Nerf-Enabled Battlebot with Automated Target Detection using Multiple Sensing

Department of Electrical Engineering and Computer Science
University of Central Florida
Dr. Lei Wei

Sponsor: Lockheed Martin

Final Document

Group 9

Aaron Hoyt	Electrical Engineering
Daniel Agudelo	Computer Engineering
Rafael Ramirez	Computer Engineering
Rachel Gremillion	Computer Engineering

Table of Contents

1. Executive Summary

Robots are truly an innovative technology and have the ability to perform tasks that humans could not to do alone. As the age of computer programming and electronic circuitry progresses, robots are becoming less limited to solve problems and create a better lifestyle. There is no doubt that without the aid of robots manufacturing would not be made possible, dangerous missions would not be carried out, and automation would not exist. Some would say that robots are faster and smarter than humans since computing can be processed at lightning speed. Although robots do not take on the full nature of a human being, they do require some parts to be able to act like a human. Most robots contain a brain for processing information and the use of sensors that act as eyes for interacting with the outside world. Robots would also contain arms and legs or wheels and motors to help them move around. Some robots are built for everyday use, some robots are used for fun, and there are also robots that can be used with Nerf-blasters.

A human can be able to detect a field of interest and identify a target and shoot these targets with a nerf-blaster, but what if a robot could do the same thing yet faster and with more accuracy? This project is set out to use technology in order to create a robot that would be able to automatically find targets and fire upon them with the use of Nerf-blasters. This will allow faster reaction time and accurate readings of targets all automatically. The robot will able to map out an area using sensors and cameras and would be trained on what targets to fire a nerf-blaster upon.

This project documentation paper goes in detail with the motivation of the project, extensive component research, algorithm, and computer programming research. There is also research dedicated to the related standards on the market today that are compared to the different areas where this project would utilize these standards. There are sections devoted to strategic hardware and software design as well as physical chassis designs. Each section is labeled with the title and a brief description of the title and its subsections.

This project is funded and overseen by Lockheed Martin with an overall $2,000 budget - $1,000 maximum for an as-demonstrated robot cost. This project is one of three robot projects each with a full engineering team consisting of computer science, electrical, mechanical, and computer engineering majors. All three robot projects will participate in a competition set out to battle against each team's robot to find out which one would be the best in automatic detection and accurate firing mechanisms.

2. Project Description

This chapter covers the motivation, objective and goals for this project.

2.1. Motivation

The motivation for this project to explore the possibilities of what it takes to make robot automation a reality. Also taking up the opportunity of managing among other engineering disciplines to get an idea what it takes to plan a project with several different departments. The challenge will be understanding how a system can map an area using sensors and create an image within the mapping. Since this type of project has not been approached before, it will be interesting to be able to utilize each person on the team and put their education to the test and to be able to practice research strategies as a team.

2.2. Objective and Goals

The objective for this project, amongst other things, is to build a robot that has capabilities to be manually navigated through wireless connection. The robot will also contain an automatic system that uses algorithms and sensors to automatically search and detect objects, such as targets. There must be a combination of two sensor modalities using examples such as mid-wave infrared imagery, LIDAR point clouds, visible spectrum imagery or radar returns. These targets will be fired upon automatically using Nerf-blasters, darts, and balls.

When the entire robot is completed, the goal for this project is to win a competition that will take place against two other robots with the same automated system. This competition has a set of rules and regulations to follow. All the requirements for this project can be found section 2.3, Requirements and Specifications.

As for the competition, Figure 2.1 displays an example of the playing field. Two opposing robots will be placed in their respective zones, A and B, which will have dimensions of 20ft x 20ft. There will be a 20ft x 10ft restricted zone with obstacles in the center of the field. In order to win the competition, the robot must be able to score the most amount of points.

The point system is as follows: Two points for enemy bot impact with Nerf-ball. Four points for enemy bot impact with a nerf dart. Two points for course target impact in opposite zone; choice of ammo can be ball or dart. Only a maximum of two hits per match is scored on stationary targets. Eight points is scored for impacts with enemy target robot medic. The robot medic takes on the responsibility of repairing the robot during the match and if the opposing robot can detect the medic, it is allowed to shoot at them. Only one robot repair is allowed per round. A deduction of five points will be in effect if the robot enters the restricted zone or moves out of bounds.

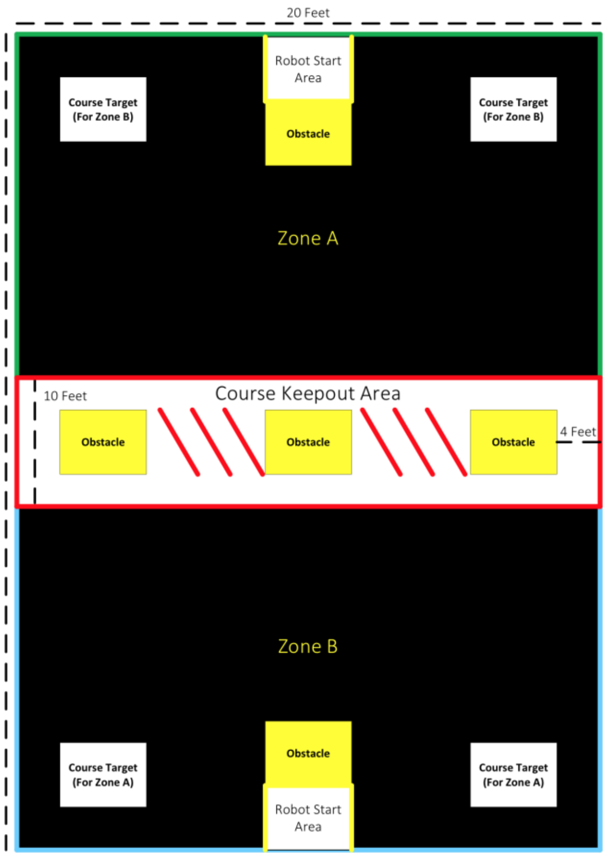


Figure 2.1: Example of the playing field

2.3. Requirement Specifications

This project has a set of requirement specifications to ensure that the robot is being built correctly and successfully. Lockheed Martin has set specifications to create a standardized robot for the competition and some members of the group conducted some specifications that are abstract, unambiguous and verifiable. The following sections note the requirement specification as well and the details on how to verify them.

2.3.1. Size

The first engineering specification is the size, seen in Table 2.1. Lockheed Martin set this standard so that the robot cannot be too large or inadequate to hold all the components of the robot.

Requirement ID	Description	Verification
S1	Total robot dimensions shall not exceed 3ft x 3ft x 3ft.	Standard to battlebot competition

Table 2.1: Size Requirements

2.3.2. Object Detection

There are four engineering requirements in object detection, seen in Table 2.2. The requirements are specified by how far the target is, what kind of target and fire promptly on the appropriate targets.

Requirement ID	Description	Verification
OD1	Be able to automatically detect objects to a distance up to 45ft.	Standard to battlebot competition.
OD2	To be able to automatically detect and highlight three different targets with a 51% accuracy rate.	Standard to battlebot competition.
OD3	Must be able to automatically determine the distance of three targets up to 50ft.	Simulating various targets at different ranges.
OD4	Be able to detect moving target with a movement speed range of 1 to 3 m/s.	Utilizing a sample robot for target acquisition.

Table 2.2: Object Detection Requirements

2.3.3. Power

There are two power requirements, seen in Table 2.3, to ensure the robot stays working and can supply enough power to the robot. The table below will identify the requirement number, give a brief description and list the way to verify.

Requirement ID	Description	Verification
P1	Power supply must be able to last a minimum of 25 minutes.	Using the system for two 10 minute rounds.
P2	To be able to operate at 12V and draw a maximum of 3A.	Based on total power consumption of the system

Table 2.3: Power Requirements

2.3.4. Mobility

There are two requirements specifications on what kind of movement this robot will be allotted to have. These requirements are listed in Table 2.4.

Requirement ID	Description	Verification
M1	Be able to remotely control the Battlebot.	Testing with the use of a remote control in each direction the Battlebot must turn.
M2	Be able to track the distance it travels.	Measure the distance it moves.

Table 2.4: Mobility Requirements

2.3.5. Cost

While there are several possibilities on building a robot, there is a budget and this robot cannot exceed the demonstrated cost of $1000.

Requirement ID	Description	Verification
C1	As demonstrated cost cannot exceed $1000.	Optimizing components under $1000 for the final build. This is based on Lockheed Martin’s budget requirements.

Table 2.5: Cost Requirements

2.4. House of Quality Analysis

This section discusses a house of quality for this project. The house of quality is a diagram that shows how this project will be able meet consumer’s standards, but with some engineering tradeoffs. Since there are tradeoffs between different requirements, it is important to understand them so that they can establish the meaning of the requirement as well as provide solutions to these trade-offs.

The four marketing requirements that deem suitable for this project would be detection accuracy, firing accuracy, low power and cost. The six areas of quality tradeoffs are the range of detection, object recognition accuracy, fast processor, memory dimensions, and cost.

In the table below each requirement contains a polarity. This polarity indicates positive and negative symbols which shows a desire of the requirement. An example would be that low power is a positive requirement since it is more desirable to the project while cost is a negative requirement since the project does not want a high cost.

The arrows are for positive and negative correlation. The arrows in the upward direction indicate positive correlation in which both requirements can be improved together while the arrows in the downward direction are where the two requirements will contradict one another. An example would be having too high firing accuracy will result in more cost in technology which becomes a negative correlation to the requirements.

Table 2.6 contains the marketing tradeoffs for this project with specific number values at the bottom of the table. The legend is for the reader to understand what each symbol in the table represents. All trade-offs are considered in order to prevent an imbalance within the system and are mapped in Table 2.6. Ideally, considering all positives and negatives will lead to a complete build that will satisfy all requirements listed above in section 2.3 Requirement Specifications.

Legend

↑	Positive Correlation
↑ ↑	Strong Positive Correlation
↓	Negative Correlation
↓ ↓	Strong Negative Correlation
+	Positive Polarity (Increasing requirement)
-	Negative Polarity (Decreasing requirement)

		Range of Detection	Object Recognition Accuracy	Fast Processor (Clock Speed)	Memory	Dimensions	Cost
		+	+	+	+	+	-
Detection Accuracy	+	↑↑	↑↑	↑↑	↑↑	↑↑	↓
Firing Accuracy	+	↑	↑	↑↑	↑↑	↑↑	↓
Low Power	+			↓	↓	↓	↑↑
Cost	-	↓↓	↓	↓↓	↑		↑↑
		≈45ft	≈51%	≥1GHz	≥512MB	3 x 3 x 3 ft	≤$1000

Table 2.6: Engineering-Market Trade-Off Matrix

3. Research

Section 3, Research, contains extensive study among similar projects as well as component comparisons in order to produce the robot for this project. It is broken down into different subsections addressing the areas of needs to make this project work.

3.1. Similar Projects

This section displays similar projects that have been found and assembled as research for the current system discussed in this paper. These projects showcase a self-targeting autonomous turret system, an autonomous sentry robot, and an autonomous chasing robot. All of these projects are past projects completed at the University of Central Florida. The projects are explained in further detail within this section including brief descriptions, components researched, and software approaches for development each project.

