the minimum and the maximum width of a pulse. Generally, the minimum pulse is 1ms and the maximum pulse is 2 ms wide. The other parameter is turn rate, which varies from servo to servo. It is the time that is required for a servo to change its position to another. The worst scenario is when a servo is holding at its minimum rotation and commanded to go to the maximum rotation. This might take seconds on a high torque servo.

3.3.9 PCB

A printed circuit board (PCB) is a component made of one or more layers of insulating material with electronic conductors. The insulator is typically made on the base of fiber reinforced resins, ceramics, plastic, or some other dielectric materials. Currently, the main generic standard for the design of printed circuit boards, regardless of materials, is IPC-2221A. Whether a PCB board is single-sided, double-sided, or multilayer, this standard provides rules for manufacturability and quality such as requirements for material properties, criteria for surface plating, conductor thickness, component placement, dimensioning and tolerance rules, and more.

The width of the circuit conductors should be determined based on the temperature rise at the rated current and acceptable impedance. The trace should not melt during short surge currents that can develop in the circuit. This requires sufficient cross-sectional area of copper as a function of amps and seconds. The spacing between the PC traces is determined by peak working voltage, the coating location of the circuit, and the product application.

3.3.10 Cameras

To capture the visual information of the target field, a camera was required. Based upon their convenience of use, the power consumption, and the applicability to the present application, several classes of camera were considered.

3.3.10.1 USB Webcams

Under first consideration was the Logitech QuickCam Pro 4000 that the group had prior to the project. The parameters under investigation were the frame rate of the camera, the resolution of the camera, its power consumption, and the ease with which it can be mounted onto an existing hardware framework. The QuickCam Pro 4000 has a frame rate of 15 fps (frames per second) and a resolution of 640x480 pixels. Since the field of targeting was expected to be a room no more than 40 meters deep and 20 meters wide, and the average human is .7m wide, the resulting pixel width was

$$\frac{.7m}{20m} * 640\ pixels = 14.93 pixels \approx 15\ pixels \quad (3)$$

Thus there were plenty of pixels - even at maximum range - with which to calculate the centroid of the target for the maximum probability of successful impact. That vastly reduced amount of data throughput – compared to a 1600x1200 pixel camera - meant faster processing and therefore quicker response. That made this camera, despite its age and simplicity, attractive for this project. Unfortunately, because the system needs to be able to track moving targets, and a frame-rate of 15 fps made moving objects appear blurry at a very low velocity, it was clear that this camera would not work in the final project.

Since the existing QuickCam was too old, it was logical to look at a newer camera of a similar type. The choice examined was the Logitech QuickCam Orbit AF(Logitech). This camera featured a frame rate of 30 fps, which meant a sufficient frame rate with which to capture the images of moving targets without blurring. The resolution maxed out at 1600x1200, but the camera could be set for lower resolution, allowing the lower data throughput which would be desirable for such an application. Less desirable, however, was the rounded plastic housing present which this device featured; it did not appear sturdy and seemed as though it would be difficult to mount. Still, the relatively low price point of $129.99

could potentially offset the worries about its housing. After exhaustive searching, there were no specifications about power usage available online for any USB cameras; however, in investigating the standards for USB devices, the maximum is 2.5W and thus, this figure was used in the group's power considerations.

3.3.10.2 Infrared

Infrared cameras are attractive because they operate in the absence of visible light, but detect the motion of objects based upon their near-IR heat signatures; this is a feature which all of the project's targets have since it is designed to track humans.

The first IR camera investigated was a Sony CM307. This camera is small and cheap, but it was discovered that it used a BNC video-out rather than USB, and an external converter would add needless complexity. Converters from ambery.com are reasonably priced at $28.00 though, so the option of using such a device was not immediately overturned. However, at a realistic price point IR cameras have long range blurriness issues, so the group discovered that usable cameras are initially cost-prohibitive. An alternative, made from information available on hoagieshouse.com (Hoagieshouse.com), shows that it is possible to modify a visible-spectrum camera to operate in the infrared spectrum. The procedure is to disassemble the camera, find the IR filter, and remove it; then to manufacture an IR-admitting and visible-spectrum blocking filter from a black area of film negatives; and finally, to install this filter, and reassemble the camera. This procedure is both time-consuming and fraught with uncertainty, and thus the group chose to avoid it. The mounting of this camera appeared to be more difficult than the group desired, which also fed into the decision not to use this camera.

The high-cost option, which would be the most desirable if cost is disregarded, presented the group with an Edmund Optics NT56-567(Edmund Optics). This camera outputs 768x494 pixels, which was in the range the group desired, and remarkably, operated in the range of 60-100,000 fps. The mounting for this camera is a pre-threaded socket on one side of the device. Clearly, by the specifications, this device would have met the requirements of the project. However, the camera cost $1,995.00, and was immediately rejected as a possibility in the prototype design, since it would have occupied just under a third of the budget allotted to the group by Workforce Central Florida.

At the reduced cost of $495.00, maxmax.com (maxmax.com) offers a modified version of a Sony DSC-S980 which operates in the IR-only range. However, the battery life when operating in video mode is greatly reduced, and the cost – though lower than the NT56-567 - is still somewhat prohibitive. Additionally, though it is likely a website error, the frame-rate specification is only 1 fps, which clearly did not meet the project's needs, and thus this camera was rejected as well. Also, this was a handheld consumer-grade camera with no permanent or

secure mounting hardware readily available, and thus did not immediately appeal.

Other options explored included the VH HK High Tech Limited brand's VGSION USB Camera, available at alibaba.com (alibaba.com), which presented a frame rate of 30 fps and VGA resolution, which is the 640x480 previously determined to be desirable for this project. At a price point of $46, this camera presented a very viable option, though there was some concern about the lead time of acquiring the product from the supplier in Shenzhen, China. The base of the camera featured three mounting holes which could easily be mounted onto a wood framework.

3.3.10.3 USB 2.0 Hi-Def

Next considered was a line of uEye USB cameras available from IDS Imaging (IDS Imaging), with very high frame rates. These cameras were all connected via USB, as the heading suggests, so power was not a primary consideration when comparing them. All cameras were 640x480 pixel resolution; the primary differences between them were the frame rates and mountability; the group did not seek to explore custom mounting hardware at extraordinary depth. The UI-2410ME with USB 2.0 interface features a CCD sensor from Sony. It has a remarkably high maximum frame rate, 75fps, and is manufactured with six mounting holes such that it could easily be screwed into wood or bolted to a steel frame. Compared the other options, this was a reasonable choice without price considerations; pricing was not available at the time of this writing, so it could not be reasonably compared.

Also from IDS Imaging, the USB 2 UI-1225LE-C was under consideration. This camera has a resolution of 752x480, a maximum frame rate of 87 fps, and a small footprint as well as low weight. The resolution of this camera was deemed acceptable, and the high frame rate was desirable, but there is no existing mounting hardware, so upon consideration it was ruled out. The UI-2220ME-C is nearly identical to the first-considered UI-2410ME, except that this model has 52 fps and a resolution of 768x576, and it is also available in a black-and-white version, which offers the fetching quality of potentially much less data process demands, but which does not allow for the attractive user interface that the group desired. Aside from the aesthetic quality, having color enables a user to more easily distinguish between targets, so the black-and-white option was ruled out. The footprint of the UI-2220ME-C is identical to the UI-2410ME, so mounting concerns were obviously low, identically to that model.

The UI-1120ME-M has the same footprint as the above model, but offers an interesting bonus: this model operates in the visible spectrum, but also operates well into the IR spectrum, which would potentially allow the group to investigate both of these options in one camera and compare the results. The UI-1120ME-M

features a resolution of 768x576 and a frame rate of 50 fps; so, without the IR option, it is very similar to the UI-2220ME-C.

Some higher resolution cameras were investigated as well. For example, the UI-1640SE-C, which has an Aptina CMOS sensor in 1.3 Megapixel resolution (1280x1024 pixels), also from IDS Imaging. The high resolution apparently limits the frame rate available on this model and several others in that resolution range, as the highest available was 25 fps. The group was unsure as to whether this would be below the minimum frame rate needed, but it was determined that gambling on such an item working was a poor design decision. The casing of this model also did not lend itself well to mounting. While there were other models of similar resolution available and better mounting options, none had a frame rate with which the group was comfortable, so it was decided that these would not be used, and that frame rate was a higher priority than high resolution.

3.3.11 Power Sources

The power sources investigated were batteries, solar power, wall AC, and generators. Each of these has their own advantages; a summary of the group's research is shown below.

3.3.11.1 Batteries

Because of the different firing platforms considered, there was a multitude of battery types that had to be considered to power them. The types of batteries considered are: alkaline, nickel-metal hydride, nickel-cadmium, lithium-polymer, metal-chloride, zinc-air, zinc-mercury oxide, and silver-zinc.

Although disposable, alkaline batteries are the most common type used in small household gadgets and are supplanting carbon zinc and zinc chloride batteries as the most common technology. There is not much difference from brand to brand since each battery uses the same chemicals. Comparison tests was done by Consumer Reports between different alkaline brands and showed that the best and worst batteries only differ between 9% - 15%.

Rechargeable alkaline batteries have good capacity, but limited rechargeability. However, they do give out higher voltage than NiMH (nickel-metal hydride) batteries. This means that they perform well with devices that take in multiple batteries. For instance, LED flashlights produce brighter light with the alkaline batteries than with NiMH. Infrequent use along with high self-discharge of can make NiMH batteries go dead on their own between periods of use.

NiMH batteries offer great capacity and long recharging life, but are slightly more expensive than alkalines. One possible downside with the NiMHs and NiCads (nickel-cadmiums) is that they both put out less voltage in comparison to alkaline batteries. This means that devices that work with multiple batteries may not work

with NiMH or NiCad batteries. Another downside with both batteries is that they quickly self-discharge so they lose power even if they are not being used. A positive distinction about the NiMH battery is its discharge path when compared to alkaline. The voltage and current of a NiMH starts high and does a good job of holding this rate for about 75% of its battery use; the current will then will gradually diminish at the end of the battery's capacity.

NiCad (nickel-cadmium) batteries deliver good capacity, long life recharging, are an economically conservative rechargeable battery. NiCads have approximately 25% less capacity than NiMHs; a NiCad Sub C cell can be 1200 or 1800 mAh while NiMH battery starts at 2400 mAh and goes as high as 3800 mAh. NiCad has higher current delivery per cell however, having 3 units as opposed to NiMH which has 5 units.

Similar to NiMH when comparing it to alkaline, the downside with both NiMH and NiCads batteries is that the voltage output is lower than those of alkalines. One advantage of Ni-Cads over NiMH is that it holds its discharge rate as well. NiCads are a good choice for airsoft and paintball applications since they continue to put out a high current until almost completely dead. NiCads put out a fairly consistent 1.2V for the length of their charge. As soon as the charge is spent, the voltage drops rapidly. The voltage would be around 1.0 to 1.1V.

Lithium polymer batteries feature high capacity and less combustibility than lithium-ion, but are slightly more expensive. Unlike lithium-ion cylindrical, or prismatic cells, which have a rigid metal case, polymer cells have a flexible, foil-type (polymer laminate) case, but they still contain organic solvent. Lacking this metal battery cell casing, the battery is lighter and can be specifically shaped to fit any device it should need to power. Because of the denser packaging without intercell spacing between cylindrical cells and the lack of metal casing, the energy density of Li-poly batteries is over 20% higher than that of a classical Li-ion battery and store more energy than nickel-cadmium and nickel-metal hydride batteries. Li-Poly is widely considered to be the best power solution for airsoft Electric Guns. They rarely harm these guns, provided that the right battery pack is used. Battery packs that are commonly used are 7.4V and 11.1V; the 7.4V pack is usually used on stock AEGs, and the 11.1V is used for more tuned or upgraded setups.

3.3.11.2 Solar

Solar power was investigated for the setup, but was summarily rejected for a few reasons. First, since the system would require the panels themselves, it would be bulky and expensive. Notably, there would still be a need for the battery setup already required; it would simply be way to charge this setup. Since this primary version of this concept device would not be field-deployed in an environment that lacks AC power, it was considered to be out of the scope of the project. Consider that simply to power a USB device is 2.5watts, and reasonably-sized USB solar chargers (reasonable meaning readily portable and light) only produce 4 watts, a

charger of that type could not power a camera and a laser at once. Additionally, since an 11.1V battery pack typically puts out 1800mA, that means that to match such a battery pack, five chargers would be needed. Thus, solar power was not seriously considered for the current project.

3.3.11.3 AC Power

It was determined early-on in the project that a wireless USB hub would be used, and this device runs on US-Standard wall-outlet power. Similarly, the user interface tablet that would allow control of the system would have battery power, but would fundamentally be charged via AC; and most importantly, the servos that control the aiming of the targeting system would draw excessive amounts of current for short periods of time, which would either require large lithium-ion batteries, or readily available DC after conversion from wall-AC. Since standard wall AC is readily available, the group opted to use it.

Overall, the project uses a consumer-grade power strip. Into this is plugged the adaptors for the Arduino and the driver board. The Arduino runs on a 9V DC “wall wart” style adaptor with a 2.1mm barrel plug, utilizing a positive tip. The driver board required 2.857A of current and 3.3V to drive the voltage regulators, which in turn drove the servos. The laser pointer requires no extra power source, as it is actuated through the Arduino.

The User Interface Tablet is not powered by the system during operation – it is charged separately prior – so its power consumption is not taken into consideration for this portion of the project.

3.3.11.4 Generators

Since the system was to be designed around the use of AC power, the use of generators is trivial; they fundamentally produce the same type of AC power that is produced by the United States’ electrical grid. Thus, in a final version of a system such as this, it is viable to use a generator as an emergency replacement, but in a prototype it adds needless and bulky redundancy.

4 PROJECT HARDWARE AND SOFTWARE DESIGN DETAILS

4.1 Initial Design Architecture and Related Diagrams

4.1.1 Hardware Block Diagram

The hardware block diagram, shown in Figure 6 below, illustrates the basic connections of the hardware components. The system is powered through AC, with the 5V regulator powering the servos and microcontroller, and the 3.3V regulator powering the XBee and laser pointer. The camera is powered separately through its own 5V AC/DC converter.

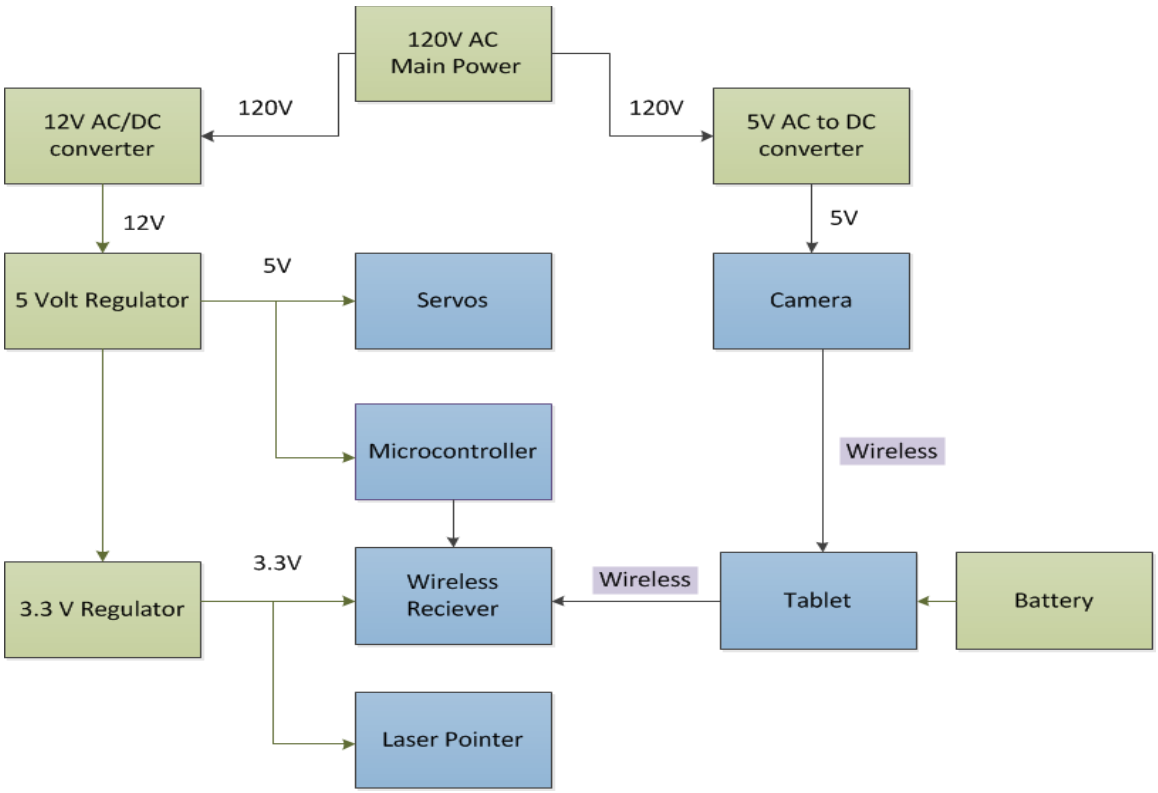


Figure 6: Hardware block diagram

4.1.2 Software Block Diagram

The initial software block diagram was created from the conceptual design at the outset of the project, and can be observed in Figure 7 below. Note that some of the features were later altered. The basic inputs to the system programs can be seen in the four inward pointing arrows. Orientation of the gun alerts the program to whether the gun is positioned in its default orientation, which is in the center of the visual frame pointing straight ahead, or whether it is currently in the process of tracking a target and has previously been positioned for aiming and firing. This information is used by the program to determine how far to move the turret from its current position until its ready to fire again. User Tactile Input reads the user touch selection to determine either which target to aim at through manual selection of a point on the target field, or which target to fire at by pressing the colored button onscreen that corresponds to the matching target outline. Visual Input is relayed both from the wireless camera and the linear sensor that works as part of the range finding system. The camera input is used in the rendering of the user interface to provide visual user feedback, while the sensor aids in calculating the range. This is then used to alter the angle of tilt of the gun, to compensate for the physics of projectile motion. Finally, a sensor reading the pressure in the nitrogen tank is sent to the tablet and available to the user through the user interface.