The first project, the self-targeting autonomous turret system (STATS), is very similar to the current proposed system. STATS was developed by UCF students Elso Caponi, Michale Lakus, Ali Marar, and Jonathan Thomas. STATS uses software in order to successfully attack moving targets. STATS is a camera based weapon system; therefore, it uses the camera sensor in order to detect, aim and fire. Targets were named to be hostile or friendly with by different coloring or different radio frequency indications, RFID.

The weapon of choice for STATS was an automatic airsoft gun. On the gun, a warning siren was implemented to signal possible incoming threats. The gun was able to move in both the X and Y plane via two servo motors. Individuals involved with STATS chose to have the turret controlled by a tablet. Students chose to have the tablet connect wirelessly to the turret. A live video feed was also sent to the tablet from the turret. Wi-fi modules were necessary in order to successfully send over this video feed.

Components used in implantation of STATS include a microprocessor, two servo motors, a camera, a PCB, a tablet, power supply, a microcontroller, a laser device, alarms and an airsoft gun. Most of these components have already been explained above. However, further detail will be described regarding some of these components. A laser device was implemented with the guns in order to accurately point and shoot at targets reading in data from a distance. The system was powered with both internal batteries as well as AC-DC electricity.

STATS team implemented concepts based off computer vision in their software build. The idea that the STATS team used involves algorithms that are used to determine the distance pixels have moved in between frames. These algorithms are initiated as the tracking method begins. Using all the different pixel locations, estimations on the targets next position is calculated. By estimating a target’s next location, the STATS team was able to fire accordingly taking into account the time it takes between a shot fired and the shot actually hitting the target.

Another similar project to the robot of the proposed system is the autonomous sentry robot. The autonomous sentry robot was developed by UCF students Brian Dodge, Nicholas Musco, and Trevor Roman.This robot has many similarities to that of STATS and the proposed system. There is heavy research within this project on the capabilities of SLAM as well as sensor research that includes Lidar, Kinect, and a standard webcam.

The purpose of this autonomous sentry robot is for security in enclosed buildings. More than one sensor is used in this project as well in order to navigate autonomously and build a map of the surrounding area effectively. Sensor data is processed by either a microprocessor or a laptop. Once the robot has built an efficient map of an area, the robot is able to detect if any changes occur to the original mapping. If so, the owner can be alerted through a mobile application of the change in environment. The owner can navigate the robot using the mobile application and will receive a live video feed straight to their smartphone.

The sentry robot’s design allows it to navigate fully autonomously. Only when the owner chooses to use the mobile application for navigation will the robot stop navigating on its own. The robot will even go back to its charging station when necessary in order to “refuel”. Objects that are unable to be viewed or seen by the camera was taken into account through development of this project. Many requirements such as low power, low maintenance, ease of use, and low latency were described accordingly in this project.

The hardware design resembles that of STATS and of the proposed system. The electrical components consist of a battery, charger, charging station, sensors, and a microcontroller. For software design, more information is given on SLAM and how it is proposed to work with autonomous navigation in their system. A logical breakdown for every autonomous feature is broken down and explained accordingly for further reference.

The last project that shows similarities to the proposed system is the autonomous chasing robot. The autonomous chasing robot was developed by UCF students Bryan Diaz, Victor Hernandez Salomon, Khanh Le, and Luis Sosa. The robot was designed to be able to follow any object in motion. As the robot detects an object in motion, it will prepare to get close as fast as possible to said object. Instead of pointing directly at a target, the robot was designed to be able to follow the same path of an object in motion.

Sensors as well as image processing are two main features to the functionality of this robot. A big factor that needed to be considered in their project implementation was the acceleration and deceleration of the robot depending on the position of the object in motion. The system also calls for an android application as a means of controlling the robot manually if preferred. One idea of implementation for this robot in an everyday setting was its use with police officers. Their idea was to have a police officer able to navigate the robot via their smartphone. If an item is defined as a “blacklist” item, in this case in terms of a license plate, it will pick up on these numbers and inform the officer accordingly.

All of these projects demand focus in the area of computer vision. The software implementation is one of the main features of success in every autonomous system. Hardware designs for all projects have many similarities while building an autonomous system within similar price ranges. Even though the proposed system will be used to fire at another battlebot in motion, the characteristics of each project remain the same with different objectives.

3.2. Field Mapping/Localization

Field Mapping and Localization may assist in the autonomous detecting, aiming, and firing of the proposed system.

3.2.1. SLAM

In order to satisfy the requirements of this project that the robot must have autonomous detecting, aiming, and firing, various algorithms were researched. One concept, SLAM, was chosen as a top contender for this project. In the following paragraphs, an introduction, history, consideration before implementation, explanation, and ideas of SLAM or SLAMMOT implementation will be illustrated. Finally, some robotic projects with implementation of both SLAM and SLAMMOT will be discussed to show possibilities that each should offer.

SLAM, which stands for Simultaneous Localization and Mapping, is a process that can be used to create a map of an environment and compute a current location on this map. SLAM consists of a wide range of algorithms that come together in order to solve the problem.

The idea is to be able to place a robot in any unidentified environment and allow the robot to build and constantly improve upon a map of its surroundings. The robot will then be able to navigate within its environment knowing its exact location which is shown on the map. SLAM has heavily impacted robot designs as it has the capability of creating autonomous robot systems. SLAM has been implemented in various different robots, ranging from a standard indoor robot to some robots that can navigate effectively underwater. An example of a SLAM map has been provided below in Figure 3.2.

SLAM was originally developed by Hugh Durrant-Whyte and John J. Leonard. Durrant-Whyte and Leonard aimed to solve the constant conceptual and computational mapping problem that was found within robotics. A crucial point that Durrant-Whyte and Leonard were trying to prove was that there is a connection between estimates of different landmark locations in a map. Once landmark correlation was focused on and the mapping and localization problem was looked upon as convergent, the SLAM problem was established. As more research and development came into play, individuals began using SLAM efficiently within various projects.

The SLAM concept consists of various parts; these parts include data association, state estimation, state update, landmark extraction, and landmark update. Some of these concepts, such as data association, will be explained in further detail within this document.

Before considering SLAM algorithms, it is important to note the different variables that must be considered for development. This includes the range detector or vision sensor as well as robot capabilities and specifications. Odometry performance must also be considered, which measures how accurately the robot can determine its location based on wheel rotation. An encoder can be used in order to capture the robot's location based on wheel rotation within this project. A robot cannot have more than a 2 cm per meter moved error as well as a 2 degree per 45 degree turned error.

Depending on whether a vision sensor or a rangefinder is implemented in order to develop a map of the robot's surroundings, various possibilities must be considered. Since two sensor modalities must be utilized in order to successfully detect, aim, and fire at targets, either sensor can be used in correspondence to the SLAM process while the other sensor may be used for more information in order to achieve further accuracy.

A rangefinder can provide little computation, efficiency, and precision. However, prices in regards to capabilities of each rangefinder must be considered. A vision sensor can provide more information than a rangefinder, but requires more computation. Also, changes in light may also affect the vision sensor. Since brightness conditions for the robot's environment have not been divulged for this project, the possibility of error using a vision sensor must be extensively considered.

Assuming the robot uses the LIDAR Lite v3 Laser Rangefinder (discussed in section 3.3.1.1) to implement the SLAM process, the position gathered by the encoder will be corrected by the information collected from the laser. In order to accomplish this feat, characteristics of the environment, often referred to as landmarks, must be extracted and reexamined when the robot is in motion. An EKF, Extended Kalman Filter, is responsible for updating a robots' position based on these landmarks. A simple outline regarding how the SLAM process works in accordance with the laser scan, landmark extraction, data association, and EKF is shown in Figure 3.1 below.

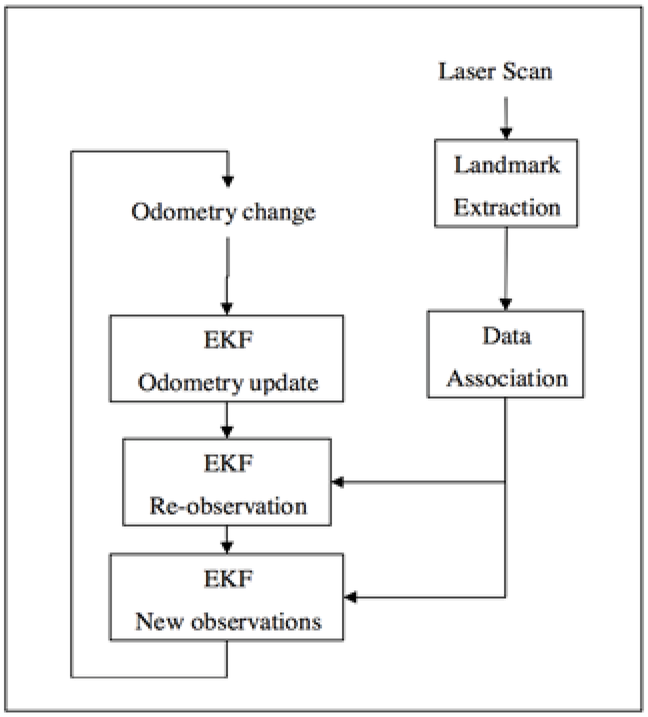


Figure 3.1: Outline of the SLAM Process
(Request Pending by SLAM for Dummies)

As the rangefinder and the encoder work in accordance with one another in order to find out the exact position of the robot, timing of both data enquiries must be considered. The laser data may become outdated if odometry data is achieved too late. Extrapolation of the odometry data, estimations of what the data will be, are used in the SLAM process to avoid this error of incorrect data timing.

Establishing which landmarks will be found by the robot in this project requires more research. Points to consider regarding landmarks includes being re-observable, distinguishable, plentiful, and stationary. For this project, using a rangefinder to detect boundaries will be very difficult, since boundaries were described as marked but not necessarily 3D. For instance, the boundary might be tape or another flat 2D object, which will give the rangefinder nothing to detect.

If boundaries are needed in order to effectively detect, aim, and fire, a vision sensor should be considered for project implementation. However, a rangefinder will be able to successfully pick up on stationary targets as well as the stationary obstacles placed within the field. Discounting objects in motion as landmarks, such as the robot or medic, is highly important to note, since the main target will always be in motion.

One algorithm that can be used regarding SLAM is the RANSAC algorithm. RANSAC, Random Sampling Consensus, may be used to extract lines from a rangefinder. Once the rangefinder scans multiple times, these reading are then compiled into a best fit line.

The algorithm then checks how many readings are actually close to the best fit line. Taking into account a certain threshold, one can successfully assume whether a landmark has been found accordingly. This threshold is called the consensus within the RANSAC algorithm.

Some things must be taken into consideration regarding the RANSAC algorithm; these include the amount of times the programmer would like to attempt to find lines, the amount of lines the programmer would like to use in order to create the best fit line, how many degrees to take into account for each reading, the maximum distance the reading will be able to go, and the number of points on each line for it to be considered. Without taking into account these scenarios, the RANSAC algorithm may result in failure.