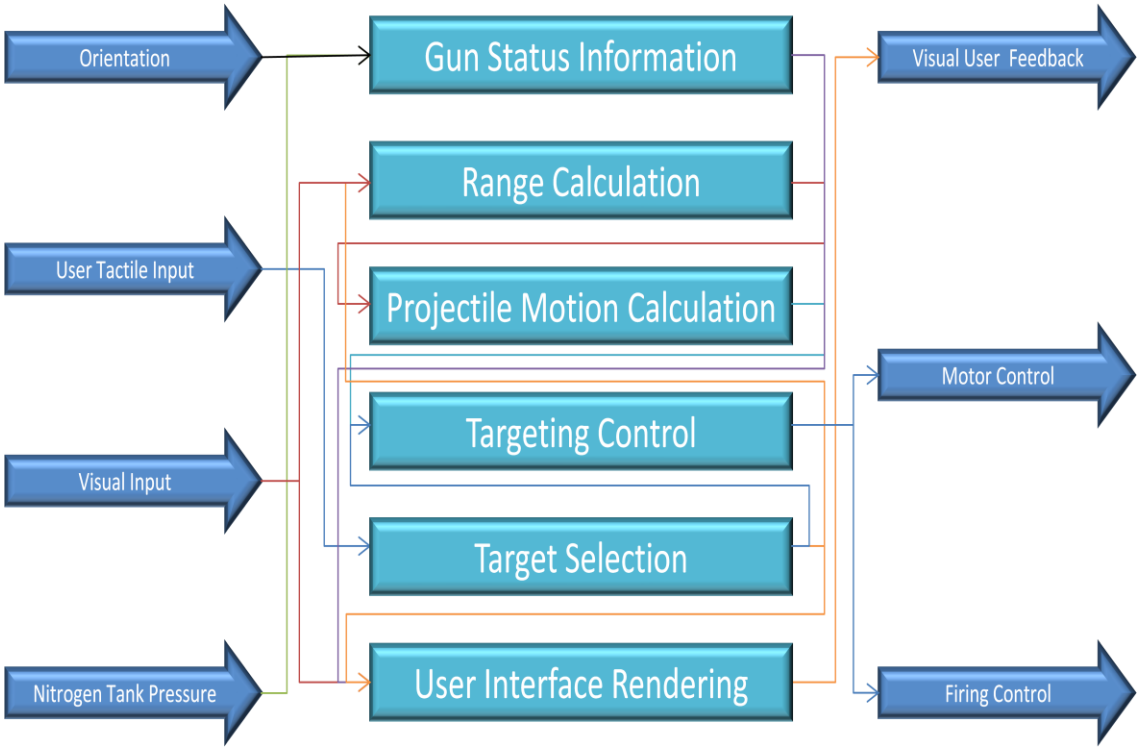


Figure 7: Software block diagram

4.1.3 Turret Design

Since any field of fire is three dimensional, but can be viewed and targeted in two dimensions, a turret must have control in both of these. Thus, this project required both a pan and a tilt control. To accomplish this, the group used a turret armature from paintballsentry.com called the Medium Turret, which was pre-machined to directly accept servos from servocity.com. This would control its base. The firing mechanism, which was finally determined to be a laser pointer for this prototype, would be mounted to the upper part of the armature via hose clamps for ease of adjustment. The group created a representation in Google's SketchUp software to allow for easy design of the systems around the turret itself, as shown in Figure 8 and Figure 9 below.

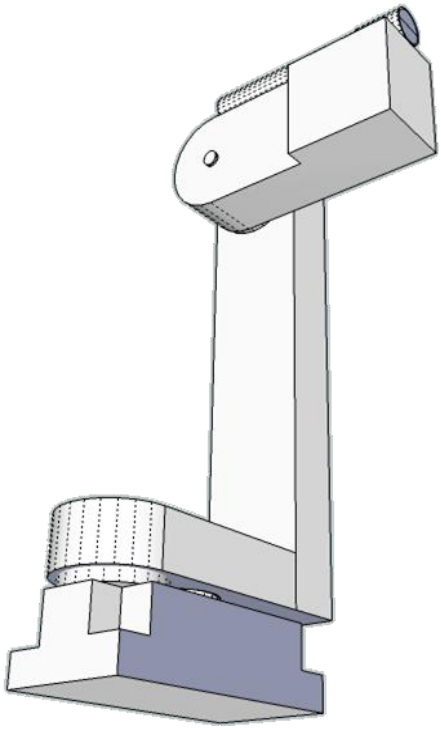


Figure 8: Turret Armature, view 1

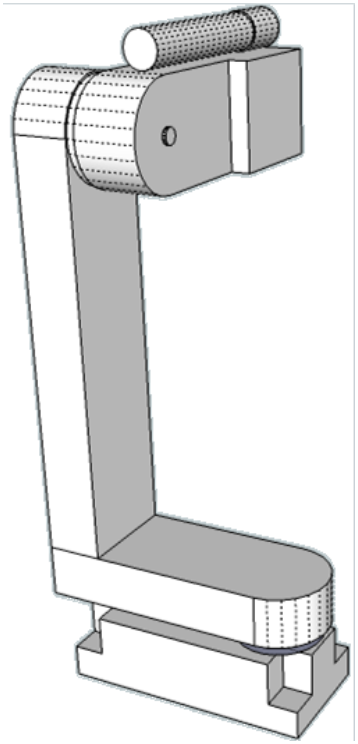


Figure 9: Turret Armature, view 2

The system's User Interface was designed in OpenCV, generously borrowing code from the OpenCV wiki, which is protected under the GNU General Public License, which allows for users to "…modify your copy or copies of the Program or any portion of it…" This was a primary motivator in the decision for the group to use OpenCV, the secondary being that, because it is free and open, there is so much source code available to use. Additional resources were the cvBlobs library and Videoinput library.

The user interface is the heart of this system, and like any vital structure, it had be robust. The group conceived of its use in terms of Occam's razor: roughly, the simplest complete solution is the best one.

4.2 User Interface

Building the Window consists of six functions, outlined below with their corresponding input variables.

Function: `CreateTrackbar`

Parameters: `trackbarname, string winname, value, count, onChange, userdata`

`trackbarname`: Name of the created trackbar.

`winname`: Name of the window which will be used as a parent of the created trackbar.

`value`: The optional pointer to an integer variable, whose value will reflect the position of the slider. Upon creation, the slider position is defined by this variable.

`count` The maximal position of the slider. The minimal position is always 0.

`onChange` Pointer to the function to be called every time the slider changes position. This function should be prototyped as `void Foo(int,void*);` , where the first parameter is the trackbar position and the second parameter is the user data (see the next parameter). If the callback is NULL pointer, then no callbacks is called, but only `value` is updated

`userdata` The user data that is passed as-is to the callback; it can be used to handle trackbar events without using global variables

Function: `getTrackbarPos`
Parameters: `trackbarname, winname`

`Trackbarname` Name of the trackbar.

`winname` Name of the window which is the parent of the trackbar.

Function: `imShow`
Parameters: `winname, image`

`winname` Name of window

`image` Image to be shown

Function: `namedWindow`
Parameters: `name, flags`

`name` Name of the window in the window caption that may be used as a window identifier.

`flags` Flags of the window. The only supported flag is `CV_WINDOW_AUTOSIZE` . If this is set, the window size is automatically adjusted to fit the displayed image (see *imshow*), and the user can not change the window size manually.

Function: `setTrackbarPos`
Parameters: `trackbarname, winname, pos,`
`trackbarname` and `winname` are as above.

`pos` Sets the trackbar position.

Function: `waitkKey`
Parameters: `delay`

`delay` Delay in milliseconds. 0 is the special value that means "forever"

4.3 Targeting Control

4.3.1 Pan-and-Tilt Control

Motor controllers, for both pan motor and tilt motor, receive the coordinates of a chosen target, which was analyzed and transformed from analog to digital data by the Arduino using function analogWrite(). In order to have smooth movement on the motors, the group utilized a multichannel PID controller, which was implemented in the servo motors, to eliminate as much error as possible before the signals were sent from the user interface to the motor driver. A closed loop PID controller has better control of the servo system to improve transient response time, reduce the steady state error, and reduce the sensitivity to the load parameter (inertia). Improving transient response time implies that the bandwidth had to be increased. The faster the response time is, the quicker the system will settle. Steady state error indicates the accuracy of the servo system. Load parameters represent the tolerance of fluctuation in both input and output parameters. Servo control could be broken into two fundamental classes, command tracking and the disturbance rejection characteristics of the system. The servo control system is shown in Figure 10.

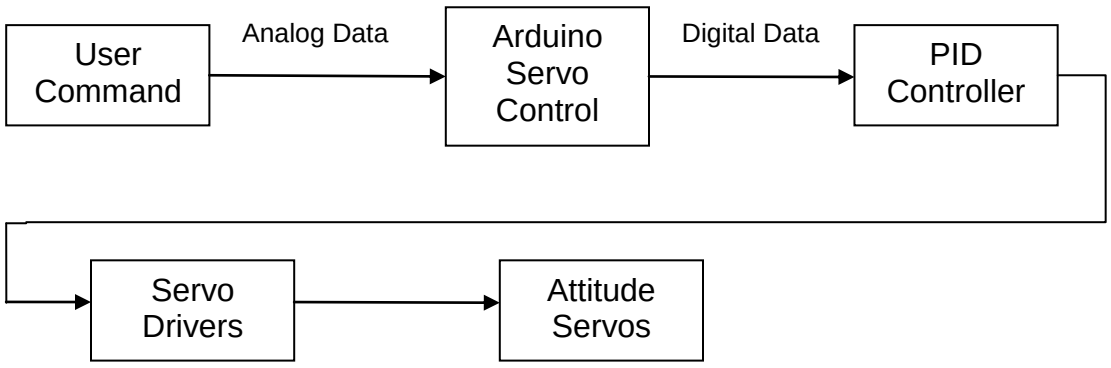


Figure 10: Servo Control System

Command tracking is that, when the coordinates of a designated target are fed into the system control module, the servo control addresses how smooth the movement of a motor is and how accurately the user commands are followed. This part of the servo control is referred as “Feedforward control”. This can be interpreted as what internal commands are needed for the user’s motion commands to be followed without any error.

Disturbances can be the torque disturbances on the motor shaft or false motor parameters estimations used in the feedforward control. Disturbance rejection characteristics address the prediction of the needed internal commands for zero following error. Disturbance rejection control reacts to unknown disturbances and modeling errors. A combination of both feedforward control and disturbance rejection control provides the best overall performance.

After errors have been filtered out by PID controller, signals corresponding to the designated target are fed into the motor drivers. These two motor drivers have the exact same two functions, amplification and calculation of the cycle rate. In order for motors to move, the voltage fed from the PID to the motors is increased to meet their operating voltage. Cycle rate determines how long motors remain at their positions. Motors position themselves back to their center position after a firing command has been received and executed. That is, when the user interface send a firing command to laser pointer driver, it also stops updating the coordinates of designated targets to Arduino (No voltage is applied to motor drivers, therefore, motors orientate themselves back to center points.) The compensator block diagram is shown in Figure 11.

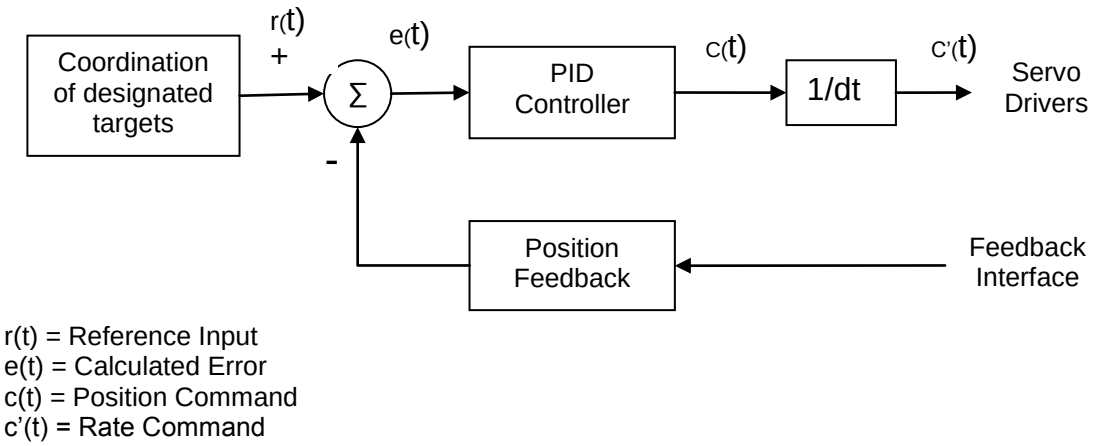


Figure 11: Compensator Block Diagram

$$\text{Rate Command} = \frac{PID\ Position\ command - Position\ feedback}{Cycle\ Period} \quad (4)$$

4.3.1.1 Arduino Servo Control Library

One of the reasons why the group decided to use the Arduino Uno microcontroller was that Arduino is an open-source electronics prototyping platform. It has pre-existing powerful libraries that help with various tasks. For the project, the Arduino servo control library is used, which supports up to 12 motors on Arduino Uno. Unlike complex delay or timer/interrupt sequences in pseudo code, the coding in the Arduino environment is much simpler. The Arduino's programming language makes PWM easy to use; simply call analogWrite(pin, dutycycle), where the duty cycle is from 0-255 and the pin is one of the PWM pins (3, 5, 6, 9, 10, or 11). The analogWrite function provides a simple interface to the hardware PWM, but does not provide any control over frequency. Even though the function name is called analogWrite, the output is a digital signal. Below is a brief overview of the servo library functions that is implemented:

Function: `attach()`
Parameters: `servo, pin, min`
Syntax: `servo.attach(pin)`
`servo.attach(pin, min, max)`

`attach()`: Attach servo variable to a pin
`servo`: A variable of type servo
`pin`: The number of the pin that servo is attached to
`min`: The pulse width, in ms, corresponding to the minimum (0 degree) angle on the servo (defaults to 544)
`max`: The pulse width, in ms, corresponding to the maximum (180 degree) angle on the servo (defaults to 2400)

Function: `write()`
Parameters: `angle`
Syntax: `servo.write(angle)`

`write()`: Write a value to the servo, controlling the shaft accordingly. On a standard motor, this will set the angle of the shaft. On a continuous rotation servo, this will set the speed of the servo.

`angle`: The value to write to the servo. 0-180

Function: `writeMicrosecond()`
Parameters: `uS`
Syntax: `servo.writemicrosecond(uS)`

`writeMicrosecond()`: Write values in microseconds to the servo. On a standard servo, a parameter value of 1000 is fully counter-clockwise. 2000 is fully clockwise
`uS`: The value of the parameter in microseconds

Function: `read()`
Parameters: `angle`
Syntax: `servo.read(angle)`

`read()`: Read the current angle of the servo (the value passed to the last call to `write()`

Function: `attached()`
Parameters: `pin`
Syntax: `servo.attached(pin)`

`attached()`: check whether the servo variable is attached to a pin

Function: `deattached()`
Parameters: `pin`
Syntax: `servo.detached(pin)`

`detached()`: Detach the servo from its pin. If all servo variables are detached, then pins 9 and 10 can be used for PWM output with `analogWrite()`

As mentioned earlier, servo motors have three parameters, minimum pulse (ground), maximum pulse (power) and a repetition rate (command). The power wire is connected to the 5V pin on the Arduino board. The ground wire is connected to a ground pin on the Arduino board, and the signal wire is connected to a digital pin on the Arduino board. Since a PID controller is built in the motors, the motors are directly connected to the digital pins of ATMega328.

4.3.1.2 Servo Motors

To decide which servo motors would be most successful for this particular case, it was necessary to determine how much torque was needed and how fast the servo system operates. Another factor to be taken into account was the cost. Since the group used laser painting instead of a paintball marker, the system only need a little torque and responsive servos for the system to achieve its best performance. There are many websites that carry the same models but price varies from one site to another. After web surfing and comparing the prices and performances of motors, the group decided to choose two HS 325-HB BB Deluxe digital servos motor from servocity.com at price $12.99 each. This motor can handle torque up to 51 oz/in or 3.7kg/cm and has standard operating voltage from 4.8v to 6v with speed from 0.19 second per 60 degrees to 0.15 second to 60 degrees. Since the weight of the laser painting that is implemented in the system is ultra light, this motor that was chosen is able to provide enough torque so that it does not burn out under the stress of responsive tracking of designated targets.

4.3.1.3 PID Controller

As mentioned earlier, in order to have the servo motors run as smoothly as possible, the servos had to be connected to a PID controller; having closed loop control of a motor with truth encoder feedback increases the degree of the

accuracy. A PID controller generally computes three mathematics functions to tune a system. Proportional term (P) computes function K_P * V_{error} , where V_{error}= V_{set} (designated destination) – V_{sensor} (current position). This is the main drive of a control loop; K_p reduces a great amount of the overall error. Integral term (I) computes function K_I * $\int$ V_{error} dt. This function sums even a small error over time and produces a drive signal that is large enough to move the system toward a smaller error. In other words, this reduces the final error in a system. Derivative term (D) computes function $K_D * \frac{d\text{Verror}}{dt}$. This function has no effect on final error. It counteracts with proportional function and integral function when the output changes quickly and helps reduces overshoot and ringing. Each term plays an important role in the system. For the project, the derivative term is relatively crucial since the turret will be tracking designated targets instantaneously. That is, the output (current position) will keep changing constantly. Figure 12 illustrates the servo topology.