Another algorithm that is used often in SLAM with RANSAC is the Spike algorithm. Extrema is used in the Spike algorithm in order to successfully locate landmarks. One must be cautious of the Spike algorithm, since it has a chance to fail in smooth environments. The algorithm itself relies on landscape change between two laser beams. This algorithm is great for finding things that lie in front of a wall; objects that are static, but are not actually the barrier of the room. In this case, the obstacles and static targets may be considered as “spikes’ within SLAM. A more intricate example of these spikes as well as the edges drawn on a SLAM map can be seen below in Figure 3.2 Figure 3.2 illustrates just one of the many possibilities that SLAM has to offer in terms of accurate mapping.

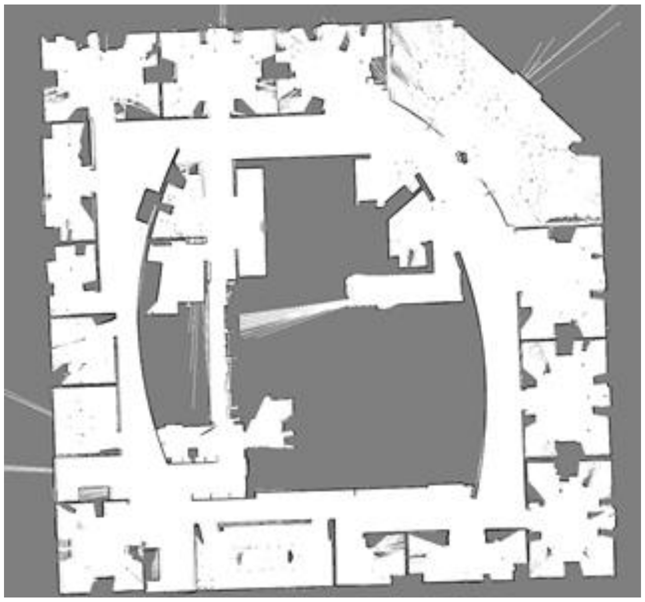


Figure 3.2: Intricate SLAM Map showing spikes and edges
(Request Pending by SLAM for Dummies)

3.2.2. SLAMMOT

If SLAM proves to be incapable of delivering the specified requirements due to the lack of being able to track moving targets, an alternative solution must be considered. Building onto SLAM in order to successfully track moving targets, SLAMMOT may be able to fill this role. SLAMMOT, Simultaneous Localization, Mapping and Moving Object Tracking, is very similar to SLAM, but encompasses moving object tracking, which will be necessary if the medic or robot will be marked on the map. It encompasses SLAM as described above with stationary objects and SLAM with detection and tracking of moving objects (DATMO). SLAM alone may perform poorly since the assumption of only static objects will be violated.

SLAMMOT may be used with either a vision sensor or a rangefinder, similar to that of SLAM. If a camera is used with SLAMMOT, appearance based approaches are widely used to detect moving objects. Either monocular SLAMMOT or stereo based SLAMMOT can be considered. However, if a laser scanner is used, feature based approaches can be used to detect moving objects alternatively. Opposite of SLAM, a camera may actually be more beneficial with the use of SLAMMOT, since objects in motion may be easier to track as opposed to using a laser scanner.

Assuming monocular SLAMMOT is used in this project in order to build a map of the environment as well as successfully track moving targets, data association can be achieved using either 2D image matching or 3D estimates of the filter. New features or landmarks are still put under the stationary hypothesis that would be used with local SLAM.

Using SLAM, up to 30 static features can be seen at at time and the map will update after 20 EKF updates. When a new landmark is found, two monocular SLAM would be initialized under two hypotheses. One is SLAM without adding a new landmark and one is SLAM assuming the landmark is stationary. Using both the negative inverse depth based method and the binary Bays filter-based method will be that basis of SLAMMOT.

Many projects that have been found using SLAM involve autonomous vehicles that can navigate an area. Outdoor projects and airborne projects have also used SLAM to properly navigate. One project found using SLAMMOT describes trying to incorporate SLAMMOT on a large scale in urban areas to lead towards fully autonomous vehicles in a human interactive environment. SLAMMOT has been attempted in different projects using either a monocular camera or a stereo camera. It has been found that it is possible to use SLAMMOT with a range sensor alone, however, data may be harder to track with objects in motion.

3.3. Sensors

As per project requirements, two sensor modalities were needed. Below is information on the sensors that were discussed.

3.3.1. Range Finders

While computer vision algorithms can be used to detect objects based off images, whether a picture or a frame from a video, they typically do not provide information on how far the object is from the source of vision. This type of information, target distance (the distance of an object from a source point), is typically done by what is known as a rangefinder.

It is necessary to determine how far a target is from the robot so that the robot can make a decision on whether or not it should fire at the target and how it should fire (i.e. is the target in bounds of the course, is it in the enemy zone, do the guns need to accommodate for distance by arcing, etc.). Therefore, one of the two sensor mechanisms of the battlebot system should be able to acquire the distance from the vision sensor.

Several range finders exist, but many are simply out of the budget or do not meet at least a 20 foot range of detection. Most of the range finders in the robotics market that are within the budget are made of laser or ultrasonic systems.

While researching range finders, of any kind, the first thing that was observed before progressing through any other specifications was the maximum range the sensor could reach. For the purposes of this robots targeting system, an ideal max detection range would be at least 40 feet in order to span the course from front to back. If the sensor specified that it could not detect an object at more than 40 feet the research of that sensor was immediately tossed.

3.3.1.1. Laser

Laser range finders are quite expensive, but their advantages warrant the cost. The laser range finders reach farther distances for target detection and are able to provide distance information at a more frequent and faster rate. This will allow the the robot to update the changes in position of an enemy target more rapidly and reduce delay with the firing mechanism. As an added bonus, some laser sensors provide a laser guide that may be beneficial for testing purposes. For example, the laser guide could be used to calibrate targeting so that the rangefinder is pointing at what it needs to be pointing to.

Some minor disadvantages of a laser range finder are its inability to detect the range of transparent objects as the laser will go right through the material. Another possibility is loss of the laser on the return trip by interference from other forms of light such as the sun when outdoors or by windows. However, both of these disadvantages are believed to be non-issues for this project as the field will be in an indoors environment and enemy targets are presumed to be non-transparent.

With a limited as demonstrated budget for the nerf battlebot system, it was only logical to search for a product that met the basic needs of range detection for a distance of about 40 feet and did not over do the job. Unfortunately, there are currently only a handful of laser range finder models on the market that met both budget and performance demands. The laser range finders found and researched are listed and compared in Table 3.1.

Name	**Price ($USD)**	**Max Range**	**Accuracy**	**Size**
TeraRanger Duo ToF Rangefinder with Sonar Sensor	$207.20	45.93 ft	+/- 2 cm	5.3 x 4.4 x 2.5 cm
LIDAR-Lite 3 Laser Rangefinder	$149.99	131.23 ft	+/- 2.5 cm	2 ✕ 4.8 x 4 cm
LeddarTech Leddar One Optical Rangefinder	$115.00	49.21 ft	+/- 5 cm	2” Diameter

Table 3.1: Comparison of laser range finders

Any three of these range finders listed in Table 3.1 appear to adequately do job solely based on the specifications listed, but the question is price versus performance. Is the slight increase in performance worth the moderate increase in cost considering the budget is very limited?

The specifications for the LIDAR-Lite 3 are very appealing in comparison to the TeraRanger Duo and LeddarTech Leddar One rangefinders. This is because its max range is almost triple that of the others and it is twice the accuracy of the LeddarTech One and only 0.5 cm less accurate than the more expensive TeraRanger Duo. As for size, it is much larger than the LeddarTech Leddar One, but slightly smaller, by volume, than the TeraRanger Duo. The TeraRanger Duo does offer a second form of range finding - a sonar sensor - but it is unlikely that both the laser and sonar functionality will be used in conjunction with each other or that the sonar will be used over the laser sensor.

A 0.5 cm difference in accuracy (LIDAR-Lite 3 versus the TeraRanger Duo) can make a world of difference however. These evaluations must be justified after reviewing other electrical components and also the mechanical construction aspect of the robot to see where there is any freedom in budget. At any rate, it is of best interest to invest a good portion of the budget on the automated detecting and firing system of the robot as that is a fundamental function.

3.3.1.2. Ultrasonic

While there are a plethora of ultrasonic sensors available on the market, almost all of them fail to reach a max range detection distance of over 20 feet (even with the higher end and more expensive models).

Additionally, ultrasonic sensors have other drawbacks than not having long distance range detection. During the transmission of an ultrasonic sound, sound interference is possible, whether that be absorption of the sound into low density materials or a similar sound being produced by another source resulting in a false reading. This is a cause for concern, because it is unknown if anything in the building hosting the playing field will create a sound emission that will confuse the sensor or if an obstacle is made up of low density material such as low density foam.

This combination of difficulties meant that ultrasonic sensors were out of the question for this project. These type of budget sensors are not designed for long distance range detection and are not very reliable for autonomous firing systems.

3.3.2. Camera and Vision Sensors

It is no secret that the camera plays an important role in the overall success of this project. Its job is to provide the processing unit with a stream of images that will be analyzed to detect, track, and aim at the targets. This is an enormous responsibility, and thus we had to study the impact of many of their features on our design and understand the tradeoffs at play. When considering cameras, the things that mattered most were the resolution range, frame rate, angle-of-view, cost, and compatibility with the other hardware and software in our system.

Resolution is instrumental to the accuracy of our system. The higher it is the more detail we have to determine if an object is a target. It makes clear sense to maximize this, but we also have to keep in mind that the higher the resolution, the larger the processing overhead would be. The group agreed that a High Definition resolution of 720p (1280 x 720) would suffice, but we searched for options with higher resolutions, that could be downgraded if proved to be too much.

Frame rate was very important since our system needs to work in real time. Any latency between receiving frames and their processing could be the difference between hitting and missing the target. Thus the group decided that a frame rate of around 30 frames per second would be needed.

Angle-of-view is also important since our strategy involves having as much of the playing field in view as possible. The bigger the angle-of-view, the more of the physical scene we capture at a given distance.

The Raspberry Pi Camera Module v2, Pixy CMUcam5, and the Logitech C920 are among the most popular choices today in robotics projects and would be well suited for ours. The comparison of these vision sensors are seen below in Table 3.2.

3.3.2.1. Raspberry Pi Camera Module v2

The Raspberry Pi Camera Module v2 is a great option for our system. First, because it is a high-definition camera, capturing a picture at resolutions up to 3240 x 2464 at 15 frames per second. 15 frames per second is less than we require, but the resolution can be lowered to achieve higher frame rates. For example, it can record a resolution of 1080p at 30 frames per second, and it can do 720p at 60 frames per second. Also it has a decent angle-of-view of 62.2 degrees horizontal x 48.8 degrees vertical. A huge plus for this camera is that it is supported by a large community of Raspberry Pi users, which will be handy if we encounter any trouble interfacing with it. Although it is compatible with all Raspberry Pi models, it cannot interface with many other boards directly since it uses CSI (Camera Serial Interface) as its output, a specialized interface for the Raspberry Pi. Therefore, making this camera a great choice only if we have a Pi to pair it with.