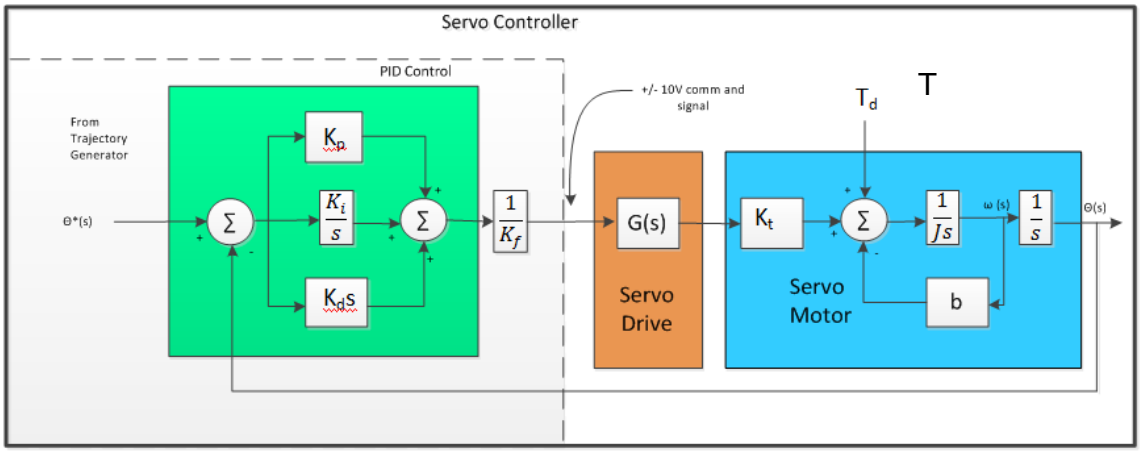


Figure 12: Basic PID Servo control Topology from Parker Hannifin

There are two primary ways to select the PID gains. One is using a trial-and-error method, while the other is an analytical approach. The trial-and-error method requires personal experiences with other servo systems. Since none of the team members had decent knowledge of controlling servo systems, the group decided to take an analytical approach. That is proposed by Ziegler and Nicolas. Their approach can be broken down into two steps.

Step 1: Set K_I and K_D to be zero. Excite the system with a step command. Slowly increase K_P until the shaft position begins to oscillate. At this point, record the value K_P and set K_O equal to this value. Record the oscillation frequency, f_O .

Step 2: Set the final PID gains using equations below

$$K_P = 0.6 * K_O \frac{\text{Nm}}{\text{rad}} \quad (5)$$

$$K_I = 2 * K_{P*} f_O \frac{\text{Nm}}{\text{rad}*\text{sec}} \quad (6)$$

$$K_D = \frac{K_P}{8*f_O} \frac{\text{Nm}}{\text{rad}*\text{sec}} \quad (7)$$

The proportional term affects the overall response of the system to a position error. The integral term is needed to force the steady state position error to zero for a constant position command and the derivative term is needed to provide a damping action, as the response becomes oscillatory. However, all three parameters are inter-related so that adjusting one parameter does not affect any of the previous parameter adjustments. The motors that were decided to be utilized for the project have PID controller built it.

4.3.1.4 Servo Drivers

There are various servo drivers that were found in the market. However, In order to implement drivers on the PCB, the team would have to build its own driver initially. Fortunately, the servo that was implemented in the project is digital and has built in PID controller therefore servo drivers were no longer needed. Drivers simply take signals from PID controller. Those signals are the coordinates of the designated target in PWM with cycle rate. Drivers simply power motors to move to a certain degree. They are simple circuits, like on-off switches. The group found this specific circuitry, shown in Figure 13, that can be implemented in the project with a few value adjustments of electronic components. It was designed by Andy Batts, a CS associate professor at Murray State University.

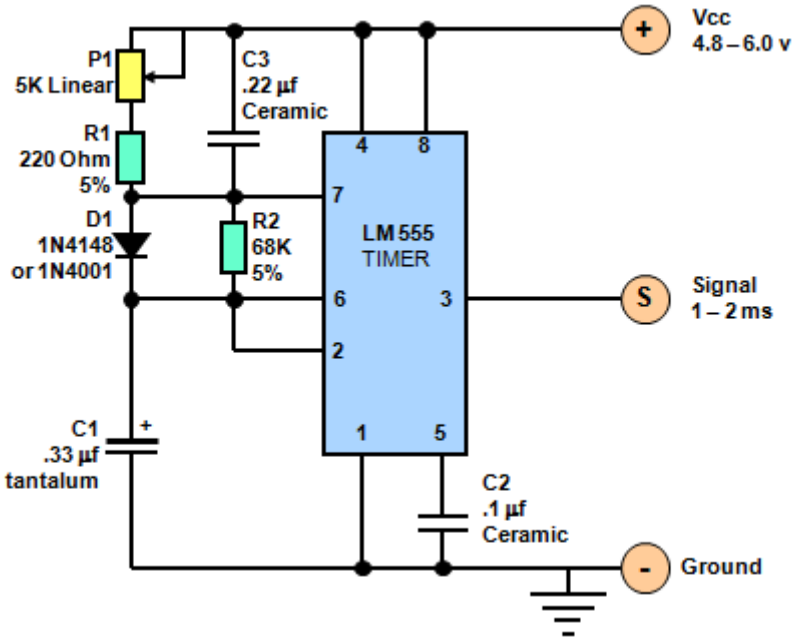


Figure 13: Servo Driver Schematic diagram

4.3.2 PCB Design

As mentioned earlier in the PCB Requirement section and Initial Design Architecture section, the group decided to include three subsystem components on the PCB: the voltage regulator, the microcontroller, and the wireless module. The wireless communication module is necessary to communicate to the user interface. The voltage regulator is needed since the PCB board is powered at 5v and the Atmel Atmega328 is operated at 3.3V. The board was designed using the free version of EAGLE. The size of the board is restricted to be 4 x 3.2 inches due to the limitations of the free version. Since the size of the board is not spacious, the space arrangement of the PCB becomes important. In order to have an efficient PCB layout, understanding wire connection between each unit is essential. The Atmel Atmega328 in particular has multiple connections. Therefore, decent knowledge of its architecture was required before assembling the PCB. A block diagram is given in Figure 14.

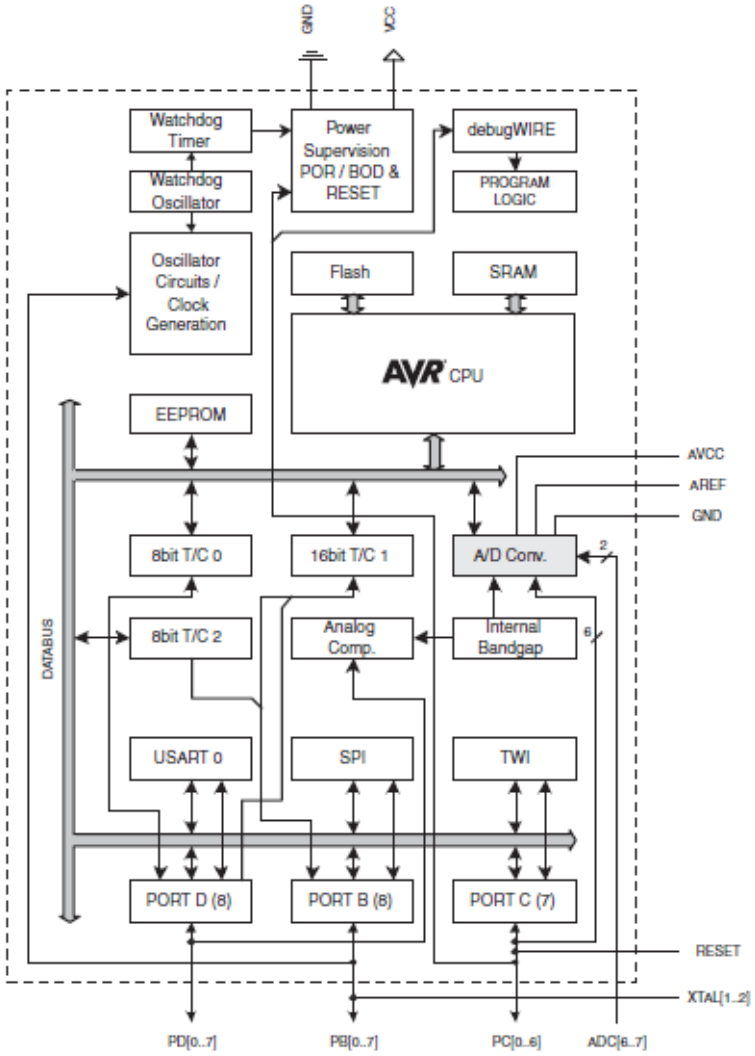


Figure 14: Block Diagram of Atmel Atmega328 Architecture

Another important factor that had to be considered while designing the PCB board was that different units are powered at different voltage levels. It is possible to burn out electrical components if close attention is not paid to the power level for each unit. For example, the Arduino Uno is powered between 9V to 12V but the Atmel Atmega328 requires 5V. To solve the problem, it is necessary to include two LM7805 regulators, which amplify an input of 7V to 20V, and output 1 A individually. One was used as the board power while the other one was used to power the wireless module. The motors that are being used are 168oz-in, which draw 500mA individually. However, instead of implementing a heavy weight paintball marker, it was decided to implement a light weight laser pointer. In that case, motor does not draw up to 500mA individually. Therefore, LM7805 should be sufficient to power the board and drive the motors. Its characteristics are listed in Table 10 below.

Table 10: Electrical Characteristics (LM7805) from Fairchild Semiconductor

Symbol	Parameter	Conditions		Min.	Typ.	Max.	Unit
V_O	Output Voltage	T_J = +25°C		4.8	5.0	5.2	V
		5mA ≤ I_O ≤ 1A, P_O ≤ 15W, V_I = 7V to 20V		4.75	5.0	5.25	
Regline	Line Regulation(1)	T_J = +25°C	V_O = 7V to 25V	–	4.0	100	mV
			V_I = 8V to 12V	–	1.6	50.0	
Regload	Load Regulation(1)	T_J = +25°C	I_O = 5mA to 1.5A	–	9.0	100	mV
			I_O = 250mA to 750mA	–	4.0	50.0	
I_Q	Quiescent Current	T_J = +25°C		–	5.0	8.0	mA
ΔI_Q	Quiescent Current Change	I_O = 5mA to 1A		–	0.03	0.5	mA
		V_I = 7V to 25V		–	0.3	1.3	

4.4 Firing Control

4.4.1 Tablet/Microcontroller Interface

The interface between the tablet and microcontroller was utilize the IEEE 802.15.4 wireless standard, since this standard calls for a 70m indoor/250m outdoor range which affords the project a margin of error beyond its initial 40m specification.

The tablet broadcasts control information using this protocol, and the microcontroller receives it via a XBee coordinator end device. The microcontroller in turn processes this information on-board and output control to the servos.

4.4.2 Microcontroller/Gun Interface

The Arduino Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, so to fire the laser pointer, which operates normally on two AA batteries, the microcontroller simply to, from any one of the A0 to A5 pins, be fed a binary value representing 3V. Since the 5V is divided by 1024 distinct values, a binary value is needed to represent that fraction. Thus, it is simply discernable by

$$1024 * \frac{3}{5} = 614.4 \tag{8}$$

which, rounding down to 614 and converting to binary, is a value of

$$1001100110$$

which can be fed to the board via the `analogreference{}` function.

4.5 Image Acquisition

4.5.1 Camera Hardware

After deliberating amongst camera hardware and using a wireless USB interface, the group decided that using an integrated wireless-N camera would be the most efficient, elegant, and discrete solution. The model chosen was the Linksys WVC80N (LINKSYS by Cisco) for its price ($109.95 on Amazon.com), resolution (640x480), and relatively high frame rate for consumer cameras (30 fps). While this did not quite match the initial requirements, it was still sufficient, as the velocity of the moving target was overestimated by a large margin.

4.5.2 Target Acquisition

One of the larger anticipated tasks of the project was using the tablet to find and track moving targets based on the images transmitted by the wireless camera. This process was divided into three fundamental operations. First, when an object in the camera's field of view was selected by the user, it had to be recognized as a potential target. Next, the object had to be represented in some way so that its basic boundaries were recognized and the processor knew where to aim the gun to hit the center of the target. Finally, since the object was not stationary, its movement had to be tracked so that the turret could find the distance between the target location and the current gun position, and aim accordingly.

4.5.2.1 Object Detection

4.5.2.1.1 Background Subtraction

One of the common methods of object detection, especially for use in video security applications, is a technique known as background subtraction. In this technique, the processor "learns" what the background image looks like. This is then used as a reference with which to compare the current image. The known background parts are subtracted away, which leaves only the object in the foreground.

There are numerous weaknesses in dealing with this method, which had to be overcome to properly identify the targeted objects. One consideration the group took into account was that the camera had to be stationary for this method to work; otherwise each new image would differ entirely from the original frame of reference, which would negate the purpose. Another concern was the separation of foreground and background. If, for example, a trash can was moved, it would be read as a foreground image in both its new location and in the hole it originally inhabited. To solve this problem, the program would need to classify three distinct layers: new foreground, old foreground, and background. Then, when an object moved initially, it would be placed in the new foreground. After a specific amount of time had passed with no further movement, the object would be

relegated to the old foreground, where its pixels would be gradually learned and joined to the learned background image. Another problem dealt with a change in lighting. If the lights in a darkened room were suddenly turned on, it would appear that the entire image had changed, and thus everything would be in the new foreground. To avoid this scenario, it could have been stipulated that if a large number of pixels changed at the same time, this would constitute a global change rather than a local one, which a new model could have been set up to handle.

At the most basic level, background subtraction can be demonstrated in the following procedure, which is also illustrated in Figure 15. First, the difference between two frames, which may or may not be consecutive, is taken and set as the new foreground image. The program can utilize the function cvAbsDiff to find the divergence between the original frame, called frametime1, and the current frame, frametime2, and detect the magnitude of differences in frameForeground. To more clearly delineate the layers, the function cvThreshold can then be employed to set any pixel with a value less than 15 to 0, and any pixel with a value greater than 15 to 255. This way any small fluctuations due to noise can be moderated.

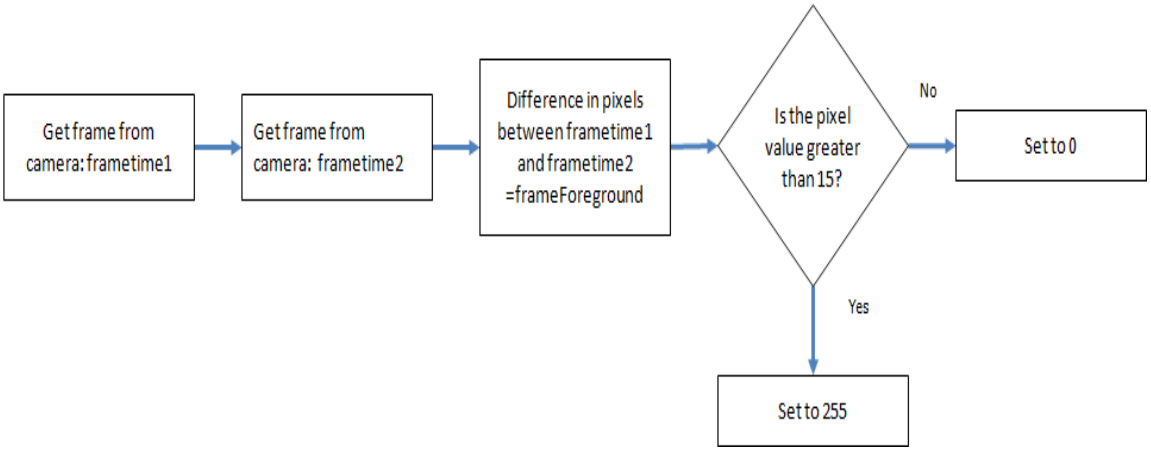


Figure 15: Basic Flowchart of Background Differencing

The simplicity of this program is useful for demonstrating the basic process of background subtraction, but in the real world it doesn't correct for errors such as slight background movement. One method that is a little more accurate is the averaging background method, which learns the average and average difference for each pixel in the background image. The average difference is similar to standard deviation but faster to calculate, which makes it the better option for this procedure. This is useful because the background image will probably still contain miniscule movements of background objects due to environmental factors. This movement can be taken into account by analyzing the image over a period of time and finding the average value for each pixel, as well as the

average difference which is due to background motion. When there is movement detected in the image, the program calculates whether or not the new pixels fall within the given range for average pixel difference. If they do, it is assumed to be part of the background layer. If, however, they fall outside of the computed range, the object is determined to be a new entity and classified to be a foreground object.

The following procedure was based off an example from the book “Learning OpenCV” by Gary Bradski and Adrian Kaehler. The first step was to initialize all the intermediate images that the program would use to hold the various values for each pixel, such as average, pixel difference, previous image, highest and lowest points in the range, etc. It also created a variable called Icount, which was declared as a float, to be the counter for the number of images analyzed, so the average can be calculated at a later point. The function IplImage was used for these purposes

The program had to make all intermediate images the same dimensions, so that corresponding pixels could be accurately compared. The function cvCreateImage generated the images, and used the function cvGetSize(I) to set all the intermediate images to equal sizes, based on the image I. This image was just an arbitrary sample frame from the camera that was used for allocation purposes. Afterwards, the three-channel images were initialized to zero with the function cvZero.

Next the program learned the average value and the average differences in values between frames. In order to obtain the correct values, the image had to be changed from an 8-bit three channel image into a floating point three channel image. For an 8-bit grayscale image, the value is based on the brightness of the pixel, on a scale of 0 (black) to 255 (white). A 24-bit RGB colored image has three channels with 8 bits per channel, and essentially follows the same scale. Each of the three colors can have a value ranging from 0-255, with 0 being the least saturated and 255 being the most. One problem with manipulating these values, such as adding or subtracting a specified quantity to each pixel, is that they ‘hit a wall’ once they reach their maximum or minimum value. For example, adding 50 to two pixels, one valued at 210 and the other at 245, will result in both having values of 255. Subsequently subtracting 50 will give values of 205 for each, which are not only different from the original values, but also makes them indistinguishable from each other, which leads to blurring of details in the picture. To avoid this, the program below converts the 24-bit image into a floating point image, which changes the scale from 0 to 1 but allows for decimal values and keeps track of numbers greater than 1 (“whiter than white” values), and those less than 0 (“blacker than black”). This way the original values are not lost during pixel manipulations.

The newly generated floating point image was stored in Iscratch, then passes to an if statement, where the data for each pixel was accumulated in IavgF. Next

the absolute difference was found between the current image and the previous one, with the results accumulated and stored in IdiffF. Icount was incremented to keep track of the number of images, and the current image was moved to IprevF, to be compared to the next image in the cycle. The process was shown in Figure 16.

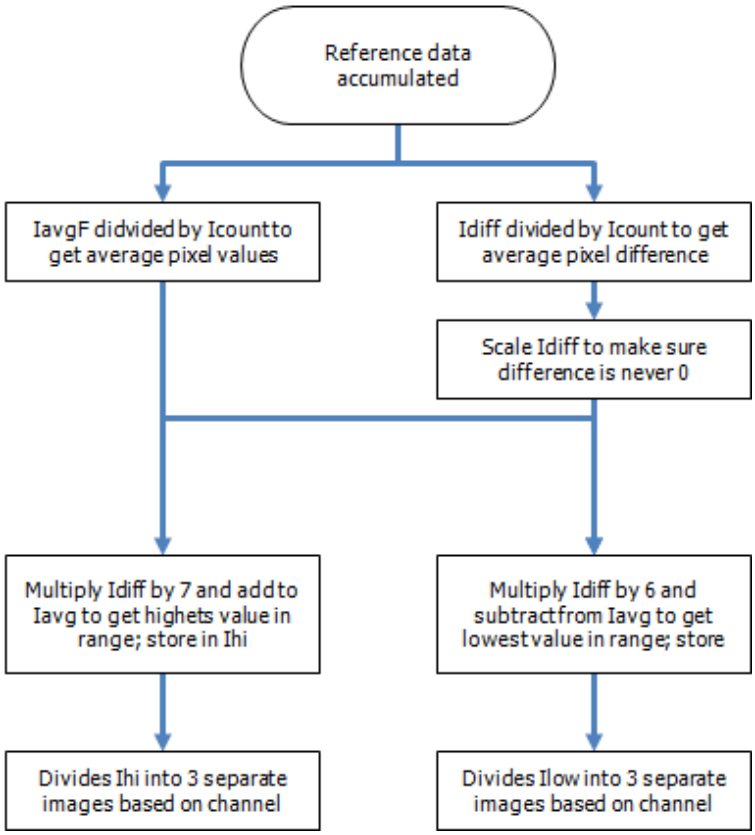


Figure 16: Accumulation of background data

The next step was to find the high and low thresholds (Figure 17). When enough data was accumulated, the program divided the values stored in IavgF and IdiffF by the total number of images to find the averages. The command cvAddS ensured that the minimum value stored in IdiffF was at least one, so that it could be properly scaled later. Finally, the thresholds were set for the high and low ends of the average pixel difference. For the high threshold, the values stored in IdiffF were multiplied by 7. This was then added to the matrix of averages stored in IavgF, and the results were saved to IhiF. Finally, cvSplit divided IhiF into 3 separate images based on channel, so that the range for each could be computed individually. The low threshold was set following the same procedure, except the average differences were multiplied by 6 instead of 7, and subsequently subtracted from the average values.

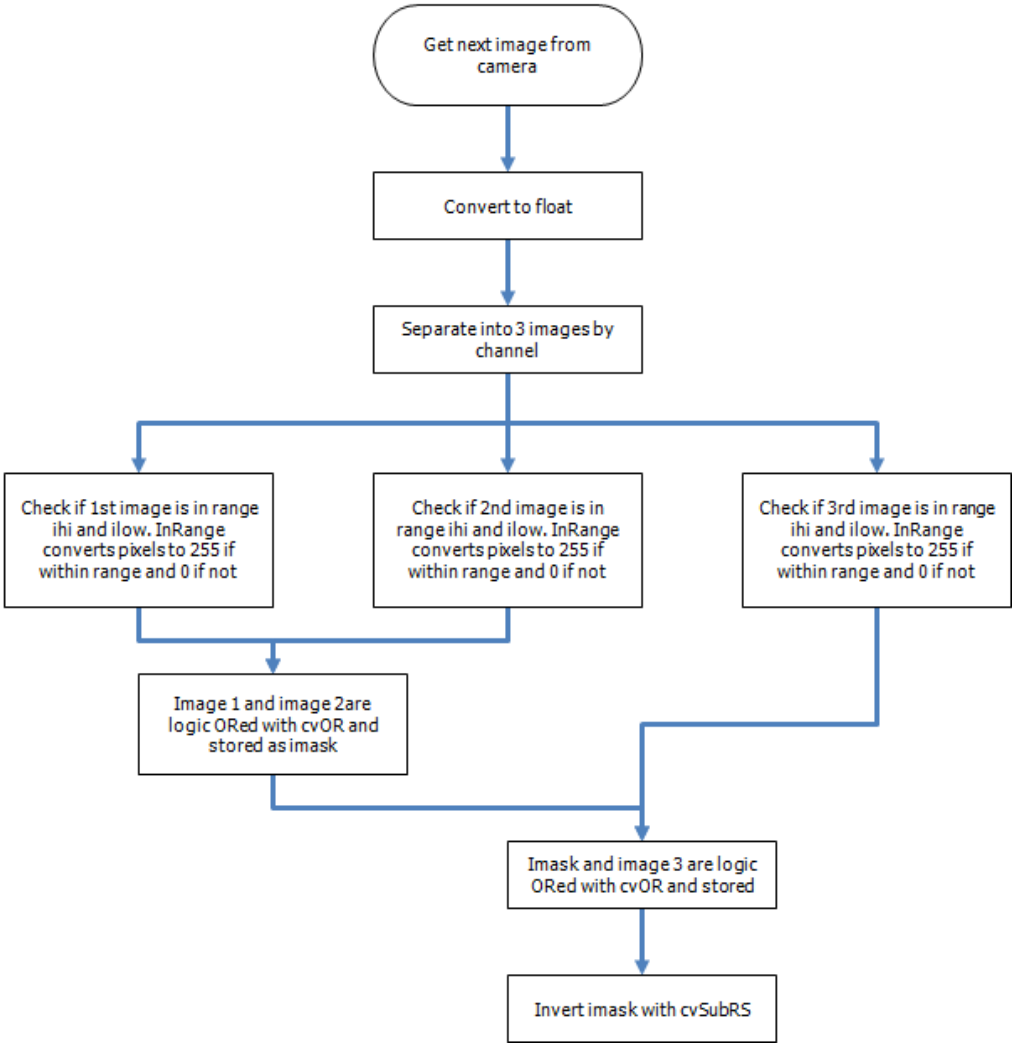


Figure 17: Finding High and Low Thresholds

Finally, the preliminary work has been completed, and the program was now at the point where new images could be read in to ascertain whether they contained foreground objects (Figure 18). Again, the current image I was converted to floating point, then separated into three grayscale images based on channel. The function cvInRange checked to see whether the pixel values fell within the allotted range, converting them to 255 if they did and 0 if they did not and storing them in the 8-bit grayscale Imask. After each channel's image underwent this process, it was logically ORed with the previous masks and this combined image became the new Imask. That way, if even one channel's pixels fell outside the range, meaning there was a large difference in any single color, the program would indicate that an object was present. The last step was to invert the values within Imask, since any objects found were actually out of range instead of in range, which was the opposite calculated by cvInRange.

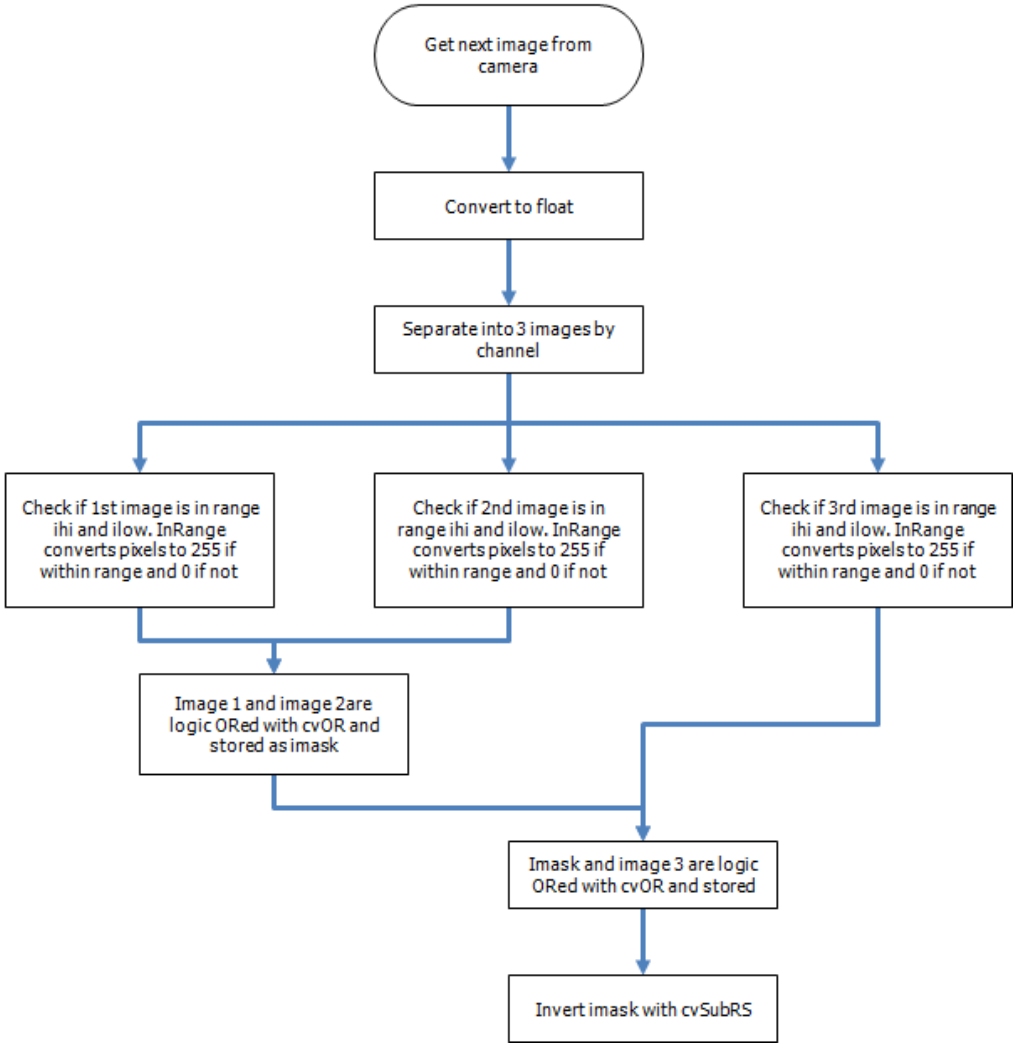


Figure 18: Flowchart for comparing background difference

The last thing that needed to be done was to release the images from memory, so that they did not take up unnecessary space. This was carried out with the command cvReleaseImage() for all the intermediate images used. The final result was then stored in Imask for use in the next section of code, which represented the object geometrically.

4.5.2.1.2 Color Recognition

After implementation of the background subtraction method, it was discovered that due to its complexity, a substantial time delay occurred between receiving a frame and displaying the newly processed frame on the user interface. Consequently, a faster method was needed; the group decided upon a color recognition technique, which works as follows. First, each incoming image is converted from RGB to HSV. The user selects an object on the screen that they

would like to track, and the program then targets that selected pixel and stores its value, which represents the color. This value can then be fed into the program, along with a specified threshold, which serves to filter out any pixels that fall outside the given range. A new image is created that contains a thresholded binary image, where all the pixels of the selected color are converted to the maximum value (white) and the rest of the pixels, which fall outside the threshold are given the minimum value (black). This image is displayed to the user for debugging purposes.

4.5.2.2 Object Representation

Now that the program has detected the existence of an object, an outline must be formed around the target so that the centroid can be calculated. In order to accomplish this, the newly created blob that is the moving object must be enclosed with a geometric representation. For efficiency as well as ease of the centroid calculation, which will be used later for the tracking portion, a simple rectangle was chosen as the shape of choice. First, a smoothing filter is applied to the incoming frame using a median filter, which analyzes the neighboring values of each pixel and sets it to the calculated median. Then an included library, cvBlobs, detects consecutive pixels of the chosen color as blobs, and represented them to the user by outlining them in colored rectangles. Additionally, a "filter by area" function is applied so that only blobs of a minimum size would be read by the program, thus limiting noise errors. Figure 19 below shows the binary image along with the live stream window with the blob detection representation.

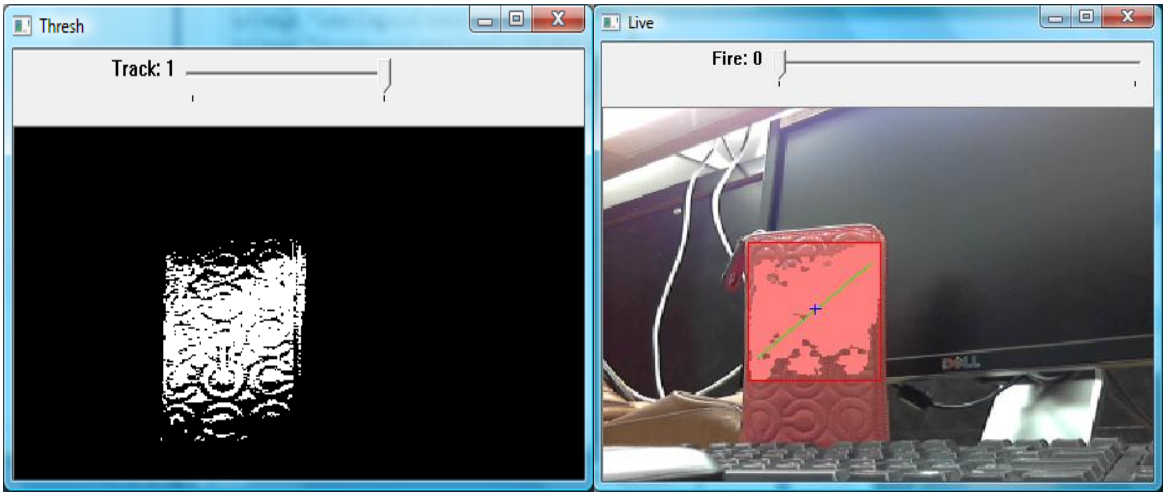


Figure 19: Target Acquisition User Interface

4.5.2.3 Object Tracking

The next step is to find the center of the target. This is the point where the servos will aim the firing device, whether it a gun or a laser pointer, to fire, and it also acts as a means by which the target motion can be tracked. The position of the

thresholded binary image is found by using the moments calculated by the function cvMoments. These are then divided by the screen dimensions to give representative location values within the viewing window. By comparing the difference of location between the current frame and the previous frame, the object can be accurately tracked. This will inform the microcontroller how much it needs to move the servos to maintain its aim with the target. If the gun is in its reset position facing straight ahead, the program must compare the target's position with this default location in order to orient the gun correctly. The coordinates are displayed to the screen for the user.

Once this information is calculated, it needs to be sent wirelessly to the microcontroller for interpretation into commands to move the servos, so that the laser pointer is pointing at the desired target. In order to accomplish this, an included serial library is used to send the x-coordinate and y-coordinate to the serial port, where an XBee explorer transmits the information to another XBee connected to the microcontroller. The microcontroller then converts the coordinates to PWM signals and outputs the signals to the servo motors for tracking. The figure below demonstrates the process of object tracking through centroid comparisons.

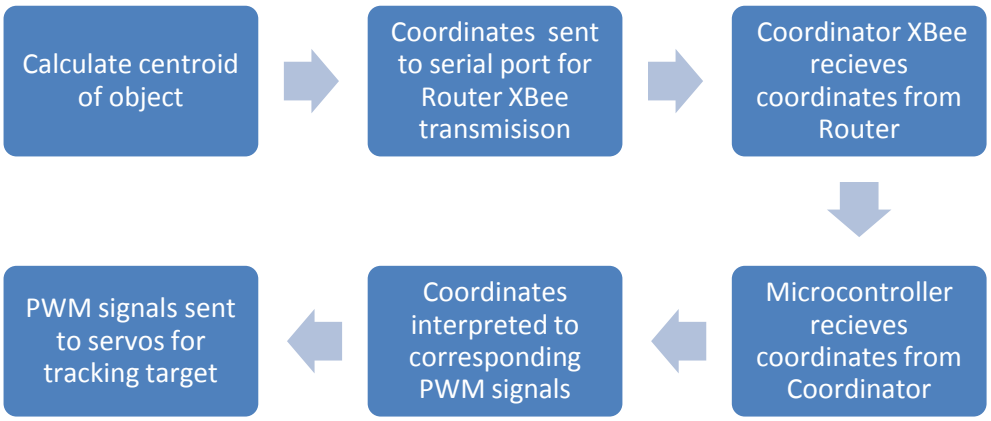


Figure 20: Calculation to aim turret

When these steps are completed, the gun is now properly aimed towards a target. Then the cycle starts over again, with the camera sending a new frame to the tablet for processing. If the target is found to be still within range, the turret continues to track the selected target. If the target is no longer within range, the turret stops and waits for new user input.

4.6 Wireless communication

4.6.1 Camera-UI

As mentioned before, the group expected massive analog data that would be transmitted from the camera to the user interface. After deliberating, it was decided to use a wireless camera. A wireless camera is the easiest, and the most sufficient wireless technology that the group has found. Its “plug and play” scenario would have saved some hassle programming it. However, setting up the AD-hoc network was not as easy as the group expected. But the group still solved the problem. Time delay is critical in this project. The group had set the resolution to be 320X240 and frame rate to be 10fps to minimize to the delay.