3.3.2.2. Pixy CMUcam5 Image Sensor

Another camera that we have looked at is the Pixy CMUcam5 Image Sensor. It has a native image resolution of 1280x800 and records this at 25 frames per second. It has the largest angle-of-view of the cameras we considered, getting 75 degrees horizontally and 47 degrees vertically. It is highly compatible with microcontrollers, as it has data outputs of UART serial, SPI, I2C, USB, digital, and analog. What makes the Pixy unique is that it has its own processor and it is pre-programmed to do object detection. By the push of a button, it can learn to detect objects of a certain color signature, and shape. Another plus for the Pixy is that it is compatible with programming languages C, C++, and Python.

3.3.2.3. Logitech HD Pro Webcam C920

The Logitech HD Pro Webcam C920 is another great option we found. It has been a popular choice in vision projects, and it is easy to see why. It records in full high-definition 1080p, at 30 frames per second. It has a large diagonal angle-of-view of 78 degrees, which at an aspect-ratio of 16:9 translates to 70.42 degrees horizontal and

43.30 degrees vertical. It is a universal plug-and-play device which connects via USB 2.0. Another advantage of using this camera is it has H.264 video coding. By accessing this format of video stream the video quality will remain high, but compressed to low bit rates for quicker transmission.

Make	**Angle of View**	**Resolution**	**Frame Rate**	**Price ($USD)**
Raspberry Pi Camera Module v2	62 x 48	3240 x 2464	15	$25
Pixy CMUcam5 Image Sensor	75 x 47	1280 x 800	25	$68
Logitech HD Pro Webcam C920	70 x 43	1920 x 1080	30	$58

Table 3.2: Comparison of vision sensors

3.3.3. Thermal Camera

The main target we seek to detect in the battles will be our opponents robots. One of the reasons why it will be a difficult task to do this is that there will be other objects on the course to distinguish them from. A key difference between the robots and those other objects though are that they will be composed of electrical devices. These electrical devices will radiate infrared light or produce thermal energy that would otherwise go undetected by the human eye or any visible light camera. Infrared or thermal imaging camera sensors could make use of this detail and provide an advantage to the detection algorithm.

The problem with using thermal cameras in our design are that there are few options available to us due to their expensive cost. The thermal cameras that we could afford are compared in Table 3.3.

Sensor	FLiR Dev Kit	Seek Compact
Resolution	80 x 60	206 x 156
Field of View	51 ° (Horizontal); 63.5 ° (Diagonal)	36 ° (Horizontal)
Wavelength Band	8 to 14 microns	7.5 to 14 microns
Temperature Range	-4°F to 248°F	40° F to 626° F

Connectivity	I2C (SPI)	Micro USB
Platform Compatibility	Raspberry Pi, Arduino, ARM	Android
Price	$259.95	$250

Table 3.3: Comparison between FliR Dev Kit and Seek Compact thermal sensors

After analyzing the specifications, there were different tradeoffs we would make in selecting one the similarly priced thermal sensors. The Seek Compact offers a higher resolution than the FLiR, but at a smaller field of view. A higher resolution would give the software more pixels to detect an object, which would be crucial at longer distances. A wide field of view would be preferable since our idea is to align the thermal sensor with the view of our visible-light camera. The temperature range of the Seek Compact is larger than the FLiR's, but its floor detection temperature of 40° F could be too high to be useful. A disadvantage of the Seek Compact is that it's built to operate on the Android operating system which would limit our platform choices.

3.3.4. Encoder

A dilemma in having a specified course size and designated zone is preventing the robot from overstepping its boundary. Constantly capturing where the robot's position is within its boundary can be used to prevent user fault (e.g. the user remote controlling the robot goes in the wrong direction and leaves its zone resulting in a deduction of points). Limitations could be set to lock a motion that would cause the robot to leave its zone if it is at close quarters with the border.

One idea of keeping tracking of the robot's relative position on the field was to check when the remote control was active and using trial and error to approximate how far the robot has traveled in relevance to the direction the user has input and the size of the circumference of the wheels. This would take a great deal of time and most likely end with very inaccurate results.

But as it turns out, technology already exists to simplify this method of tracking - encoders.

An encoder is a sensor that can be attached to the wheels of the robot to track the rotation of the wheels. It converts an electronic signal into a digital signal that can then be sent back to software algorithms to maintain a relative position of the robot on the course. Given the size of the wheels it is possible to determine how far the robot has traveled and even how fast it is going.

There are several different types of encoder systems, all with varying amounts of accuracy and pricing.

For project cost savings, in order to stay within our budget limitations, it was recognized that there exists package deals of gear motors that each include encoders. A majority of the gear motors on the market tended to robotics include encoders that use the hall effect; rotational changes are tracked by a change in voltage invoked by a change in magnetic field by the magnets attached in the encoder system.

While these budget encoders are not very accurate for robotic systems that rotate constantly at high speeds (e.g. an autonomous label attaching system for bottles of soda), it is estimated that they will work fine for the purposes of an autonomous robot constrained to roughly a 300 square foot course zone. This is because it is predicted that the robot will not be moving all that much to reduce strain on the firing mechanism and processing unit. The more the robot moves the more inaccurate the tracking system could become. If rotational precision accuracy is off by even a millimeter than the more the wheel rotates the more inaccurate data adds up. To clarify, if one millimeter of distance is lost per rotation than after 150 wheel rotations it is entirely possible to have lost about half a foot of distance in positioning data.

As long as manual remote controlled movement of the robot is limited, low-budget encoders will suffice for the recording of its position relative to its restricted zone boundaries.

3.4. Microcontrollers

The microcontroller will be used to connect the electrical subsystems together. It will read from the sensors and send their data to the main computer for processing. As targets are detected, the main computer will supply the microcontroller with their coordinates. These coordinates will be translated into the movements of the stepper motors to aim the weapons, or the servo motor to keep the tracked object in the view of the camera. It will also be in charge of controlling the gear motors that will drive the robot.

Due to the immense popularity and support available for the Arduino boards, we have focused our search on microcontrollers available with this platform. A list of potential microcontrollers can be seen in below Table 3.4. This will provide us with a lot of resources and software libraries to speed up our development.

Arduino Software is the integrated development environment provided by Arduino that we will use to program the microcontroller. It runs on the most popular operating systems such as Microsoft Windows, Mac OSX, and Linux. It is based on the Processing programming language which builds on Java, but it also supports the languages C and C++. This was important to us as the group is familiar with them, and have experience programming embedded systems.

Development Board	Arduino Due	Arduino MEGA 2560	Arduino UNO
Microcontroller	AT91SAM3X8E	ATmega2560	ATmega328P
Architecture	32-bit ARM	8-bit AVR	8-bit AVR
Clock Frequency	84 MHz	16 MHz	16 MHz
Max Operating Voltage	3.3	5	5
Flash Memory	512 KB	256 KB	32 KB
SRAM	96 KB	8 KB	2 KB
GPIO pin count	54	54	14
Price	$40.99	$36.99	$22.39

Table 3.4: Comparison of microcontrollers

Arduino Due is the most powerful microcontroller of the three. It has the fastest clock, most flash memory, most SRAM, and GPIO pins. It can dish out more power than other two microcontrollers as well. It can output both 3.3V and 5V at 800mA. The Arduino MEGA 2560 and the Arduino Uno cannot match this. All the Arduinos come preloaded with bootloaders that makes them ready to download new code.

One important detail that cannot be seen in the specifications of the microcontrollers are their popularity. The Arduino Uno is the most widely used Arduino, and therefore there are the most documented.

3.5. Microprocessors

The main objective of our robot is to autonomously detect targets. The team plans on accomplishing this by applying several computer vision techniques such as object detection, facial recognition, and motion tracking, which are known to be computationally expensive. Even a system dedicated to solving one of those tasks, requires a high amount of power to perform.

From researching similar projects, we realized we would need a device more powerful than a microcontroller to do this amount of processing. In fact, the most common setups of these systems were on desktop computers with multi-core processors, Gigabytes of RAM, with 64-bit Operating Systems. These kinds of conditions are needed because these algorithms are fighting to work in real-time. The computer we chose would need to be of a similar mold as those desktops.

In order for it to be useful, to us it would also need to run an operating system compatible with OpenCV to call the vision algorithms. That leaves us with machines that either run Linux, Windows, or Macintosh. Since we are looking for single-board computers we are mostly left with a system running on Linux.

The power requirements of the machine are also relevant. Each battle will last about 10 minutes, and many components will be drawing power from the battery. Most of the single-board computers we found ran on 5V and 2A.

Below, in Tables 3.5-3.8, are the different options we found:

Name	Banana Pi Pro Computer Board BPI-M1
Price	$37.50
OS	Raspbian, Android, ArchLinux
CPU	ARM cortex-A7 dual-core CPU @ 1.0GHz
GPU	ARM Mali400MP2
RAM	SDRAM: 1GB DDR3 (shared with GPU)
Storage	4GB 8-bit eMMC on-board flash storage
USB	2 x USB 2.0 ports
Power	5V, 2A
Size	Size: (L x W x H): 92.00mm x 60.00mm x 0.00 mm
Weight	48g

Table 3.5: Banana Pi Pro specifications

Name	ODROID-C2
Price	$40
OS	Ubuntu, Android, ARCHLinux, Debian
CPU	Amlogic ARM® Cortex®-A53(ARMv8) 1.5Ghz Quad Core CPU
GPU	Mali™-450 GPU (3 Pixel-processors + 2 Vertex shader processors)
RAM	2Gbyte DDR3 SDRAM
Storage	eMMC5.0 HS400 Flash Storage slot, UHS-1 SDR50 MicroSD Card slot
USB	4 x USB 2.0 ports
Power	5V, 2A
Size	85 x 56 mm (3.35 x 2.2 inch)

Table 3.6: ODROID-C2 specifications

Name	Raspberry Pi 3 Model B
Price	$39.95
OS	Raspbian
CPU	Broadcom BCM2837 64Bit ARM Cortex-A53 Quad Core 1.2GHz
GPU	Broadcom VideoCore IV
RAM	1GB LPDDR2 (900 MHz)
Storage	MicroSD
USB	4x USB 2.0 Ports
Power	2.5A @ 5V
Size	85.60 mm × 56.5 mm (3.370 in × 2.224 in)
Weight	45g

Table 3.7: Raspberry Pi 3 Model B specifications

Name	NVIDIA Jetson TK1 Development Kit
Price	$192.99
OS	Linux for Tegra
CPU	NVIDIA 4-Plus-1 Quad Core ARM Cortex A15 @ 2.3GHz
GPU	NVIDIA Kepler GPU with 192 CUDA cores
RAM	2GB 64bit
Storage	16GB 4.51 eMMC
USB	1 x USB 3.0 port, 1 x USB 2.0 micro-ab socket
Power	4.8A @ 12V
Size	133mm x 133mm x 30mm (5.2in x 5.2in x 1.18in)
Weight	120g

Table 3.8: NVIDIA Jetson TK1 specifications

After collecting a group of options at single-board computer we were able to compare them over a few metrics. Computational power was a top priority for our robot so we started there. In terms of CPU clock speed, the boards were very competitive. The Nvidia Jetson TK1 had the fastest CPU by a wide margin, with its Nvidia ARM Cortex A15 quad-core clocking in at 2.3GHz. The next best CPU was found on the ODROID C2 with its ARM Cortex A53 quad core running at a solid 1.5GHz. The Raspberry Pi 3 and Banana Pi CPU's were slightly slower. The NVIDIA Jetson also provided additional computational power with its GPU containing 192 CUDA cores. Our system could absolutely be benefitted by this type of GPU as there are OpenCV libraries optimized for CUDA. Both the Odroid C2 and the Nvidia Jetson had the most RAM, with 2GB of it on each.