4.6.2 UI-Arduino

The protocol between the user interface and Arduino that the group decided to use is XBee Pro RF modules. This module is certified IEEE 802.15.4 module. XBee modules have integrated PCB antennas. It can connect to the Arduino board simply using SPI interface and is an ideal solution for a low-power, low data rate pin for pin network. Figure21 below shows internal data flow of the module. According the XBee data sheet, those modules interface to a host device through a logic-level asynchronous serial port. Through its serial port, the module can either communicate with logic and voltage compatible UART or through a level translator to a USB interface port. Figure 22 below illustrates the serial bit pattern of data passing through the module. Serial communication depend on the microcontroller’s UART and the RF module’s UART to configure with compatible settings such as baud rate, parity, start bits, stop bits, and data bits.

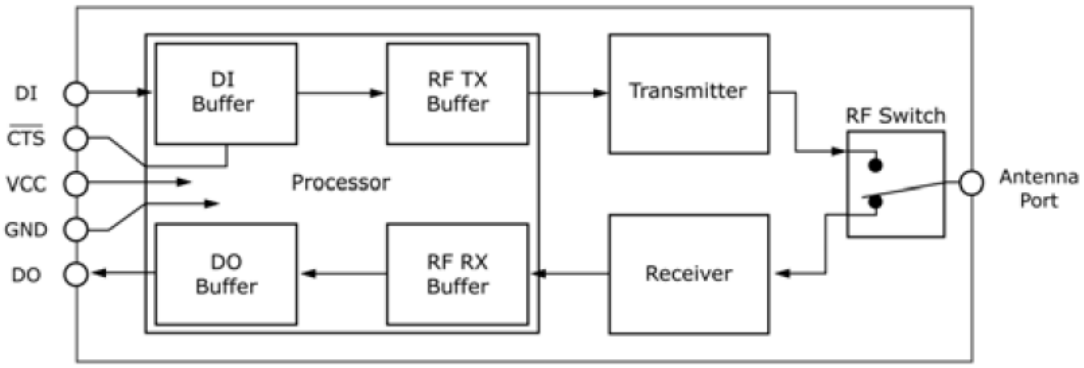


Figure 21 Internal Data Flow diagram

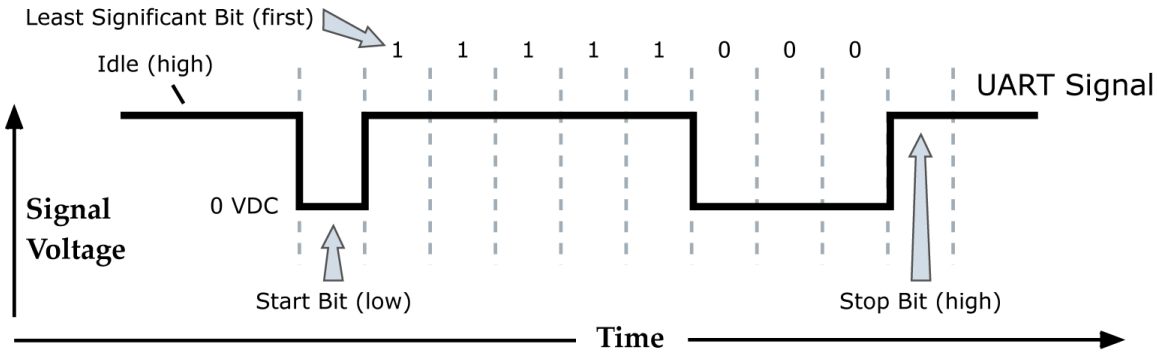


Figure 22 UART data packet as transmitted through the RF module

To pair the two modules in the project, the group used X-CTU to reconfigure them. It was important to know the basic computer communication topology to have hose two modules paired successfully. Since XBee modules operates at frequency 2.4GHz. It is important to assign unique PAN ID address for the network and link the destination addresses. One end of the wireless unit was set to be XBee Pro Router end device, which is connected to the tablet using XBee USB explorer, sending out data. The other end of the unit was set to be XBee 802.15.4 coordinator end device, which is implemented on the PCB and communicating to the microcontroller through SPI interface.

XBee modules supports sleep mode. Sleep modes enable the RF module to enter states of low power consumption. The SM command is central to setting sleep mode configuration. By default, sleep modes are disable (SM=0) and the module remains in Idle/ Receive mode, which indicates the module is constantly ready to respond to serial or RF activity. Table 11 below shows the sleep mode configurations.

Table 11 Sleep Mode Configurations from Digi International, Inc

Sleep Mode Setting	Transition into Sleep Mode	Transition out of Sleep Mode (wake)	Characteristics	Related Commands	Power Consumption
Pin Hibernate (SM = 1)	Assert (high) Sleep_RQ (pin 9)	De-assert (low) Sleep_RQ	Pin/Host-controlled / NonBeacon systems only / Lowest Power	(SM)	< 10 µA (@3.0 VCC)
Pin Doze (SM = 2)	Assert (high) Sleep_RQ (pin 9)	De-assert (low) Sleep_RQ	Pin/Host-controlled / NonBeacon systems only / Fastest wake-up	(SM)	< 50 µA
Cyclic Sleep (SM = 4)	Automatic transition to Sleep Mode as defined by the SM (Sleep Mode) and ST (Time before Sleep) parameters.	Transition occurs after the cyclic sleep time interval elapses. The time interval is defined by the SP (Cyclic Sleep Period) parameter.	RF module wakes in pre-determined time intervals to detect if RF data is present / When SM = 5	(SM), SP, ST	< 50 µA when sleeping
Cyclic Sleep (SM = 5)	Automatic transition to Sleep Mode as defined by the SM (Sleep Mode) and ST (Time before Sleep) parameters or or on a falling edge transition of the SLEEP_RQ pin.	Transition occurs after the cyclic sleep time interval elapses. The time interval is defined by the SP (Cyclic Sleep Period) parameter.	RF module wakes in pre-determined time intervals to detect if RF data is present. Module also wakes on a falling edge of SLEEP_RQ	(SM), SP, ST	< 50 µA when sleeping

In order to design the PCB board properly for the project, the team had to know the digital electrical characteristics of this module. Width of trace is determined by the current that flows in the circuitry. To sum the total current, electrical characteristics of each electronic unit that were going to be implemented in the PCB board had to be acknowledged. See Figure 22 and Table 11 below for the detailed module specification.

Table 12: DC Characteristics from Digi International, Inc

Symbol	Characteristic	Condition	Min	Typical		Max	Unit
V_{IL}	Input Low Voltage	All Digital Inputs	-	-		0.35 * VCC	V
V_{IH}	Input High Voltage	All Digital Inputs	0.7 * VCC	-		-	V
V_{OL}	Output Low Voltage	I_{OL} = 2 mA, VCC >= 2.7 V	-	-		0.5	V
V_{OH}	Output High Voltage	I_{OH} = -2 mA, VCC >= 2.7 V	VCC - 0.5	-		-	V
II_{IN}	Input Leakage Current	V_{IN} = VCC or GND, all inputs, per pin	-	0.025		1	µA
II_{OZ}	High Impedance Leakage Current	V_{IN} = VCC or GND, all I/O High-Z, per pin	-	0.025		1	µA
TX	Transmit Current	VCC = 3.3 V	-	45 (XBee)	215, 140 (PRO, Int)	-	mA
RX	Receive Current	VCC = 3.3 V	-	50 (XBee)	55 (PRO)	-	mA
PWR-DWN	Power-down Current	SM parameter = 1	-	< 10		-	µA

4.7 Range Calculation

In the scenario that a paintball gun was implemented as the firing device, it was necessary to include a rangefinder, so that the trajectory of the paintball pellet would terminate at the center of the target. The range finding system that had been chosen, as described above in Research section 3.2.1, was the laser pointer and image sensor system. The laser pointer that would be used is the Instapark® Green Laser Pointer, which comes with multiple tips in different geometric patterns. The crosshair tip aids in pinpointing the exact spot on the target, reducing the uncertainty caused by a larger area of light. The image sensor chosen for the project was the ELIS-1024A from Panasonic, a 1024x1 linear image sensor that adequately fits the specifications. The high resolution should provide relatively accurate results. Since the sensor measures only one pixel in width, it would have had to be aligned perfectly with the laser pointer so that it saw the light, such that the laser is parallel to the optical axis of the sensor. This precision was important for the effectiveness of the range finding, and both the sensor and the pointer would have been firmly fixed in place once it had been achieved so that it was continuously maintained.

The turret would have used the laser in the following manner. Once the target was detected through the image processing done by the tablet, the laser would be pointed at the specified object. Then the image from the linear sensor would be sent through the processor and wirelessly transmitter to the Windows tablet. The tablet would use a programmed algorithm to look for the brightest pixels,

which is where the laser was reflecting off the object. The location of this spot of light could be calculated by finding its distance from the edges of the frame.

Next the system would go through the calculations to determine the depth of the target. The distance could be calculated from known geometrical formulas, as described previously in the Rangefinder section of Research. See Equation 2 and Figure 5 for reference. The unknowns, then, would be h, the vertical distance between the image sensor and the laser pointer, and θ, the angle. Equation 10 given below shows how to obtain θ from pfc, which is the number of pixels from the center of the focal plane, rpc, which is the radians per pixel pitch, and ro, which is the radian offset. Of these variables, only pfc would be known, so it would be necessary to perform some calibration of the rangefinder derive the other two.

$$\theta = pfc * rpc + ro \tag{10}$$

The system would collect some sample data of objects a known distance away and use this distance, as well as h, a constant, in Equation 2 to find the angle for each case. Since the pfc for each is also known, the linear relationship from Equation 10 could be used to approximate the angle. Once multiple instances were obtained, it would have been a simple task to solve for approximate values of gain rpc and offset ro. With the newly obtained values, angle θ could then be calculated by measuring the number of pixels away from the center of the focal plane that the image lay. Since this made both h and θ known values, the distance could then be computed. The final step was to send the depth information back to the microcontroller, where it could be converted into commands for moving the servos.

4.8 Power Supply

The group decided that the power supply should be a simple consumer-grade power strip for the prototype, and into that would be connected the power source for the Arduino, as discussed in section 2.3.8. However, throughout testing, the group noticed that the servos draw more current than expected. In order to have the best performance of the servos, an additional AC to DC adapter is used for one of the servo. And the other one is still powered on PCB.

The User Interface tablet operates via its battery when the system is in use, and charge via an AC outlet when the system is idle.

4.9 Hardware Housing

The housing for the project was constrained by requirements of durability, portability, accessibility, and visibility. In terms of durability, it needed to be able to withstand the torque applied by the servos to the armature, as well as frequent transport and disassembly. Since the project would be carried from workspace to

workspace, portability would be a high priority. Since very few engineering projects work on their first attempt, the device needed to be easily accessible to alteration. Also, the group was advised early on that during the final evaluation of the project, the parties performing the evaluation would be very interested in seeing the inner workings of the device, so it should be visible.

To address the durability requirement, a material with a strong tensile strength but low brittleness was required. In this aspect, a multitude of materials were considered. Wood would be supremely easy to work with, and of middle-range weight, however it could wear down quickly after repeated teardowns and rebuilds, as well as being prone to breakage if dropped or over-stressed. Aluminum is relatively light, but working with it would require an entirely new set of tools, and the sharpness of the edges is always a concern; even sharp right-angles of such a metal could lacerate a careless handler, and the group was highly concerned with the safety of the work. Plastics such as Poly (methyl methacrylate) – more popularly known as Plexiglas – offered a marriage of these features, even at a relatively heavy weight, but has one constraint: for long-term use, it cannot be worked with by amateurs. From Wikipedia:

> “PMMA vaporizes to gaseous compounds (including its monomers) upon laser cutting, so a very clean cut is made, and cutting is performed very easily. However, the pulsed laser cutting introduces a high internal stresses along the cut edge, which when exposed to solvents produces undesirable "stress-crazing" at the cut edge and several millimeters deep. Even ammonium-based glass-cleaner and almost everything short of soap-and-water produces similar undesirable crazing, sometimes over the entire surface of the cut parts, at great distances from the stressed edge. Annealing the PMMA sheet/parts is therefore an obligatory post-processing step when intending to chemically bond laser cut parts together. This involves heating the parts in an air circulating oven from room temperature up to 90°C (at a rate of no more than 18 degrees per hour) down to room temperature (at a rate of no more than 12 degrees per hour).” (Wikipedia, 2011)

Clearly this work is not within the scope of an EECS senior design project, so the group opted to simply use sheets of Lexan, available cheaply at Home Depot. The case has side-panels attached to the base. The top corners of each of the side-panels were drilled to allow for a cotter-pin to slide through. There is also a lid attached to the housing which has a cube on each corner, with a hole drilled through each; cotter-pin passes through one side panel, then the cube, then the other side panel, allowing for a compromise of simple deployment of the system and sturdy construction. See Figure 24, below.

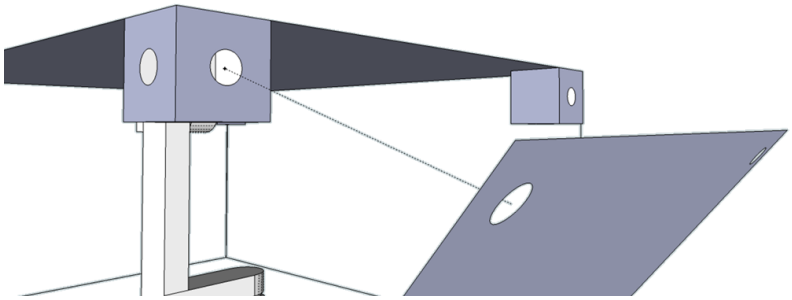


Figure 23: Detail of Closure Mechanism

5 DESIGN SUMMARY OF HARDWARE AND SOFTWARE

5.1 Turret and Case

The turret armature was purchased from paintballsentry.com, and the laser pointer was attached to it via hose clamps. The servos were inserted into their pre-made receptacles into the armature (Figure 26), and electrically connected via the harnesses that come with the armature.

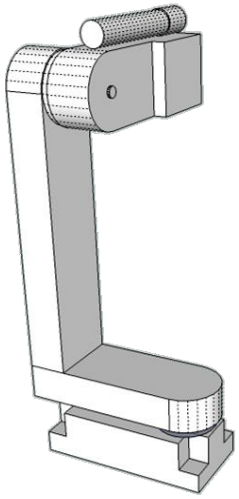


Figure 25: Servo Insertion into Armature

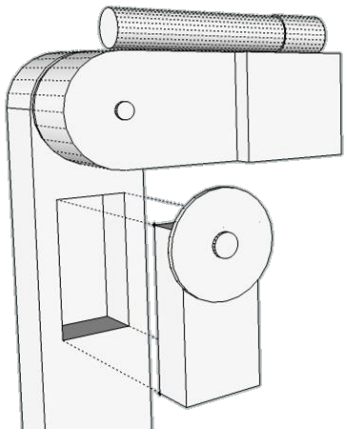


Figure 24: Armature and Laser Pointer Servo

The armature was secured to the base by four simple bolts, and the side-flaps attached to that assembly (Figure 27).

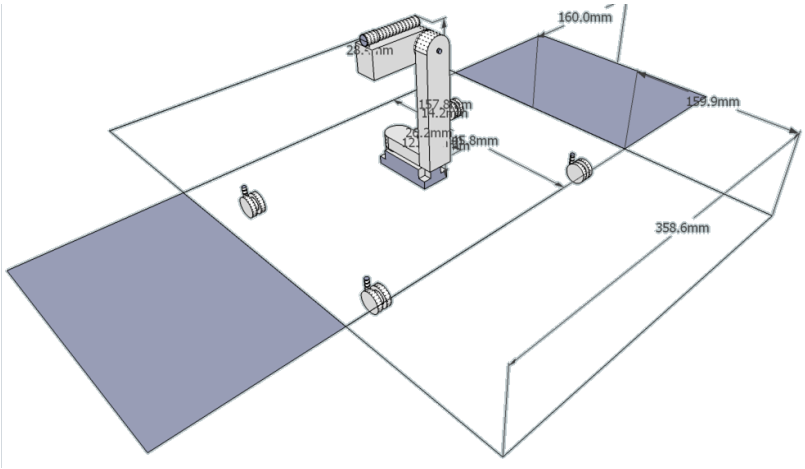


Figure 26: Turret and Housing

The lid of the enclosure has four blocks attached to the bottom which allows it to be secured to the side-flaps (Figures 28 & 29).

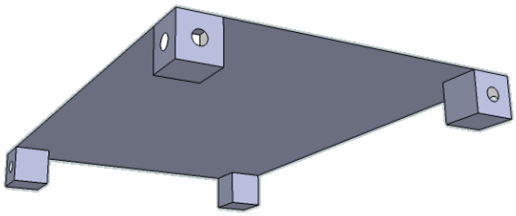


Figure 27: Lid of hardware housing

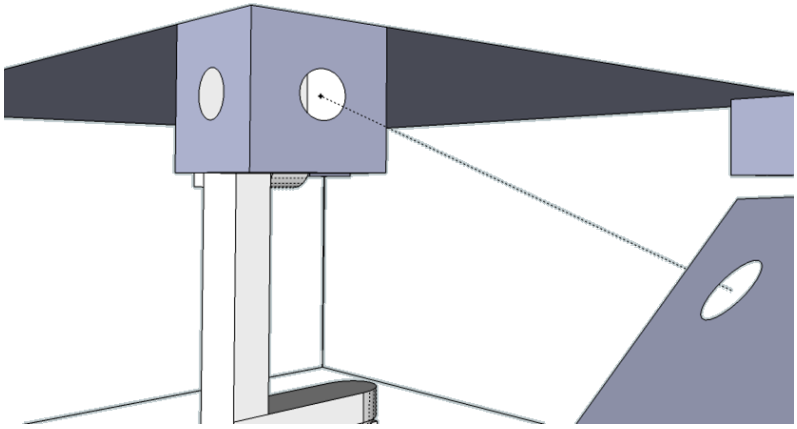


Figure 28: Closure of hardware housing

Clevis pins were used for this connection.

5.2 Image Processing

The camera sends the captured frames wirelessly to the tablet, both for implementation into the user interface as well as for image processing. The user interface is an application on the windows tablet that displays the incoming frames from the camera, with selected targets highlighted in a unique color.. A user selected moving target may be tracked, or a single point on the target field can be selected as the target. For the group's prototype model, a laser pointer will be employed in place of an actual paintball gun. The laser will be modified to demonstrate firing by turning on for approximately 0.5 seconds.

Object detection was implemented with the use of a color recognition program. The user selects an object on the screen that they would like to track, and the program will target that pixel and store its value, which represents the color. This value will then be fed into the program, along with a specified threshold, which

serves to filter out any pixels that fall outside the given range. A new image is created that contains a thresholded binary image, where all the pixels of the selected color are converted to the maximum value (white) and the rest of the pixels, which fall outside the threshold are given the minimum value(black). This image is displayed to the user for debugging purposes. A smoothing filter is applied to the incoming frame using a median filter, which analyzes the neighboring values of each pixel and sets it to the calculated median. Next each image is converted from RGB to HSV. Another included library, cvBlobs, detected consecutive pixels of the chosen color as blobs, and represented them to the user by outlining them in colored rectangles. Additionally, a “filter by area” function was applied so that only blobs of a minimum size would be read by the program, thus limiting errors.

The position of the tracked target is found by using moments. By comparing the difference of location between the current frame and the previous frame, the object can be accurately tracked. This will inform the microcontroller how much it needs to move the servos to maintain its aim with the target. If the gun is in its default position facing straight ahead, the program must compare the target’s position with this location in order to orient the gun correctly. The coordinates are displayed to the screen for the user. Finally, an included serial library is used to send the x and y coordinates to the serial port, where an XBee explorer transmits the information to another XBee connected to the microcontroller, which in turn converts the coordinates to PWM signals and outputs it to the servo motors for tracking, along with a firing command to the laser pointer. Figure 29 summarizes the process.

Another feature of the system is the ability to select stationary targets. When a user has selected a random point in the viewing window, the system must recognize the specified point as the new target. As with the moving targets, the location must be calculated, however since the object does not move, the ‘tracking’ is unnecessary, so this calculation only needs to be done once. The coordinates are found and sent to the XBee transmitter, along with a firing command, which the microcontroller receives and interprets by moving the turret to the specified point and flashing the laser pointer.
.

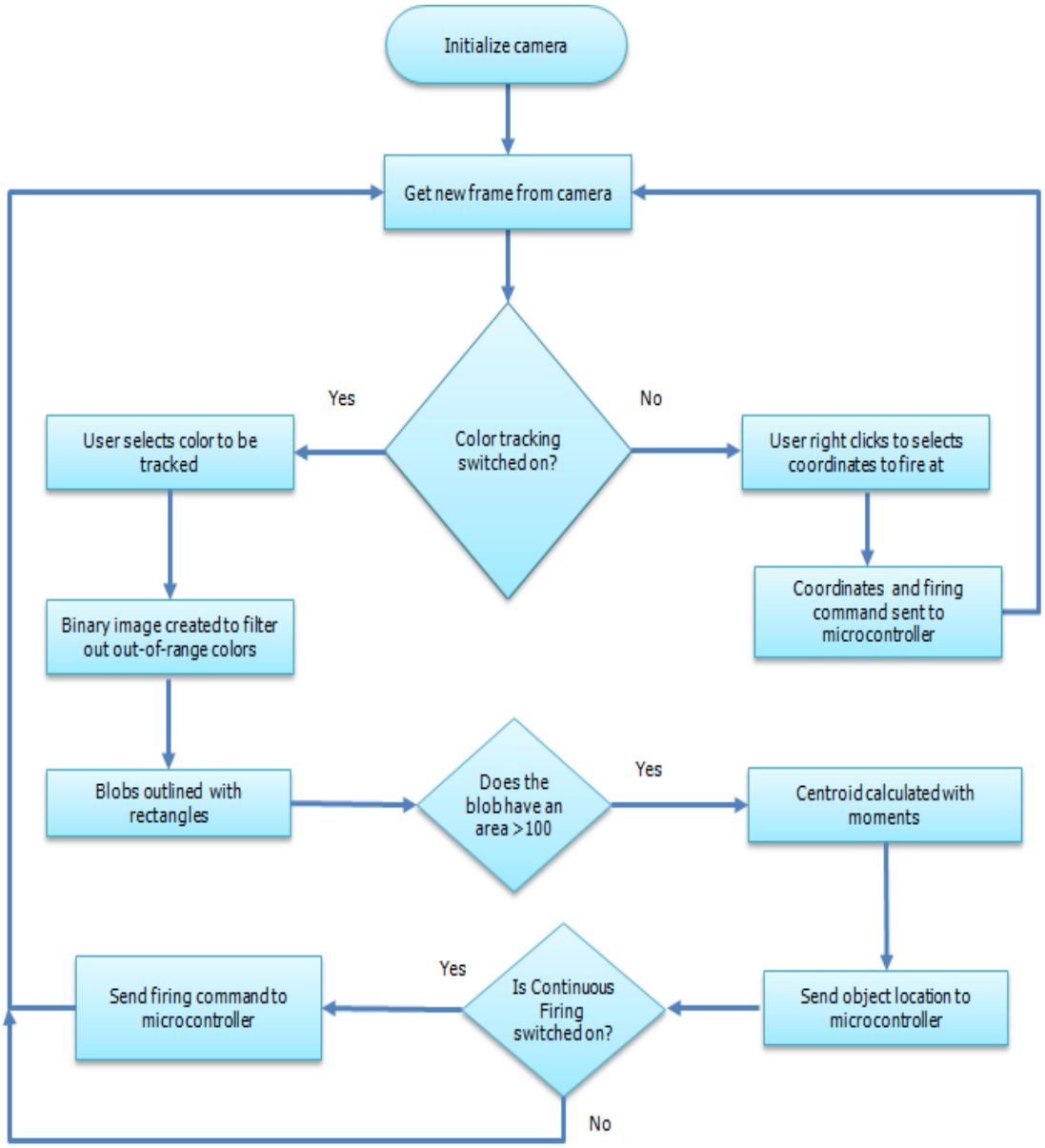


Figure 29: Flowchart for image processing

5.3 Electrical Hardware

The whole system is controlled by two major subsystems, the user interface and the microcontroller. Basically, the user monitors the field based on what is displayed on the tablet. When a designated target is chosen, servos move to the position and wait for a firing command. In order to control the turret system without being close to the battle field, a wireless module is implemented on the PCB connected to the microcontroller. The microcontroller controls the servo system using a simple serial port; only three wires are needed, power wire, ground wire, and command wire. Two voltage regulators are used on the PCB. One is used to power the wireless module. The other one is used to power the

Atmel ATmega328. In total, there are several electrical components are implemented on the PCB: the Atmel Atmega328, a wireless module, , two voltage regulators, resonator, and conductors.

It was decided that XBee Pro wireless module is the wireless protocol. This wireless module is powered at 5, which is powered at the same voltage level as the Atmel ATmega328, the core of Arduino Uno. The task of connecting the wireless module and the microcontroller is simple. I/O ports on the microcontroller are connected the power state ports and clock signal port on the wireless module. Signal clocks from both ends are connected. Setup for the wireless communication is relatively simple compared to the other parts of the circuitry on the PCB. After the wireless protocol is set up, the signals from the user interface can be fed into the microcontroller and then to the PID controller, built-in in servos, to eliminate errors and avoid overshoot. The PID controller is a simple chip that takes in an input from Atmel ATmega328 and feeds the compensated signal to drive the servos. The project electrical schematic is shown in Figure 31.

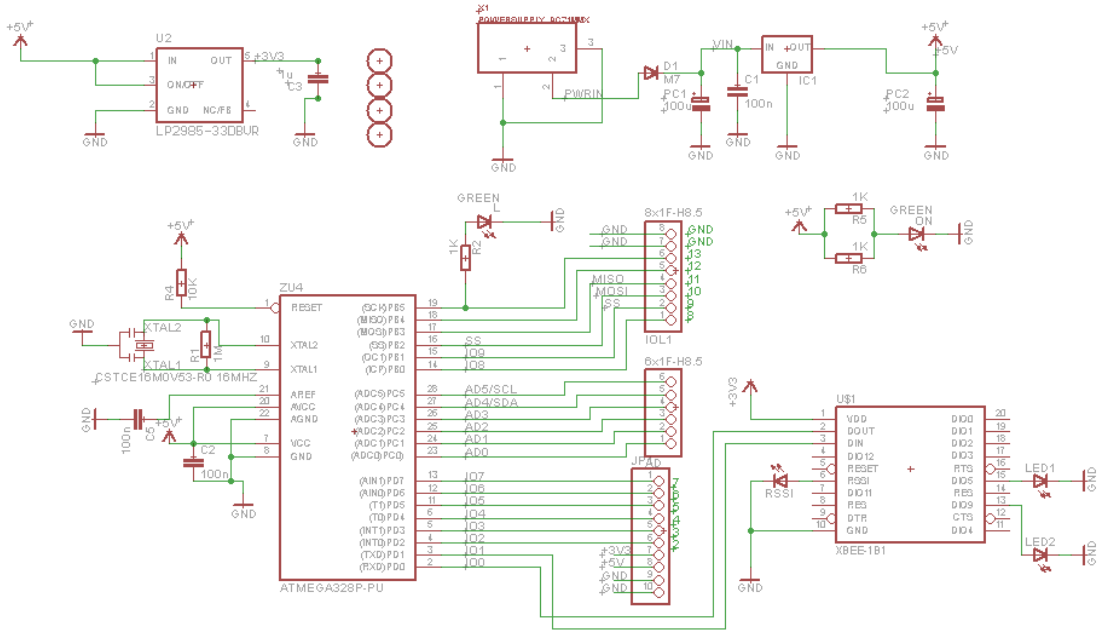


Figure 30: Project Electrical Schematic

6 PROJECT PROTOTYPE CONSTRUCTION

6.1 Hardware Fabrication

6.1.1 Housing Assembly

The only part of the system that had to be built is the housing. First, the base of the device is drilled to accept the base of the armature, hinges, the mounting screws and standoffs of the PCBs and power strip, and the casters. The first device to be attached to the base was the armature, which already has all of the components attached (see 6.1.2, Turret Assembly, below), followed by the PCB (driver board and Arduino) and power strip. At this point, no further construction was done until the device was fully tested and functional. Once full functionality was achieved, the next addition was the hinges, to which the side-panels were pre-attached. The top cover was attached via cotter pin following that.

6.1.2 Turret Assembly

As the turret comes largely pre-made, the procedure to assemble it was relatively simple; it was a matter of attaching gears to servos, inserting these servos into the armature, and then screwing in the servos. The laser pointer was then attached to the top portion of the armature using pipe clamps, and carefully secured in such a way that it properly aligns with the axis of the armature. The electrical portion of the assembly took place after the hardware fabrication in section 6.1; the harnesses included with the servos were attached to the driver board, and the power portion laser pointer were attached to the Arduino directly, such that it was provided the 3V it needs to fire at the appropriate time.

6.2 PCB Assembly

Once the PCB board was designed using the Eagle software, the next step was to send the design to a PCB manufacturing company to have the board physically constructed. There are numerous options available with regards to PCB fabrication. Three of the main companies that were looked into were 4pcb.com, pcbnet.com, and pcb123.com. The first company, 4pcb.com, offers a 2-layer board up to 60 square inches for $33, with minimum order quantity of 2 for students. The second choice, pcbnet.com, offered an introductory special with 2-layer boards at $25 each up to 60 square inches, with no minimum order required. The final company, expresspcb.com, offers a different pricing scheme, with a 2-layer board priced using the formula:

$$\$55 + (\$0.65 * NumberOfBoards * BoardAreaInSquareInches) + (\$1.00 * NumberOfBoards) + Shipping.$$

It was determined that a 2-layer board would sufficiently meet the requirements of the system, and that two copies should be purchased as a precautionary

measure. The final board house that the group chose was 4pcb, the reasoning being that, 4pcb offers free gerber files check, which lists out the problems that could possibly cause malfunction of the PCB. PCB schematic is shown in figure 32.

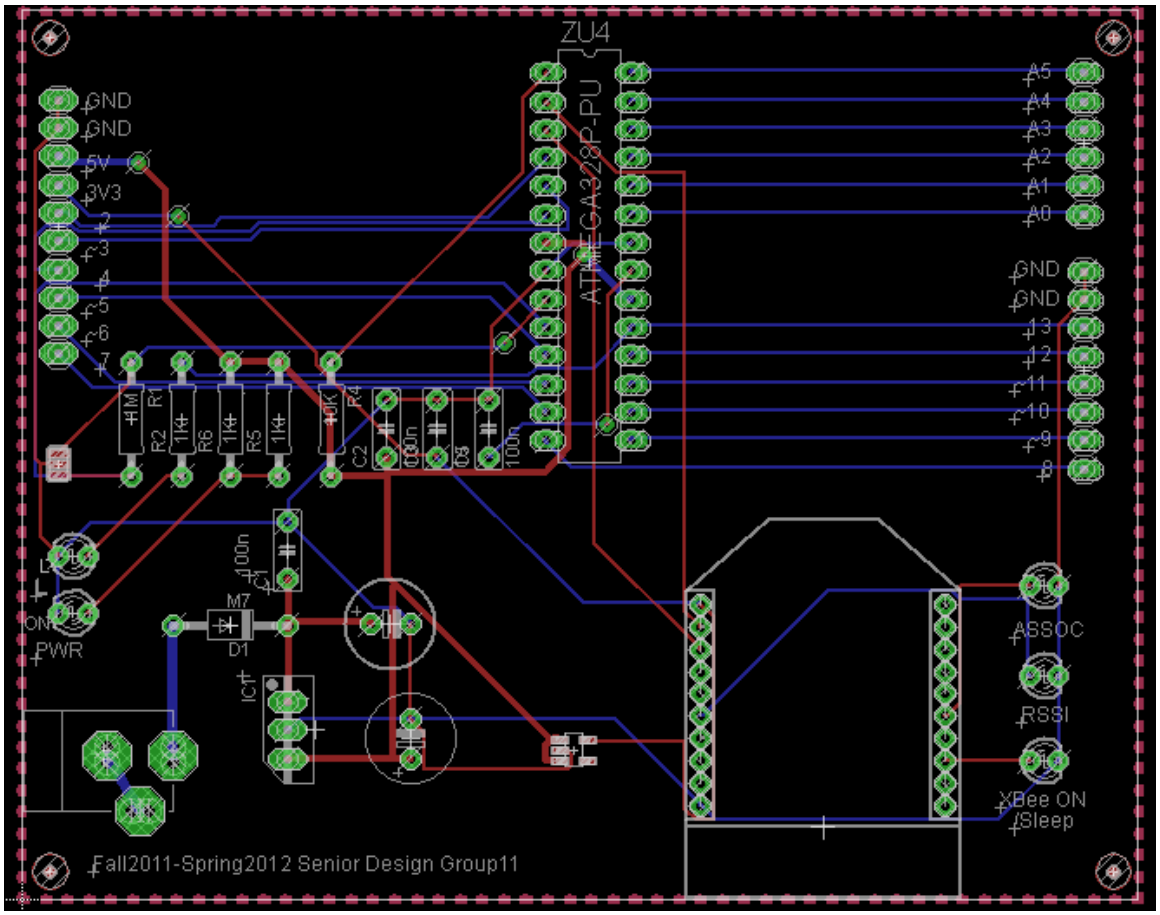


Figure 31 PCB Schematic

7 PROJECT PROTOTYPE TESTING

7.1 Component Test Procedure

Because the project has so many different components and subsystems, it is essential to test each individually so as to pinpoint and resolve any specific issues before combining them together into a cohesive whole. The testing process is arranged so that the microcontroller/motor testing can be done in parallel with the image processing and the wireless communication testing, since all three procedures work relatively independent of one another, while the rangefinder relies on the wireless connection. After each system testing had been successfully completed, they were integrated together for the testing of the entire system.

The following procedures detail the specific steps necessary to test the hardware and software components. If the expected outcome does not occur, or does not include all of the conditions of success, the system under test must be subject to thorough analysis to determine the problem. When it is discovered and resolved, the same test was re-administered, and the cycle continued until a successful outcome was forthcoming.

7.1.1 Operational Constraints

Because of the complex nature of the project, and the large number of subsystems that must be integrated together for the seamless operation of the turret, it is desirable to limit any factors which could detrimentally impact the successful functioning of the machine. The end result would be merely a prototype; it was not meant to be a final product ready for distribution. This allowed the group the freedom to test and operate the system in a relatively stable and predictable environment. When the project was successfully operating under these conditions, then more variables was introduced.

In the visual image processing program, the background and foreground layers were separated for the purpose of isolating the targeted object. While the code is meant to handle minute fluctuations in pixels, any large discrepancies would result in errors. To reduce the probability of this occurrence, a solid colored background would be ideal for testing, such as a blank wall. In addition, the test subject's clothing should be a different color than the chosen background so as not to be confused with part of the wall. For example if the background is a white wall and the test participant is wearing a similarly colored white shirt, the program does not read a difference in value for those pixels, and it therefore sees their head as separate from their bottom half, targeting the two disjointed pieces instead of the whole person. This not only results in two targets instead of one, it also gives a smaller area on which to aim, increasing the probability of missing either target.

Additionally, environmental concerns such as wind and rain would negatively affect the image processing, again causing large fluctuations in the current images in comparison to the reference frame. Another issue would be a lack of clear lighting, for instance trying to use the system at night. This would likely cause the opposite problem, that is, the change in pixels would fall within the acceptable limits, due to the decreased saturation of everything in the image. The solution to both of these issues, either too great or too little changes in value, is to isolate the system from these factors by operating it inside a building. The consistent lighting and lack of unpredictable weather optimize the group's chances for getting positive results.