After examining the options based on their hardware specifications we looked to gain more insights to their performances. Here we will show difference benchmarks we found performed by Michael Larabel, the founder of a company named Photonix Media, which analyzes and tests the performance of Linux hardware.

The C-Ray v1.1 and Smallpt V1.0 are good tests to predict image processing speeds because they respectively test floating point calculation and image rendering. The results of the test are seen in Figure 3.3 and Figure 3.5. The Raspberry Pi 3 beat out its entire competition in both test, except for the Nvidia Jetsons. John The Ripper v1.8.0, Figure 3.4, is a password cracking test, and the Raspberry Pi 3 beat out all its competition minus the Jetson TX1. Then the last figure, Figure 3.6, a general performance per dollar metric which showed the Raspberry Pi 3 beating out all the competition.

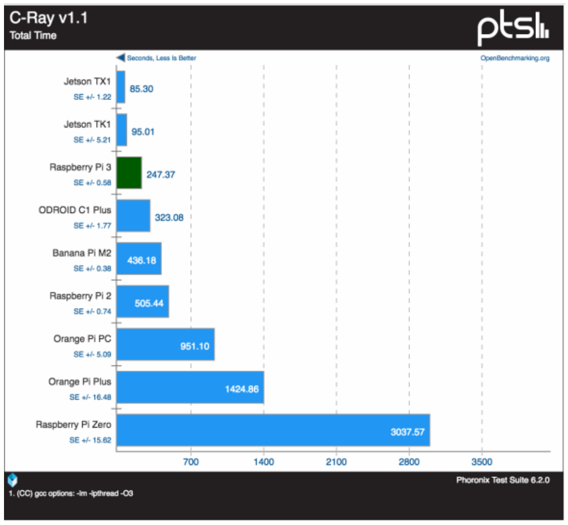


Figure 3.3: C-Ray Benchmark (Permission Granted by Photonix)

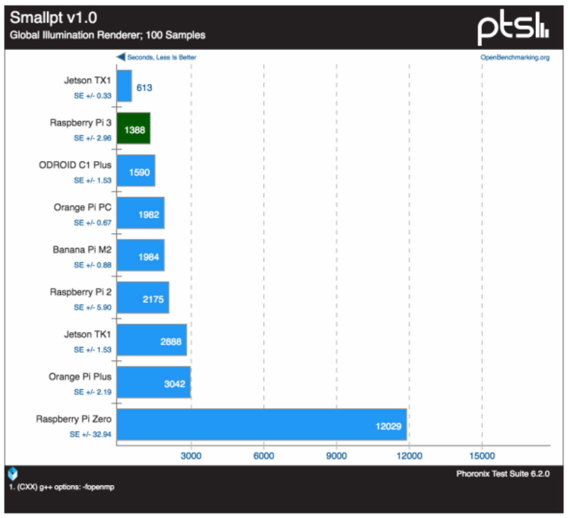


Figure 3.5: Smallpt v1.0 Benchmark (Permission Granted by Photonix)

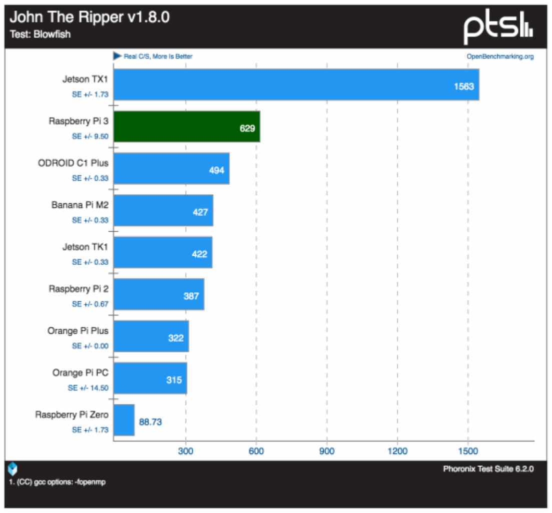


Figure 3.4: John The Ripper Benchmark (Permission Granted by Photonix)

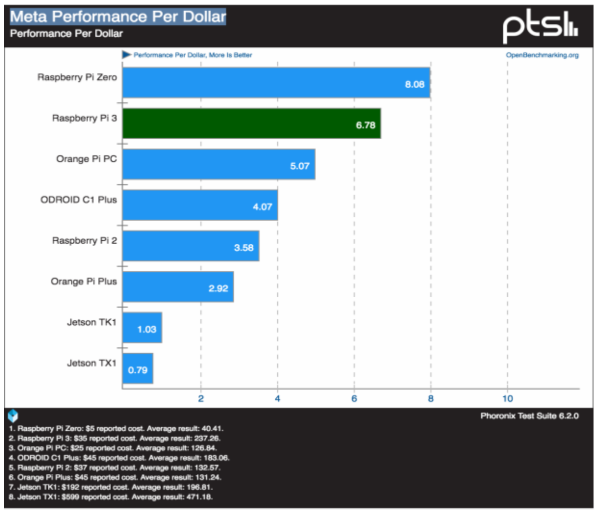


Figure 3.6: Performance Per Dollar (Permission Granted by Photonix)

3.6. Memory

Images, when converted into a readable format by memory (bits and bytes), take up a large amount of memory, or RAM (Random-access memory). The problem is that there are a lot of different image resolutions (640x480, 1280x720, etc.) and image formats available (JPEG, PNG, TIFF, RAW, etc.) and each resolution and format varies in the amount of data they take up. There are a number of different combinations to use and it is unknown at this time which settings will be applied.

Take, for example, an image that is 128x128 pixels. A general rule of thumb for estimating the total memory consumption of an image is to multiply the pixel width and height of the image by four bytes to get the number of bytes. For this image, the memory consumption would be upwards of 0.06 megabytes after conversion from bytes. This image is very small compared to what is expected for this project and it must also be taken into consideration that this is just one image. It is highly improbable that a computer vision algorithm would be able to accurately recognize an object that is more than 20 feet away from an image that is 128x128 pixels. Figure 3.7 visually demonstrates just how big a 128x128 pixel image really is.

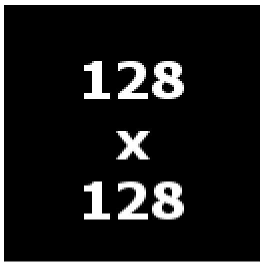


Figure 3.7: An example of a 128x128 pixel image
(Team Designed)

Assume now, that the quality of the camera that will be used for the detection of objects farther than 20 feet will stand at roughly eight megapixels - which is about eight million pixels. The memory consumption for an image of this size starts at about 30 megabytes. If the camera is recording at a rate of 30 frames per second than it can be expected that at least 900 megabytes (30 frames x 30 megabytes/frame) of memory will be necessary for memory transfer (depending on how the frames are transferred - in this example all 30 frames are assumed to be transferring at once).

Accounting for all other processes a device must use memory for (e.g. the operating system, system services, etc.), it would be of best interest to choose a device that has 1 gigabyte or more of memory. The more memory in the system the more room for early development testing before optimization and avoidance of system crashes from possible memory exhaustion. Any less than 1 gigabyte of memory and it is unclear how stable the system will remain at the example rate above. Quality of the images produced by the camera will have to be sacrificed in order to fit the limitations set by a memory amount less than 1 gigabyte.

3.7. Operating Systems

The operating system chosen to run on the main processing unit is Raspbian Jessie. Raspbian is a Debian-based distribution of Linux. It is the officially supported operating system of the Raspberry Pi, therefore it will help use its computing resources to their potential. It comes with Python 2 and Python 3 preinstalled, therefore there will not be any problems porting our code to the Raspberry Pi. Afterwards, it is a very straight forward process to install OpenCV Python bindings. Another reason why Raspbian Jessie is right for the Raspberry Pi is that the Arduino IDE is available for it. This is very important as without it there would not be a way of the Raspberry Pi and Arduino microcontroller to communicate with each other.

ROS (Robot Operating System) is a software platform designed to make robotics software more manageable. It is not a desktop Operating System but it runs on one. It contains libraries of code commonly used by robots such as to interface with hardware, and software like OpenCV or Point Cloud Library.

ROS simply makes the integration of a system of components easier to handle. It uses a publish/subscribe model to enable message passing through what are known as nodes. Nodes can be a computer, or any electrical device end point.

ROS is compatible with Linux, Mac OSX, and Windows. It can be run on the Raspberry Pi, and can integrate it with the Arduino.

After design choices we decided to use the nvidia Jetson TK1. The Jetson TK1 on this battlebot is compiled with an NVIDIA package called Jetpack 2.3.1 which contains an optimized linux operating system named L4T - Linux for Tegra and several other software packages essential to computer vision such as OpenCV, CUDA, and VisionWorks.

OpenCV and VisionWorks contain library packages that were imported and reconstructed upon for computer vision algorithms. CUDA is the platform that NVIDIA sources to its users to enable the usage of the onboard GPUs and work in conjunction with OpenCV and VisionWorks.

This robot runs L4T version R21.5 and was recompiled using kernel source code provided by NVIDIA to customize the kernel and allow for Bluetooth serial communication and a wireless connection using an AC-7260HMW mini-PCIe wifi card and coupled antennas.

Scripting was created to configure the Jetson on each boot to maximize usage of the onboard CPU’s and GPU’s in order to reduce workload and unlock the full performance potential that NVIDIA supplies.

The operating system is very sensitive as most of the setup is done via terminal command line, leaving plenty of room for malfunction if configured incorrectly.

3.8. Motors

This section of research focuses on the motors that will be used for this project. In order for our robot to able to move and transport itself around a field of interest, motors will be required. There are several motors needed in order to make the robot functional. The robot will need DC motors to transport itself in any direction by means of manual navigation. The robot will also need a stepper motor to turn the turret in order to properly aim the Nerf-blaster at a target location. Finally, the robot will require a servo motor to position the camera to visualize the field of interest. In the upcoming sections will discuss these different types of motors and provide background and product comparisons that will be served for this project.

3.8.1. Brushed DC Motor

The most important factor in choosing the right motor is knowing the weight load of the robot and the diameter of the wheels. A brushed DC motor is designed to be able to convert a direct current power source into a mechanical energy. Since brushed DC motors can supply up to four times its torque value without stalling, it would be the ideal choice for driving the robot.