7.1.2 Servo Control

7.1.2.1 Arduino Servo Library

Purpose: To use Arduino Servo Library to convert analog data from the user interface to digital data

Procedure:

1. Connected Arduino to a PC
2. Opened the IDE for simulation
3. Opened the user interface on the tablet
4. Sent a set of coordinates from the user interface to Arduino
5. Checked the output and see if it is in PWM

Expected Outcome: Arduino outputs some value of angle with a cycle rate

Conditions of Success:

- Arduino converts analog data to digital data
- Arduino gives the correct angle and cycle rate
- Arduino constantly takes inputs from the user interface until it stops receiving coordinates

7.1.2.2 Servos

Purpose: To make sure that the servos perform smooth movement

Procedure:

1. Connected servos with power supply
2. Varied input voltage (4-6V)
3. Checked the speed of the servos, which should increase as input increases
4. Cut off the power and see if it orients itself back to the center point

Expected Outcome: Servos respond to various inputs

Conditions of Success:

- Smooth movement with various inputs
- Speed increases as input voltage increase
- Orientates itself back to center point as power is cut off

7.1.2.3 Arduino- Servos

Purpose: To control the servo using Arduino Servo Library to examine if the coding is correct

Procedure:

1. Connected Arduino board to a PC
2. Connected Arduino to the servos using wires
3. Made sure the servos are ground on the Arduino board
4. Powered on Arduino
5. Sent sets of coordinates to from PC to Arduino
6. Observed the movement of servos

Expected Outcome: Some servo movements

Conditions of Success:

- Servos move to the designated position
- Servos stay at the position as long as the given rate

7.1.2.4 PID Controller

Purpose: To test the PID controller and check if it can eliminate errors and avoid overshoot that would possibly burn out servos.

Procedure:

1. Built the circuit using Multisim for simulation purpose.
2. Added scopes on both open loop and closed loop outputs
3. Varied the reference position (input value)
4. Recorded the output data
5. Calculated the error percentage and see if it is acceptable. If not, recalculated the gains in each loop and repeat step from 1 to 5.
6. Built the prototype
7. Varied the reference position
8. Recorded output data and compare it with simulation data
9. Calculated the error percentage
10. If the error percentage is not acceptable, check wire connection (troubleshoot required)

Expected Outcome: PID controller gives responses to input

Conditions of Success:

- Open loop output value is close to the input value
- Closed loop output value is close to its current position value
- The error percentage is small
- Both outputs perform smoothly
- Overshoot does not occur
- Electronic components do not burn out due to constant change of input

7.1.2.5 Arduino- PID Controller -Servos

Purpose: To eliminate error to the maximum and avoid overshoot using a PID controller

Procedure:

1. Connected Arduino board to a PC
2. Added PID controller unit between Arduino and servos
3. Fed Arduino digital output into PID controller and connect the output end to servos
4. Varied sets of coordinates
5. Observed servo movements

Expected Outcome: Some servo movement

Conditions of Success:

- Smooth servo movement
- Servos move to designated position smoothly without swings
- Servos orientate themselves back to center position after each execution

7.1.2.6 Servo Driver

Purpose: To check if the circuit works correctly as an on-off switch for servos

Procedure:

1. Built the circuit using Multisim for simulation purpose.
2. Varied the input and observe the output response. The circuit should perform a smooth output for a period of time without change of input
3. If the circuit performs properly, build the prototype
4. Recorded each output to the corresponding input (using the same set of data in simulation
5. Compared results

Expected Outcome: The circuit performs an output with a corresponding input

Conditions of Success:

- Output voltages are within proper range
- Circuit performs for a certain period of them with a corresponding input
- Output is smooth and steady for a certain period of time
- Circuit does not burn out due to a constant change of input

7.1.2.7 Servo Control System

Purpose: To connect all sub servo control units together and test the system if servos perform accurately as designed

Procedure:

1. Connected Arduino board to a PC
2. Connected the output of PID controller to servo drivers
3. Connected servo drivers to servos
4. Opened the IDE on the PC and input various sets of coordinates
5. After setting the values, ran the software
6. Recorded the output data with each corresponding input data
7. Observed the speed and position of servos

Expected Outcome: Servos perform smoothly without overshoot

Conditions of Success:

- All units are compatible
- Servos move to designated position
- Smooth movement without swings
- Duration at the designated position as assigned
- Accelerate to certain speed without overshoot
- Servos are not burnt out due to acceleration
- Self-orientation to center point
- Servos do not disorient due to stress
- Electronic components are not overheated due to a constant performance
- The system works consistently

7.1.3 Image Processor Testing

7.1.3.1 OpenCV Interfaces with Camera

Purpose: To make sure the wireless camera successfully connects to the computer, and that OpenCV commands are implemented correctly with respect to the incoming images

Procedure:

1. Powered on the camera and PC

2. Checked that necessary drivers for the camera are installed on the PC
3. Ran a simple program using the OpenCV functions for getting images from the camera

Expected Outcome: The video fed from the camera displays in a window on the computer monitor

Conditions of Success:

- Picture is sharp and clear
- Motion is not blurry
- Delay is minimal

7.1.3.2 Motion Detection

Purpose: To use the OpenCV library to effectively determine if there is a moving target within the range of the turret and to read said target as a blob

Procedure:

1. Powered on the camera and PC, confirm that the wireless connection between the two is established
2. Set camera up facing plain solid colored wall approximately 30 meters away, the range of the turret
3. Ran the background differencing section of the program, which uses the OpenCV functions for object detection
4. Had a test subject move across the field of view, testing every speed at each distance
    a. At different speeds:
        i. 7 m/s to test maximum speed detectable
        ii. 1 m/s to test low speeds
        iii. Different ranges in between, both constant and changing
        iv. Test subject stops partway through the frame, then continues after a few seconds
    b. At different distances
        i. Less than 1 meter away from turret to test close range
        ii. 30 meters away to test far range
        iii. Varying distances in between, both constant and changing

Expected Outcome: The program outputs a second window displaying the moving foreground object as a white blob and everything in the background layer as black

Conditions of Success:

- Blob matches the moving object in shape and location
- Blob is clearly defined, lacking fuzzy edges
- All white pixels are connected, minimal stray white pixels due to noise

7.1.3.3 Object Representation

Purpose: To use the OpenCV library to find the edges of the moving object so a rectangular representation can be drawn around the target

Procedure:

1. Used object detection program to display two windows, one with video from camera and one with target displayed as blob
2. Ran the edge detection section of the program
3. Had a test subject move across the field of view using the same test cases discussed in the Motion Detection procedure

Expected Outcome: The program displays a colored rectangle around the blob in the second window containing the white blob, as well as around the moving object in the original window containing the video stream from the camera

Conditions of Success:

- Rectangle encloses the entire blob, lying tangent to its outermost curves
- Rectangle is displayed correctly in the original window, surrounding targeted object
- Only one rectangle per target is displayed

7.1.3.4 Centroid Calculation

Purpose: To find the distance the gun must be moved to aim at the centroid of the target

Procedure:

1. Used object detection and representation programs to display two windows, with rectangle enclosures around target in 1st and blob representing target in 2nd
2. Ran the centroid calculation section of the program
3. Had a test subject move across the field of view using the same test cases discussed in the Motion Detection procedure
4. Selected this as the desired target

Expected Outcome: The program displays a circle to represent the centroid on the blob in the second window, and also outputs the location of the centroid

Conditions of Success:

- Centroid is located at the correct point in the blob rectangle, as verified by calculation
- The location outputted matches the centroid location

7.1.3.5 RTCDT Application

Purpose: To check that the created RTCDT application works correctly with the OpenCV functions and the user interface displays properly

Procedure:

1. Powered on the camera and PC
2. Set camera up facing plain solid colored wall approximately 30 meters away, the range of the turret
3. Opened the RTDS app to confirm that the feed from the camera is displayed in the program
4. Had a test subject move across the field of view using the same test cases discussed in the Motion Detection procedure
5. Selected the desired target using the colored button that corresponds to the rectangle

Expected Outcome: The RTDS application displays the video from the camera with the colored rectangles representing the various targets, and outputs the centroid location of the selected object

Conditions of Success:

- Delay between selection and output is minimal
- All targets are represented by uniquely colored rectangles
- Outputted centroid locations match the chosen target

At the successful completion of these test procedures using one target at a time, they were run through again, first with two test subjects, then with three. For these cases of multiple targets, every possible situation had to be tested against, with the subjects at same speeds and distances, then varying ones. In addition, they had to be tested entering the field of view simultaneously, then at different times.

7.1.4 Wireless Communication

7.1.4.1 Camera-User Interface

Purpose: To make sure that the tablet recognizes the wireless USB protocol that will be implemented in the camera system, and the video stream is successfully transmitted through the protocol without any significant delay.

Procedure:

1. Powered on both camera and the tablet.
2. Opened the wireless software installed on the tablet.
3. Ran the software so that the camera can start transmitting data.
4. Observed if the video shown on the table has any delay.

Expected Outcome: The video stream coming from the camera is clear and has no significant delay

Conditions of Success:

- Tablet finds the protocol and can receive data from the camera
- Video stream is shown on the tablet as the software is running
- Video streams in real time manner

7.1.4.2 Microcontroller – RF Wireless Module Interface

Purpose: To make sure that the RF wireless module is successfully connected to Arduino, and those two units are compatible.

Procedure:

1. Connected Arduino microcontroller to PC
2. Powered on Arduino microcontroller
3. Before power on the wireless module, made sure it is connected to 3.3 VDD.
4. Opened the IDE software
5. Made sure the software finds the module

Expected Outcome: The software recognizes the module

Conditions of Success:

- The module and Arduino are compatible
- The IDE recognizes the module
- User can program those two units as a system

7.1.4.3 Microcontroller – User Interface

Purpose: To use the protocol to transmit data and see if there is anything lost or corrupted data occurs

Procedure:

1. Connected Arduino board to a PC
2. Power edon both Arduino and the tablet
3. Set the wireless networking to be in Ad-hoc mode on both nodes
4. On the tablet, go to *Network and Sharing Center* and find the module protocol
5. Connected those two ends
6. Opened and run the IDE software on the PC
7. Sent a simple digital data from the tablet to Arduino
8. Observed if both input and output are appeared on the PC

Expected Outcome: Arduino responds to the tablet

Conditions of Success:

- Input is not corrupted or lost through transmission
- Arduino receives data from the tablet without delay
- Arduino responds instantaneously to the tablet (power-state changes)

7.1.4.4 Camera Interface with OpenCV on Tablet

This test procedure matches the one for 7.1.3.1 OpenCV interfaces with camera, but checks the wireless connection of the camera and the tablet instead of the camera and the computer. The purpose, procedure, expected outcome, and conditions of success mirror those written for section 7.1.3.1.

7.1.4.5 RTCDT Application on Tablet

This test procedure matches the one for 7.1.3.5 RTCDT Application, but uses the application on the tablet in place of the simulated one on the computer. The purpose, procedure, expected outcome, and conditions of success mirror those written for section 7.1.3.5.

7.1.5 Rangefinder Testing

7.1.5.1 Rangefinder Program on PC

Purpose: To check that the rangefinder finds the depth information of the desired target and outputs this information for use by the microcontroller in servo commands

Procedure:

1. Powered on the image sensor and laser pointer configuration
2. Connected image sensor to PC
3. Ran the program to find the range from the image sensor frames
4. Set system up facing plain solid colored wall approximately 30 meters away, the range of the turret
5. Placed an object at varying distances from the laser pointer system
6. Directed laser pointer and attached image sensor to target the object

Expected Outcome: The rangefinder program outputs the depth information of the targeted object

Conditions of Success:

- The depth information matches the distance from the rangefinder to the object, as measured beforehand

7.1.5.2 Rangefinder Program on Tablet

When the test was successfully completed with the rangefinder connected to the PC, the next step was to connect the rangefinder to the tablet, which run the algorithm for determining the distance. For this the rangefinder has to be connected to the microcontroller, which then sends the captured frames wirelessly to the tablet for processing.

Purpose: To check that the rangefinder program works correctly on the tablet, both in receiving information wirelessly from the microcontroller and outputting the distance of the object

Procedure:

1. Powered on the image sensor and laser pointer configuration
2. Connected image sensor to microcontroller
3. Ran the program to find the range from the image sensor frames on the tablet
4. Set the rangefinder to face a plain solid colored wall approximately 30 meters away, the range of the turret
5. Placed an object at varying distances from the laser pointer system
6. Directed laser pointer and attached image sensor to target the object

Expected Outcome: The rangefinder program will output the depth information of the targeted object

Conditions of Success:

- The depth information matches the distance from the rangefinder to the object, as measured beforehand.

7.2 System Test Procedure

The project consists of four major subsystems. They are wireless communication, image processing, user interface, and servo system. Under each subsystem, there are more control units that run upon each other. It is essential to test each sub system before integrating them together for the testing of the entire system. The group used the same approach to test the entire system. Instead of integrating all subsystems together at once, the group had two break points between user command module and firing control. Having break points or check points allows the group to monitor if any malfunction has occurred during each execution as a command has been sent from the user interface.

7.2.1 User Command

Purpose: To check if the user interface is successfully running on the image processing module.

Procedure/ Expected Outcome:

1. Powered on both camera and the tablet
2. Ran the camera and open the user interface application
3. Stationary objects shown on the screen were well defined. Moving objects had clear edges
4. Delay between camera and the user interface display was minimal
5. The display the moving target outlined by rectangles.
6. Besides automatic targets, a manually chosen point was recognized as a target
7. Selected one of the potential targets which was outlined as instructed
8. Switched firing to be on, which continuously fired at the target

Conditions of Success:

- User interface is programmed correctly without any malfunction during execution
- Colored command buttons are displayed at the bottom of the screen
- Image displayed on the screen is clear and sharp without delay
- Targets are represented by uniquely colored rectangles
- Outputted centroid locations match the chosen target
- The firing command is sent as the button is pressed

7.2.2 Firing Control

Purpose: To check if the coordinates of designated target data is successfully calculated and sent through the wireless protocol without corruption. Also, to check that if the firing command is processed by Arduino successfully and received by the laser pointer driver to trigger the diode. The team is going to integrate two major subsystems to test the entire system. This test module will be the complete system for the project.

Procedure/ Expected Outcome:

1. Connected PCB board to the user interface wirelessly. Check the setting for both ends, which should be in Ad-hoc mode.
2. Powered on all subsystems, including camera
3. Ran the camera and open the user interface application on the tablet
4. Examined if stationary objects displayed on the screen are clearly defined and automatic targets are uniquely outlined by rectangles
5. Examined if moving target are correctly tracked by colored rectangles
6. Examined if command buttons are shown at the bottom of the screen
7. Chose a designated target
   a. Automatic moving target
8. Centroid is shown on the screen
9. Servo moved to designated position.
10. A firing command automatically was sent when laser was aimed at selected target

11. Laser pointer illuminated the target with a bright spot of light for 0.5 seconds
12. As the target moved, servo continued tracking it until target is out of range.
13. Centroid display was gone, and the application was ready for the next command
14. Chose a designated target
    a. Manually chosen point on display
15. The chosen point was represented by a dot
16. Sent firing command by turning on firing switch
17. Centroid was displayed on the screen
18. Servo moved to designated position.
19. Servo moved back to center position after execution
20. Centroid point was gone, and the application became available for the next command

Conditions of Success:

- User interface is intuitive
- Screen display is clear and objects are well defined
- User is able to send command
- Microcontroller receives data from the user interface
- Servos receive signal from microcontroller
- Servos move to designated position accurately
- No overshoot or swings occur while servos are tracking targets
- Self-orientation performs accurately
- No malfunctions occur in any of the subsystem while operation
- No components are burnt during operation.

8 PROJECT OPERATION

The Remote Defense Turret was specifically targeted to be simple and intuitive, even for a first time user. Therefore, it does not require any extensive amounts of training or many complicated steps for operation.

8.1 Power

As mentioned, the turret is powered through AC wall adaptors, meaning user input is not necessary aside from plugging them in. The tablet runs on the included battery, which will need to be charged on occasion, but again requires minimal effort on the user's behalf.

8.2 User Interface

Once all of the turret's systems are powered on, and the tablet as well, the user may now operate the turret through the user interface by following these steps.

1. Locate and open the file TargetAquisition.exe
2. Two windows, Live and Thresh, should open as shown in Figure 32 below. The Track switch on the right switches track mode on and off, and consequently the binary image, which is displayed in the Thresh window when tracking is on. The Fire switch turns the continuous firing on for tracking mode.