3.8.1.1. CIM Motor

CIM motors will be used in a set of four to drive the wheels of the robot. In Table 3.9 shows a comparison among different DC brushed motors and their specifications.

Name	**Voltage(V)**	**No load RPM**	**Free Current (A)**	**Max Power (W)**	**Stall Torque (N-cm)**	**Stall Current(A)**
2.5” CIM Motor	12	5310	2.7	337	2.42	133
9015 Motor	12	16000	1.2	179	428	63.8
Banebot RS- 540 Motor	4.5-12	16800	1	-	278.8	42
NeveRest 40 Gearmotor	12	6600	1.2	138	396	11.5

Table 3.9: Comparison of CIM motors

3.8.2. Servo Motor

The camera that will act as the eyes for the robot will need to be able to stay fixed on the targets it is detecting while the robot is in drive mode. A servo motor will be used to allow the camera to rotate independently as the robot moves. These motors are the best fit for keeping a camera in a fixed location because they provide precise angular or linear position. It uses a closed-loop servomechanism that can provide position feedback to its initial and final position. The only cons to servo motors is they tend to draw a lot of current so choosing the right battery power will have to be taken in consideration. Table 3.10 contains comparisons to servo motors that will be considered for the project.

Name	Size (mm)	Speed	Torque	Connection
Futaba S3004 Standard Servo Moto	41x30x36	0.19 sec/60o at 6V	4.1 Kg-cm at 6V	Standard J-type connector
HS-422 Servo Motor	41x20x37	0.16 sec/60o	4.1/5	Standard J-type connector

Table 3.10: Comparison of Servo motors

3.8.3. Brushless DC Motor

Lastly, a brushless DC motor will be needed to turn the Nerf-blaster to target location. Brushless motors are highly considered since positioning the Nerf-blaster will need an accurate location and are normally controlled by a computer processing. They can also rotate in an equal number of steps. Since our Nerf-blasters will not exceed more than 5 pounds, a stepper motor with minimum standard torque can be used. A comparison of Brushless DC Motors is seen below in Table 3.11.

Name	Size	Voltage/Current	Holding Torque	Step Angle
ROB-09238	48x42mm	12V/0.33A	2.3kg*cm	1.8
RB-Spa-983	48x42mm	3V/1.7A	48N*cm	0.9

Table 3.11: Comparison of Brushless DC Motors

3.9. Autonomous Detection

The central task of this project is for the robot to autonomously detect and attack its targets. This will be a challenge because there will be a lot of unknowns it will have to overcome. The robot will be placed in a course we have never seen before. It will need to shoot at the opponent’s robot which we will not have a visual description of beforehand. And there will be course obstacles in the way which the robot will not be allowed past, and thus will hide and occlude the targets. Those are some of the unknowns that will pose a challenge and it will also need to attack stationary course targets, which are going to be large pictures of faces mounted on easels about two feet high.

Due to the complexity of these problems the team has decided to use a set of computer vision techniques to solve them. Computer vision is an interdisciplinary field that is concerned with the automatic extraction, analysis, and understanding of useful information from digital images and videos. Computer vision is a field closely related to Artificial Intelligence and has applications in agriculture, autonomous vehicles, forensics, robotics, security and surveillance, and the list goes on. Common subdomains of computer vision include scene reconstruction, event detection, object recognition, motion tracking, and facial detection.

3.9.1. Computer Vision Software Frameworks

It has been decided that computer vision would be our robot’s main means of automated target detection, but we do not intend to reinvent the wheel as developing a computer vision algorithm could be its own entire project. Therefore, our next job is to select a vision software framework with as many functions we need implemented. In this section we will look at the different options that we came across.

3.9.1.1. OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions used for both computer vision and machine learning. OpenCV was officially launched in 1999, as a research initiative by Intel Corporation. It contains more than 2500 optimized algorithms used for facial detection, identifying objects, tracking camera movements, stereo vision, and more. OpenCV has a large community with thousands of users, and downloads of it in the millions. It is written in C and C++, but it also has programming interfaces available for Java and Python. It supports several platforms such as Windows, Linux, Android, and Macintosh. It is BSD-licensed product which makes the code available for businesses and organizations to use and modify at will.

3.9.1.2. LTI-Lib

LTI lib is an object oriented library in C++ with algorithms and data structures commonly used in computer vision. It has been developed by the department of Computer Science at Aachen University in Germany, as a part of many research projects with robotics, object recognition, and gesture recognition. It was built using GCC under Linux, and Visual C++ under Windows NT. It contains over 300 classes dealing with linear algebra, classification, visualization tools, and image processing. The system requirements to use LTI-Lib are Windows NT with MS Visual C++ .NET 2003, or a Linux distribution with GCC 3.1 or later. LTI-Lib is an open source software available under the terms of the BSD License.

3.9.1.3. MatLab

Matlab is a multi-paradigm numerical computing environment and programming language developed by MathWorks. It is a family of products with toolboxes ready available for many applications such as control systems, physics modeling, image processing and computer vision. The image processing and computer vision packages provide functions such as feature detection, object detection, motion estimation, and 3D point cloud processing. Matlab is known to be easy to use with its own programming language which was written in C, C++, and Java. It supports Windows, Linux, and Macintosh. Matlab is not free, an individual license for the Computer Vision System toolbox costs $1,350.

3.9.1.4. VXL

VXL (the Vision-something-Libraries) is a multi-platform collection of C++ software libraries for Computer Vision and Image Understanding. It was created by extracting the core functionalities of two large systems: the Image Understanding Environment (IUE) and Target Junior (TargetJr) with the purpose of making a light, fast and consistent system. VXL is written in ANSI/ISO C++ and is designed to be portable over many platforms such as Windows, Linux, and Macintosh OS. As well as the core libraries, there are libraries covering numerical algorithms, image processing, coordinate systems, camera geometry, stereo, video manipulation, structure recovery from motion, probability modelling, classification, feature tracking, topology, structure manipulation, 3D imaging, and more.

3.9.1.5. Framework Comparison

After researching vision software framework for our project there we had to consider before making a decision. Right off the bat we had to eliminate Matlab from contention because of its licensing cost. It is a shame we could not use it, since it is indeed state of the art technology with loads of functionality and is raved by professionals in the industry.

We were pleased to find that the other options were available in C and C++, as that would facilitate the portability of the software between our development environment and robotic platform.

An important factor we needed to investigate was the performance of these vision libraries. We came across a benchmark produced in the book titled “Learning OpenCV” published by O'Reilly, seen in Figure 3.8. It compares OpenCV to LTI, VXL, and OpenCV with IPP using four different performance benchmarks. IPP or Intel IPP is a package of optimizations to functions of OpenCV running on a set of Intel x86 and x64 platforms with Intel® Integrated Performance Primitives.

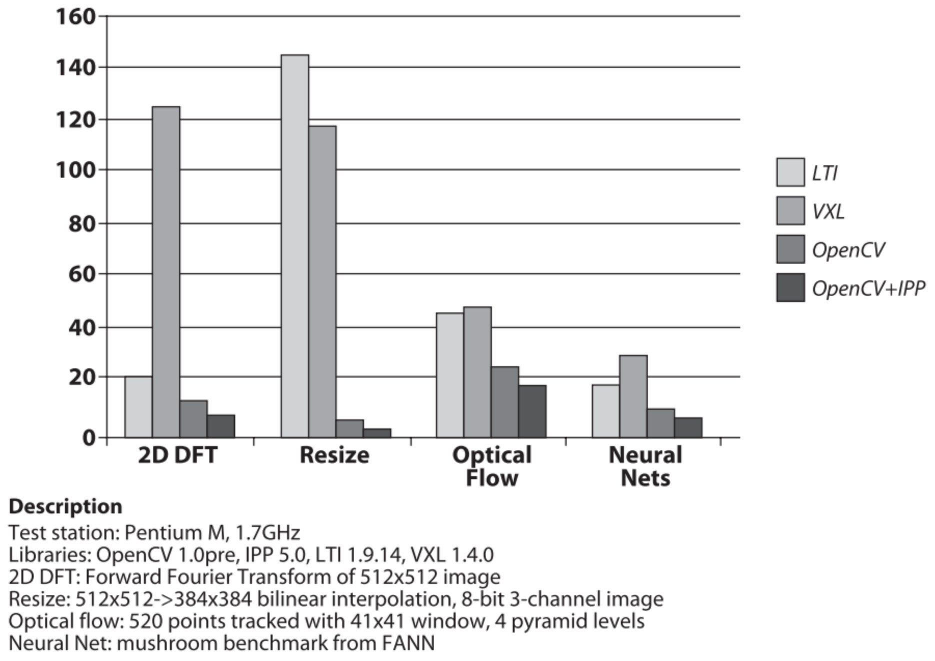


Figure 3.8: Benchmark of different vision libraries
(Permission Requested from O'Reily)

As seen in Figure 3.8, in all the tests performed, OpenCV outperformed both VXL and LTI. The biggest differences in performance were seen in the image resize tests where OpenCV was close to 24 times faster than the next runner up, VXL. The optical flow test is also important to us making our decision because it relates to motion tracking, and there the results were a lot closer. Still OpenCV was almost twice as fast as the next runner up, LTI.

One of the most important factors when choosing a software framework is its documentation. All the frameworks provided comprehensive documentation on their websites, but we found OpenCV's to be the most elaborative. Their documentation includes class definitions as well as examples of programs showing how to use the most common procedures.

All things considered, OpenCV will be our choice of vision framework. It offers the largest collection of optimized vision algorithms in one package. It is compatible with the on board processing units we considered. It is programmable in the Python language which our team prefers due to its ease of learning and in-built data structures and functions. We have read parts of the documentation available for OpenCV, and found it very useful. Tutorials are available on OpenCV's website, in books by different publishers, and videos online that will make the development process less troubling in case we encounter any problems.

3.9.2. Object Recognition

A major area of object recognition are what are called feature-based detection techniques. There is no exact definition of what constitutes a feature, but they are “interesting” parts of images used to describe their contents. There are many types of features in computer vision algorithms, and their practicality depends on the application. Generally, though, good features can be consistently found over different images of the same scene, are robust to transformations such as rotation and translation, are insensitive to noise, and are salient. Due to the complexity of the battlebot contests it is crucial to combine the detection of different types of features into our target detection pipeline. Here I will outline different feature based algorithms.

3.9.2.1. Edge Detection

Edge detection algorithms use a set of mathematical methods to find points in images where there are sharp changes in the intensity of a neighborhood of pixels, called edges. Edges are important features as they typically occur on the boundary between two different regions in an image. This helps determine the shape of objects in a frame, and also highlights a region worth looking at.

One of the most prominent edge detectors is the Canny algorithm. It uses a 2D Gaussian filter to smooth and blur the image which results in accurate edge detection. Another advantage is that it is adaptive to the distribution of intensity values present in image with hysteresis thresholding. Other edge detection algorithms use one threshold to decide whether a pixel makes a strong edge candidate. If a possible edge has a value equal to or greater than the value preset by this threshold, then it is set as an edge. Hysteresis thresholding uses two thresholds, one high and one low. If an edge is above the high threshold it is passed on as an edge. But if an edge is higher than the low threshold and adjacent to a high threshold then it is also passed as edge. This technique helps find edges that are not very visible but correspond to the outlining contour of an object.

3.9.2.2. Hough Transform

Hough transform is a technique that builds upon the results of an edge detector to help recognize shapes. Edge detection algorithms will find a series of edges which can then be linked together to form contours. If there are occlusions of the objects, then the edges will not be linked. Hough transform is used to detect lines and circles using these individual edge points.

Hough transform begins by modeling the lines that could pass through an edge. Doing this for all edges, it will then find the points where these lines intersect. Using a voting technique that counts the intersections and a preset threshold, it will determine where the most likely lines are found. A similar procedure can be done to detect circles in edge maps.

This procedure would provide a more revealing set of characteristics of the objects encountered. This can help in the detection of the opposing robot’s structure, the boundaries of the course which will be clearly marked lines, the shape of the obstacles in the keepout zone, and the mounting of the stationary course targets. It is also important to note that it is a procedure that can be done real-time, and concurrently with other processes.

3.9.2.3. Blob Detection

Another class of feature-based detection we can make use of is blob detection. A blob detection algorithm searches images for regions of a similar color or property. We will not know what the opponent's robot will look like, but by detecting blobs we will find areas of high contrast between objects and a background and objects.

Blob detectors can be simple to program. OpenCV provides function call SimpleBlobDetector to search for different types of blobs. Blobs can be searched by color, or by area coverage in pixels. Blobs can also be searched by shape. This blob detector can find concave or convex shaped blobs. It can also find lines, ellipses, and circles.

This could be a very easy way of detecting our targets, but it will definitely take a lot of tweaking to provide reliable results.

3.9.3. Motion Detection

Potentially one of the biggest considerations for this project is the idea of using motion detection in order to autonomously track, aim, and fire at an enemy robot or medic. Utilizing computer vision techniques, motion detection will allow for tracking of moving objects through the use of two sensor modalities chosen for this project, a camera and a rangefinder. Since the only objects that will be in motion within the frame will be targets, it is safe to assume that every moving target can be shot at accordingly.

Although motion detection provides key advantages to the possibilities of solving the proposed system’s problem, environment conditions as well as the fact that the robot itself will be in motion are all things to consider before choosing motion detection as a solution to this project. Since the field will be in an indoor environment, it is likely that the objects within the room will be clearly distinguishable from the rest of the environment. Knowing that the team robot will be mobile presents a more challenging problem to effectively track enemy moving targets. This creates two independent motions that must be considered; these motions are named the ego-motion of the team robot and the external motion of the targets. This processing of both motion algorithms must be done in real time in order to effectively shoot at the desired target.

Source [32] provides a detailed explanation as to how the ego-motion algorithm can be used in accordance with a mobile robot in order to detect moving objects. As a summary, ego-motion is considered a coordinate conversion procedure that computes a transformation. It is important to eliminate the ego-motion from the image in order to take into account the real position of an object in motion. This allows for more accurate motion detection and prevents errors of stationary objects being labeled as moving objects. If motion detection is a plausible solution that is chosen, it will be imperative to consider ego-motion, as the designed robot shall be in motion throughout the length of the competition.

Another thing to consider when using motion detection for real time tracking is that noise can be present during processing. Noise handling algorithms can be a solution to prevent unnecessary intervention and miscalculation with the motion detection algorithm. Ideally, the noise handling algorithm can be used as a secondary processing method. For example, with the high possibility that a monocular camera will be used in this project, there is a chance some objects may have noise on certain boundaries since there is a lack of depth. One way to solve this problem, as discussed in source [32], is to estimate the position and velocity of the moving object.

The rangefinder utilized in this project will provide the system with further information in order to implement motion detection effectively. Combining both the camera and the laser rangefinder, a projection of the rangefinder can be used onto the current image position. The rangefinder then can allow for some 3D positioning information that can be beneficial in effectively tracking the moving targets within the robot's’ line of sight. Allowing for more depth information can ultimately lead to further accuracy of the proposed system.

3.9.5. Range Detection

Range detectors work by a simple system of send and receive. A signal is sent out in the direction of the object in question. Once this signal hits the object it bounces back and returns to the source it was emitted from. The calculations are then processed depending on the type of signal that was sent out in order to determine how far the object is.

Take laser rangefinders for example. They work by pointing a laser at an object and waiting for the laser’s reflection to bounce back. Lasers travel just as any light does - by the speed of light. Therefore, since the speed of light is constant, distance can be determined by multiplying the speed of light by half of the round trip time (otherwise known as Time of Flight) of the laser. The formula for calculating the distance as a mathematical expression is:

Speed of Light (m/s) X ½ Time of Flight (s) = Distance (m)

Ultrasonic rangefinders work in a very similar fashion to laser rangefinders. A high frequency sound is emitted towards an object of interest and the time it takes for the sound to return is recorded. Fortunately, sound also travels at a constant speed so long as the air is dry. The same formula applies, albeit replacing the speed of light with the speed of sound. The formula for calculating the distance as a mathematical expression is:

Speed of Sound (m/s) X ½ Time of Flight (s) = Distance (m)

3.9.6. Stereo Vision

This section will highlight the capabilities of a stereo vision and break down the process of how stereo vision operates. Stereo vision may be beneficial in this project by giving further capabilities to 3D image processing. Stereo vision also has the capability of being integrated with SLAM as discussed in section 3.2.1. A list of stereo vision cameras will also be listed to give some examples as to what sensors may be compatible with the proposed system.

Stereo vision consists of two or more cameras; these cameras allow extraction of 3D information from digital images. By having two cameras, the ability to infer depth by triangulation becomes a possibility. This is possible by finding corresponding points in the two images.

In order to successfully explain how stereo vision works, variables will be defined. If one considers two points, P and Q, that are within the same line of sight as the image R, the planes will project onto the same image plane. Being able to narrow the search space for corresponding points from 2D to 1D, these corresponding points can be placed on the same image scanline. This is known as standard form. Standard form allows for “perfectly” aligned corresponding points with the same focal length.

Stereo vision uses disparity and depth in order to successfully process 3D information. Disparity is calculated with the use of similar triangles; in reference to the two corresponding points, it is the difference between the x coordinate. As points are closer to the camera, the disparity increases. Depth can be broken down into a certain amount of parallel planes which corresponds to each disparity value. The depth them makes up the range field otherwise known as horopter.

A stereo vision system consists of four main parts; calibration, rectification, stereo correspondence, and triangulation. Calibration includes both intrinsic and extrinsic parameters such as focal length, image center, and alignment of the two cameras. Rectification uses the calibration in order to remove any lens distortion and also makes sure the stereo images are in standard form. Stereo correspondence attempts to find homologous points if they exist in the two images. Finally, triangulation is the calculation of the correspondence discussed earlier. Each of these parts are crucial for successful implementation of a stereo vision system.

An analogy that aids in further understanding of the concept behind stereo vision is similarity to human vision. Just as an individual has two eyes, stereo vision uses two cameras in order to portray a 3D depiction of an environment. In the human body, each eye has a slightly different view of the same environment. With stereo vision, each camera has a slightly different view by capturing an image of the environment at the same time Using geometric properties, a successful 3D picture can be developed from these two images.

Stereo vision has some key advantages that make using the system more appealing for any robot project that may need 3D information. Stereo vision is a both effective and reliable approach that allows a more practical approach to tracking and detecting objects. Also, it is a passive sensor, meaning that a stereo vision sensor cannot be affected by the environment. Its ability to extract information such as dimensions and color adds to its usefulness in robotic systems.

There are also some issues that may arise while implementing a stereo vision system. One of which is the possibility of distortion and noise in an image. If distortion or noise is present in an image, it may be hard to render a 3D representation of the images. Specular surfaces can also cause an issue with stereo vision systems considering the reflective surface will diminish its actual appearance in 3D rendering. Thankfully, in this project, the use of reflective surfaces is forbidden, so there is less concern about this issue. Foreshortening, ambiguous regions, and perspective distortions are also some issues that can occur.

Stereo vision cameras are outlined below in Table 3.12. Table 3.12 includes different price points, different capabilities, as well as key specifications to look for while comparing these cameras for this project. Some things to consider while choosing a camera for this project is how many frames per second are necessary to complete the objective, keeping in mind that focal length will be constantly changing, and how to connect this camera to corresponding components. As Kinect is one of the top sensor choices for this project, it has been written about separately in its own section (reference 3.3.2.3). Prices of stereo cameras are very high for the desired budget, comparing prices is key.

Name/Option	Price ($USD)	FPS	Resolution	Pros/Cons
Kinect 3D	$132.04	30	1280x960	Great camera for budget. Compatibility may be an issue.
BlackBird 2 3D FPV Camera	$179.00	60	680x512	Has great reviews; most expensive camera choice; very high up on $1000 build budget.
NerdCam3D Mk.2 Stereoscopic FPV 3D Flight Camera	$149.99	N/A	640x480	Cannot find exact frame rate for product. Product reviews are not found; hard to gauge performance.
BlackBird 1 3D FPV Camera	$89.00	30	656 x 492	Decent camera for the price. Older model of the BlackBird 2. Kinect 3D is not much more for a much higher resolution.

Table 3.12: Comparison of Stereo Vision cameras within budget

3.9.7. Facial Detection

Perhaps the easiest target to detect will be the stationary targets. This is because apart from taking advantage of the fact they are stationary and that their location on the course will be known, we are told they will be pictures of faces. This is an enormous detail because to detect faces we can use one of computer visions best work, facial detection.

Facial detection is perhaps the most reliable target detection we will have. The reason is that there happen to be machine learning programs trained for this purpose. Machine learning algorithms or classifiers use what are called training sets, to learn the best distinguishing features found in objects you label. Essentially you feed the classifier a dataset composed of pictures with faces and pictures without faces. You teach the algorithm what the pictures contain and the algorithm determines a model that predicts what a new test image contains. Machine learning algorithms can have very high accuracy rates, but they require large datasets with hundreds of training images. Since we will not acquire any images of the other team's robots or the obstacles on course there will be no way of training an algorithm to detect them.

The Haar Feature-Based Cascade classifier is one widely known method of facial detection. This face-detection algorithm is already implemented in OpenCV. It is believed to be a reasonable solution because it will not require the collection of a dataset or training of a classifier. It is ready to use with a few lines of code and can perform in real-time.

In the end, facial detection is only one part to our target detection pipeline. It will be processed concurrently with the other detection algorithms. The detection of faces will give us a form of receiving points, and orienting the robot on the course.

3.9.8. Histogram Equalization

There are many potential downfalls to computer vision techniques. One possible problem we may encounter is with the brightness of the lighting at the course. It is probable that the lighting at the competition will be different to the conditions we develop and test under. This could change the outcome of our vision algorithms as they depend on the intensity of pixels to calculate their procedures.

We cannot change the circumstances of the course, but we can try image processing techniques such as histogram equalization to correct them. A histogram is a method of counting the pixel intensities found in image. Histogram equalization attempts to flatten or even the distribution of pixel intensities to increase contrast.