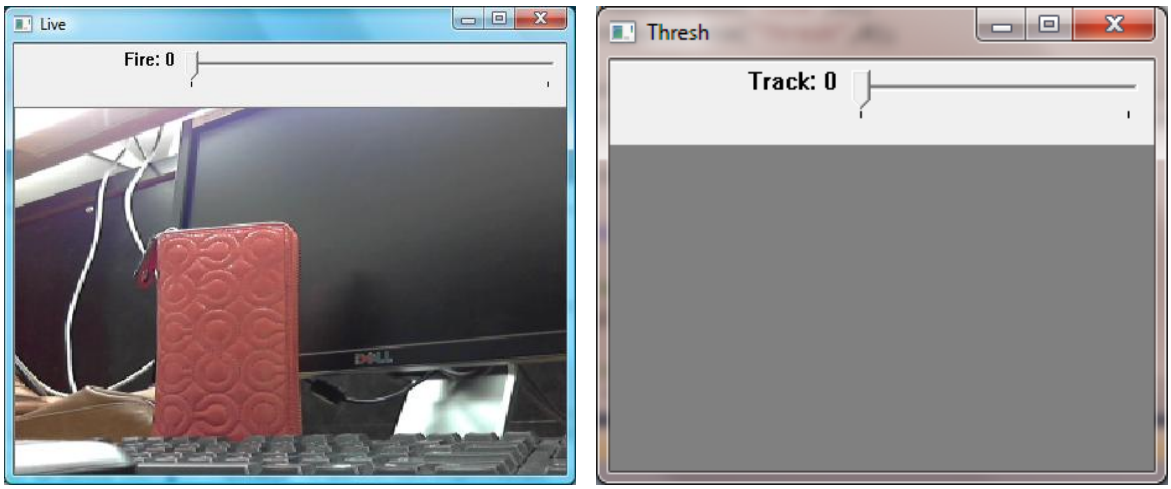


Figure 32: UI with Track and Fire set to 0

3. Toggle Track on and select the target to get a similar image to the one displayed in Figure 33.

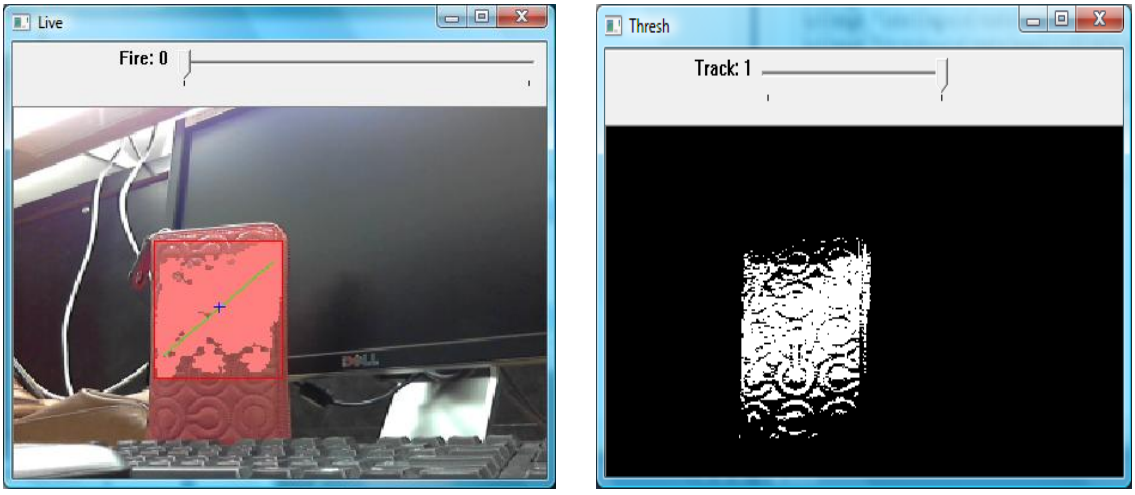


Figure 33: Tracking mode on

4. The turret will continue to track the object as it is moved around the frame. If fire is switched on, the laser will continuously flash on the target as well.
5. To enable stationary target selection mode, turn track and fire off.
6. Select the target of choice and hold the selection until the selection verification icon forms a complete circle. The turret will immediately move to aim at that point and flash the laser for 2 seconds.

8.3 Troubleshooting

Here are some solutions to commonly encountered issues

- If the XBee connected to the tablet is not flashing the transmit LED, check in Device Manager that the serial port it is connected to corresponds with the one written into the program.
- If the laser pointer is not firing correctly, a simple system reset will usually fix the problem.
- Make sure all systems are receiving power by checking each one's power indicator LED

9 ADMINISTRATIVE CONTENT

9.1 Milestone

During the first semester, extensive research was done on previous similar projects, so that the group could find potential different approaches to achieve its overall project goals, and to expand its knowledge on the matter. In addition, a detailed design had to be finalized for the whole system. To maximize the functionality of the project, a great amount of time had been spent on research, looking into different types of electronic components that could be most successful to meet the project specifications. The group was also aware of the difficulties that might be encountered while constructing the system. In order to avoid last minute problems, the decision was made to start the most important, also the most difficult part of the project during winter break. That encompasses both the software and mechanical aspects of the project.

Since none of the group members were hardcore programmers, the processes of target acquisition and user interface was started early during the winter break. This way the team members could singularly concentrate on the matter without other school work burdens that they might have during the semester. The goal of programming early during the break was to become more familiar with the OpenCV library and more comfortable with coding. In the project, multiple subsystems were being implementing that are relying on each other, in both software and hardware aspects. It is essentially important that each layer of subsystem works perfectly on its own and is able to communicate to each other as a whole system.

In the hardware aspect, the turret system was set up during the break. Since the group used a turret kit that contains all the premade parts, it did not have to be built from scratch. The major task of the turret system was to move the servos as smoothly as possible and position them as accurately as possible. Also, because the servos would be tracking moving targets, the team had to determine the maximum speed a moving target can have without burning out the system or causing overshoot.

To get started early, purchases for two servos and an Arduino microcontroller for the turret system had already been made before winter break. For the user interface and target acquisition, the code was run on a PC before implementing the programs on the tablet. This is a way to check that the program works as intended, without the added complication of incorporating the Windows system. Once the spring semester started in January, the group was able to test the turret system and had a good knowledge of programming to finish the project.

The overall schedule for the two semesters is laid out in the table below. The first semester was focused largely on the research and report. In addition, a few components were acquired, and each member started working individually with

OpenCV to familiarize themselves with it. The second semester was devoted to the physical construction of the turret and completion of the coding, with the second half of the semester focused on testing the system.

Table 12: Milestone Chart

WEEK	DATE	Objective
1	9/26/2011	Project Proposal
2	10/3/2011	Break down the project into small sections
3	10/10/2011	Detailed research on each section
4	10/17/2011	Individual Research
5	10/24/2011	Acquire components
6	10/31/2011	Begin Coding
7	11/7/2011	Documentation
8	11/14/2011	Schematic/Projet Diagram
9	11/21/2011	Final Draft Report
10	11/28/2011	Final Report
	12/5/2011	***Finals (1 week)***
	12/12/2011	Target Acquisition
	12/19/2011	
	12/26/2011	User Interface: Create Interface
	1/2/2012	Hardware Testing
1	1/9/2012	Image Processing: Target recognition & Distance caculation
2	1/16/2012	Image Processing : Edge Detection, Motion Detection
3	1/23/2012	User Interface: Implement on Android tablet
4	1/30/2012	Motor Control, Target Acquisition
5	2/6/2012	Marker Reorientation
6	2/13/2012	Wireless communication
7	2/20/2012	Prototype
8	2/27/2012	Testing
9	3/5/2012	
10	3/12/2012	
11	3/19/2012	Modification
12	3/26/2012	
13	4/2/2012	
14	4/9/2012	Project Report
15	4/15/2012	Final Project

9.2 Budget and Finance Discussion

At the conception of this project, it was evident that the financial burden would not be light, due to the large amount of components required and the complex technology needed to implement such a design. For these reasons, the group decided to look for a sponsor to handle most, if not all, the relevant expenses. However, the group was limited by the fact that many of the companies willing to sponsor projects already had a project design in mind. Fortunately, there were two companies that allowed senior design groups the creative freedom to develop their own designs, as long as they fell within the general categories specified by the companies. The first was Progress Energy, which was geared towards helping those projects focused on renewable and sustainable energy. The second was Workforce Central Florida, which allotted financial support for projects in the categories of homeland or cyber security, renewable and sustainable energy, biotechnology, and digital media or modeling. This gave the group the choice to either alter the original idea to meet the criteria of sustainable energy for Progress Energy, or to maintain the original design and submit the proposal to WCF under the Homeland Security label. The group chose the latter option, and the budget, shown in the table below, was approved for full funding from the company, with the stipulation that the group would purchase the products themselves and then receives reimbursement for submitted receipts.

In addition to the specific hardware components needed to construct the turret, it was known that a large amount of additional equipment would be necessary for testing purposes, such as a multimeter, oscilloscope, and a power supply. The facility provided for senior design students was the Senior Design Lab, where much of this equipment was available for student use. The project also required smaller circuit components for building the PCB prototype, such as resistors, capacitors, diodes, and wires, as well as the soldering tools needed to secure them to the board. The soldering tools were available in the lab, but many of the components the group had to obtain itself. For this reason, there was included in the budget a fee for miscellaneous electrical components in the amount of $150.00, which covered any unforeseen small electrical pieces such as potentiometers, wire, solder, electrical connectors, heat-shrink tubing, wire strippers, or any other parts that were needed. Also included in the budget was the price for manufacturing the PCB itself. The group decided to have 2 copies made as a precaution, in case the first one broke or was otherwise rendered inoperable. Another included expense was $400.00 for miscellaneous mechanical parts, including wheels for transport, Plexiglas housing, mechanical connectors, buffer for potential breakage and learning curve, since it was expected that mistakes would be made and components would need to be replaced.

Table 13: Project Budget

Item	Qty	Unit Price	Total Price
Wireless Transmitter	1	$6.30	$6.30
Arduino Uno	1	$25.00	$25.00
Camera	1	$250.00	$250.00
Wireless USB adapter	1	$129.95	$129.95
Motor Controller	1	$59.99	$59.99
Nitrogen tank	2	$49.95	$99.90
RC switch	1	$24.00	$24.00
Main pan/tilt mount	1	$199.00	$199.00
Servo Extensions	2	$4.95	$9.90
PCB Fabrication	1	$147.50	$147.50
Misc. mechanical parts	1	$400.00	$400.00
Misc. electrical parts	1	$150.00	$150.00
Electronic Paintball Marker	1	$399.00	$399.00
Paintball Hopper	1	$36.95	$36.95
Basic parts, required by mount vendor	1	$49.00	$49.00
Laser Rangefinder	1	$349.95	$349.95
User Interface Tablet	1	$799.99	$799.99
Pan-and-Tilt servos	2	$46.99	$93.98
Laser Pointer	1	$39.70	$39.70
Wifi Communicator Arduino Shield	1	$69.99	$69.99
Panasonic 1024 Linear Image Sensor	1	$30.00	$30.00
		Total:	$3,370.10

9.3 Mentors

The industry mentor chosen to meet the Workforce Central Florida requirements is Dr. Robert Muise. Dr. Muise is a Senior Staff Engineer at Lockheed Martin Missiles and Fire Control. He holds a PhD in mathematics from the University of Central Florida. He is responsible for leading engineering teams in applied research in the areas of image processing, compressive/computational sensing, automatic target detection/recognition, and image/data compression. His expertise is in algorithms for signal/image processing, computational linear algebra, and integrated sensing and processing. Dr. Muise holds two patents, has published many journal and conference papers, is a member of SIAM, and a senior member of IEEE. He is also part of the faculty in the University of Central Florida Department of Mathematics.

In addition, the group asked Dr. Niels da Vitoria Lobo to act as an unofficial mentor to aid with issues in the computer vision portion of the project. Dr. Lobo received his B. Sc. (Honors) degree in Mathematics and Computer Science from Dalhousie University, Canada, and the Ph.D. in Computer Science from the University of Toronto.

APPENDIX A

9.4 Written Authorization

Authorization given by Fairchild Semiconductor Corporation

From http://www.fairchildsemi.com/legal/index.html, it states that " *you may download one copy of the material (including any document, information, data, software (including all files and images contained in or generated by the software, and any accompanying data) or other materials provided that (1) you do not delete or change any copyright, trademark, or other proprietary notices in or on any such materials; (2) you include the copyright notice "© Fairchild Semiconductor Corporation" on any copy of the material or any portion thereof, and (3) you use the material only for non-commercial informational purposes. Modification or use of the materials for any other purpose violates Fairchild's intellectual property rights. The material in this site is provided for lawful purposes only*." Since this documentation is only for academic and non-commercial purpose, the materials are deemed to be licensed to the group.

Authorization given by Atmel Corporation

From http://www2.atmel.com/About/legal.aspx, it states that *"Materials from this website www.atmel.com and any other website owned, operated or controlled by Atmel and/or its affiliated or subsidiary companies (together, Atmel) are owned and copyrighted by Atmel. Unauthorized use of such Materials (e.g., information, documentation and software), including these Terms, may be a violation of Atmel's intellectual property rights or other applicable laws. If you agree to these Terms, you may download (on a single computer), copy or print a single copy of all or a portion of the Materials for informational, non- commercial, lawful purposes only*." Since this documentation is only for academic and non-commercial purpose, the materials are deemed to be licensed to the group.

Authorization given by Microchip

From http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=487&param=en023282 ttp://tinyurl.com/microchip-com, it states that *"If you use Microchip copyrighted material solely for educational (non-profit) purposes falling under the "fair use" exception of the U.S. Copyright Act of 1976 then you do not need Microchip's written permission. For example, Microchip's permission is not required when using copyrighted material in: (1) an academic report, thesis, or dissertation; (2) classroom handouts or textbook; or (3) a presentation or article that is solely educational in nature (e.g., technical article published in a magazine). Please note that offering Microchip copyrighted material at a trade show or industry conference for the purpose of promoting product sales does require Microchip's permission."* Since this documentation is only for academic and non-commercial purpose, the materials are deemed to be licensed to the group.

Authorization given by weburban.com

From: **Robert Newport** <rob@weburban.com>
Date: Sat, Dec 3, 2011 at 2:22 PM
Subject: Re: Contact Form
To: brad.clymer@gmail.com

Please feel free to use the pictures for academic or non-academic purposes!

Also, feel free to ask questions should you have them.

Regards,

-Rob.

web | urban . **rob newport** . chief technology officer
mobile . 323 . 333 . 1800
3400 w olive ave. suite 300, burbank 91505

On Dec 3, 2011, at 7:35 AM, weburban wrote:

Name: Bradley Clymer
E-mail: brad.clymer@gmail.com
Telephone: 4077335101

Comment: Hey there! I'm, with a senior design group at the University of Central Florida. We plan to utilize your WiFi board to connect a touchscreen interface to our microcontroller, and we'd like formal permission to use figures and pictures of it shown on your website in our preliminary report. Note that this is a strictly academic and non-commercial use of the information.

Also, great work on the website!

9.5 Bibliography

alibaba.com. (n.d.). *High Resolution CCTV Camera Infrared USB Camera with 23pcs IR LEDs: Alibaba.com*. Retrieved November 7, 2011, from http://www.alibaba.com/product-gs/483295591/High_Resolution_CCTV_Camera_Infrared_USB.html

Ambery.com. (n.d.). *Composite BNC S-Video Audio To USB Adapter: Ambery.com*. Retrieved November 7, 2011, from Ambery.com: http://www.ambery.com/cobncsautous.html

Arduino. (2011). Retrieved 11 2, 2011, from Arduino: http://arduino.cc/en/Main/ArduinoBoardUno

Baranda, J., Espiritu, J., & Thompson, D. (2007). *Motion-Tracking Sentry Gun.* Senior Design Project Documentation.

Catron, M., Monetti, S., & Rodriguez, O. (2008). *Paintball Targeting System.* Senior Design Project Documentation.

Colon, H., Horton, A., Steighner, K., & Yielding, N. (2011). *Automated Targeting Proximity Turret.* Senior Design Project Documentation.

Danko, T. (2009, August). *Webcam Based DIY Laser Rangefinder*. Retrieved 11 1, 2011, from https://sites.google.com/site/todddanko/home/webcam_laser_ranger

Edmund Optics. (n.d.). *Near IR Camera - Edmund Optics*. Retrieved November 4, 2011, from http://www.edmundoptics.com/products/displayproduct.cfm?productID=2384

Edwards, S. (2011, February). Microprocessors or FPGAs?: Making the Right Choice. *RTC* .

Free Software Foundation, Inc. (1991, June). *GNU General Public License v2.0 - GNU Project - Free Software Foundation (FSF)*. Retrieved November 7, 2011, from GNU Opearating System: http://www.gnu.org/licenses/gpl-2.0.html

Hannifin, P. (n.d.). *Fundamentals of Servo Motion Control.* Retrieved November 13, 2011, from Parker Motion: http://www.compumotor.com/whitepages/ServoFundamentals.pdf

Hoagieshouse.com. (n.d.). *How to make a webcam work in infra red: hoagieshouse.com*. Retrieved November 7, 2011, from http://www.hoagieshouse.com/IR/

IDS Imaging. (n.d.). *Home - Image processing, industrial cameras, uEye, GigE, USB*. Retrieved November 6, 2011, from http://www.ids-imaging.com/

Kaehler, A. a. (2008). *Learning OpenCV.* Sebastopol: O'Reilly Media.

LINKSYS by Cisco. (n.d.). *Wireless-N Internet Home Monitoring Camera*. Retrieved November 23, 2011, from LINKSYS by Cisco: http://downloads.linksysbycisco.com/downloads/datasheet/WVC80N_DS_V10_NC-WEB.pdf

Logitech. (n.d.). *Logitech QuickCam® Orbit AF*. Retrieved November 5, 2011, from http://www.logitech.com/en-us/38/3480

maxmax.com. (n.d.). *Sony DSC-S980: maxmax.com*. Retrieved November 7, 2011, from http://www.maxmax.com/sony_dsc-s980.htm

Soule, K. (2010, June 3). *Video Road.* Retrieved 11 5, 2011, from Understanding Color Processing: 8-bit, 10-bit, 32-bit, and more: http://blogs.adobe.com/VideoRoad/2010/06/understanding_color_processing.html

stefanv.com. (n.d.). *Choosing and Using Nickel-Metal-Hydride (NiMH) Rechargeable Batteries*. Retrieved November 7, 2011, from http://www.stefanv.com/electronics/using_nimh.html

weburban.com. (n.d.). *Weburban Maple wireless 802.11b/g/n WiFi wireless card for Oak Arduino*. Retrieved November 20, 2011, from weburban.com: http://store.weburban.com/store-weburban/wifi.html

Wikipedia. (2011, November 19). *Poly(methyl_methacrylate)*. Retrieved November 22, 2011, from http://en.wikipedia.org/wiki/Poly(methyl_methacrylate)

Wireless USB from the USB-IF. (n.d.). Retrieved from Universal Serial Bus.

9.6 Personnel

9.6.1 Brad Clymer

Brad Clymer is currently an electrical engineering senior at University of Central Florida. His interests include biomedical nanotechnology and micro-electromechanical engineering. He is graduating in May 2012.

9.6.2 Courtney Mann

Courtney Mann is currently an electrical engineering senior at University of Central Florida. Her interests lie in communication and controls. She is graduating in May 2012. She is planning on going to graduate school.

9.6.3 Szu-yu Huang

Szu-yu Huang is currently an electrical engineering senior at University of Central Florida. Her interests include optics and biomed electrical engineering. She is graduating in May 2012.