As seen in the first image, Figure 3.9, the frequencies of the middle pixel values dominate the image. This can pose a problem to object recognition techniques such as edge detection. The second image, Figure 3.10, shows the results of the histogram equalization method in OpenCV. It is easy to see that the puppy is more salient and can be distinguished from the background.

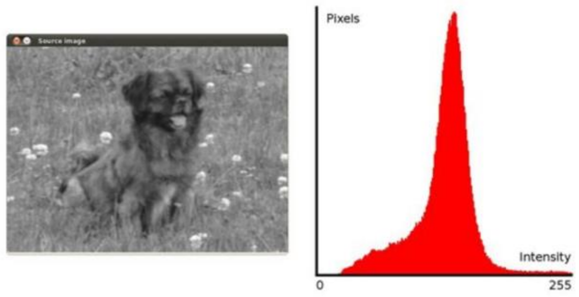


Figure 3.9: Original low contrast image
(Permission Requested from OpenCV)

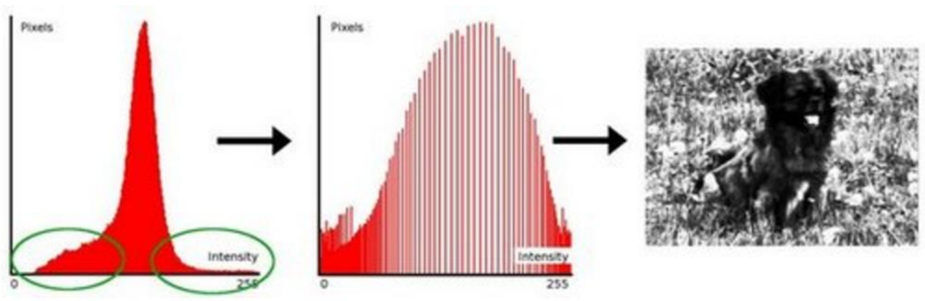


Figure 3.10: Shows Histogram Equalized Output
(Permission Requested from OpenCV)

3.10. Power

Power is the most important factor for the robot. Without power, the robot cannot function. The robot for this project will need to have enough power to supply the sensors, motors, motor controllers, microcontrollers, and computer processor. Since the battery must be able to last up to two 10 minute rounds, it is important to know what type of battery will best serve this project. Usually selecting a battery is the last component to add to the project, since there are calculations for each component and a desired lifetime of the robot. Once all the components are assembled, the battery can be chosen.

However, not all batteries are the same. Some require more maintenance than others and some batteries will not last as long as others. In this chapter, several different types of batteries are researched to compare the advantages and disadvantages of each. There are also different battery products that were examined to best function for this project.

3.10.1. Sealed Lead-Acid

Sealed Lead-Acid batteries are most commonly found in automobiles and small vehicles. What makes lead-acid batteries unique, is the ability to supply bulk power at low cost and that they are the most reliable and cost effective batteries on the market. They are very robust and take plenty of abuse without failing. They can discharge high current with ease. Sealed lead-acid batteries are capable of having a long shelf life and can be left on trickle or float charge. These batteries can come in several different capacities and have a large range of sizes to choose from.

There are some disadvantages in sealed lead-acid batteries. These batteries are are heavier than any other battery and can only have a charge efficiency of 70% after a certain amount of cycles. These batteries can take up to 14-16 hours to fully charge. These batteries are also prone to leaking and not very environmentally friendly. Deep cycling can also drain the battery life since there is a charging memory. For this project, it would be ideal to use a sealed lead-acid battery as they are the most heavy duty battery capable of powering the battlebot robot. Table 3.13 contains the comparisons between 12v, 3ah sealed lead acid batteries in terms of performance, size, and price.

Type	**Brand**	**Max cycles at 100% discharge**	**Weight (lb)**	**Dimensions (in)**	**Capacity (AH)**	**Price ($USD)**
AGM SLA	Duracell	150	3.0825	5.28x2.64x2.63	3.3	$30.99
AGM SLA	Pirate Battery	-	4.00	3.54x2.76x3.98	4.0	$21.00

Table 3.13: Comparison of Sealed lead acid batteries

3.10.2. LiFePO4

Another alternative for a battery is the Lithium Iron Phosphate battery, or LiFePO4 for short. These type of batteries are commonly found in electronics, vehicle use, and backup batteries. They are a relatively new battery technology, invented in the 1980s. While cordless tools and laptops rely heavily on this type of power, LiFePO4 batteries are known for being much safer than the sealed lead-acid batteries and can recharge much faster. LiFePO4 batteries also have a larger useable charging capacity, over 50% more than a sealed lead-acid battery, and have a lifetime cycle use of over 10 times the lifespan than of a sealed-lead acid battery. There is no risk of overcharging with an LiFePO4 battery. Even after 2000 cycles, an LiFePO4 battery can still supply 80% of its discharge. It is also known that the LiFePO4 battery can still supply the same amount of output voltage at 20% as it would at 80% discharge. According to the PowerTech Systems website, the LiFePO4 batteries are over 77% smaller and 194% than a sealed lead-acid battery. Even with a smaller size and smaller weight, the energy and power supply remains constant. There are not too many disadvantages of a LiFePO4 battery to name. A LiFePO4 battery is expensive to manufacture and requires a complicated circuitry to maintain and regulate the battery’s safety and control. LiFePO4 batteries are also sensitive to overheating and there are known safety concerns in which these batteries can explode or catch fire. Therefore, LiFePO4 batteries will be of great consideration for powering the computer processing and the PCBs for this project. They have a small nominal voltage, are extremely lightweight, and can supply constant voltage no matter how discharged the battery may be. A few selections have been made in either considering the battery to power the entire robot or just supplying power to the electronics such as the microcontroller and PCB design. Table 3.14 contains the specifications for batteries of choice.

Product ID	Voltage (V)	Capacity (mAh)	Charging Rate	Weight	Price ($USD)
LiFePO4 18650	12.8	2200	2.2A standard	295g	$80.00
Tenergy Li-Ion	7.4	2200	0.4A standard	99g	$13.99

Table 3.14: Comparison of LiFePO4 batteries

3.10.3. NiMH

A nickel-metal hydride (NiMH for short) battery is a popular choice among consumers for being the most reliable and low cost rechargeable battery. They can behave like a nickel cadmium battery, but with the different chemical responses and NiMH contains a higher energy density and three times the capacity. Some would say NiMH batteries are a less expensive version of the lithium ion battery.

For charging NiMH batteries, there is a risk of overcharging and some companies suggest keeping the charging time between 10-20 hours. Trickle charging is the safest method. If the battery is heating up during trickle charging, then the current supply is too high and could lower the efficiency of the battery. Although trickle charging is the safest, NiMH require fast charging in order for NiMH batteries to last longer. Lower current charging can cause battery memory, which is a case where a battery can lose its capacity over time.

The few disadvantages of a NiMH battery are that these batteries have a short shelf life due to the fast charging they require and the battery memory it can develop. They also require a complex charging algorithm. These batteries can generate high heat during charging and contain a high self-discharge. For choosing NiMH batteries, there is an incredible amount of resource in these kind of batteries. These batteries may be chosen because of their charge capacity, the time it would take to charge them, and their cost. Table 3.15 shows a side by side comparison between two batteries to consider for this project.

Product ID	Capacity	Dimensions	Weight	Price ($USD)
#9311	12V/300mah	60x30x19(mm)	3.8oz	$22.50
#6056	12/800mah	51x20x46	4.8oz	$16.95

Table 3.15: Comparison of NiMH batteries

3.10.4. LiPO

The research for the battery suitable for this project goes further with LiPO batteries. These batteries are popular among consumers who look to use batteries over radio frequencies and require high power output. If our robot needs a video datalink and the use of radio frequencies to manually operate the robot remotely, the LiPO battery will be able to provide enough energy to do so.

LiPO batteries contain some advantages over the NiMH batteries and there are a few to name. The first advantage from LiPO batteries is that these batteries are much lighter in weight, these batteries can offer higher capacities, and can project a higher discharge weight than of a NiMH battery. LiPO battery cells are stored in pouch form which allows them to be lighter and be flexible on retaining shape. This can be considered if there is a lack of space and strict weight limitations in the project. Every battery contains a setback or two and the LiPO battery will definitely hold some disadvantages. These disadvantages are that they have a shorter life span than of any of the batteries that have been compared. There are safety issues, such as the risk of a fire or explosion occurring if these batteries are punctured. These batteries also require special care for storing and charging.

LiPO batteries have a nominal voltage of 3.7V. If more voltage is required, they can simply be added in series. The capacity of a battery can be determined just like any other battery from the amp hour these batteries can consume. LiPO batteries can consume anywhere from 30mAh to 22,000 mAh. Another factor to consider is the discharge rating of the LiPO battery. This is the amount of discharge a battery can give without ruining the battery. The notation ‘C’ is used to determine the capacity in amps. Most applications will use 20 or 25C as a battery but for heavier duty applications such as a truck or large vehicle, 40C battery would need to be used.

Table 3.16 below contains the results from research on several LiPO products that could potentially be used for this project to best serve powering the motors, microcontrollers, and sensors.

Brand	Voltage (V)	Discharge Rate (C)	Capacity (mAh)	Price ($USD)
Dynamite	11.1	25	1500	30.99
Ellite	11.1	20	1000	24.99
E-flite	11.1	20	1250	23.99

Table 3.16: Comparison among different LiPO products

3.11. Voltage Regulators

Voltage regulators are used in order to maintain constant output voltages. These are important for power distribution and stability. While motors need to operate on 12V or less, circuit boards and microcontrollers require much less voltage and current, such as 5V or 3.3V. The next two subsections of this chapter explore the options of how linear and switching voltage regulators will be beneficial in maintaining constant voltages for this project.

3.11.1. Linear Voltage Regulators

The circuits in our project will need a constant voltage to avoid the risk of instability or overpowering the circuit. Linear voltage regulators contain a voltage controller current source in order to maintain a constant voltage at the output. Texas Instruments suggests that one must follow a few different requirements when picking out the appropriate voltage regulator.

For the best application there must be some evaluation in knowing the maximum load current, the kind of input such as a battery power source, the output voltage precision, and what special features are available. These requirements are known from previous sections in what kind of current our motors, microcontrollers, and sensors will use. Since our components are in a range from 3.3V to 12V voltages and 1A to 3A currents, there are a few linear voltage regulators that could be deemed suitable for the purposes of our project.

For the maximum load current, one must take into consideration an important value of the load current. Ultimately, the regulator must be able to withstand worst case scenarios in order to maintain a reliable performance. If our microcontroller is going to operate around 500mA, it would be ideal to keep that as the maximum current for the proposed system.

Our input source will be a battery. For applications of a battery power source, LDO regulators will be used as they are highly recommended since they can utilize the available input voltage fully.

Finally, special features - these are provided to add flexibility to our design. Some of these features in LDO regulators include a shutdown pin which can allow a regulator to be shut off through a microcontroller. Another special feature is a load-dump protection, which can be considered for the motors where a regulator can automatically shut down during overvoltage and turn itself back on when the overvoltage passes. All of these special features will be taken into consideration with the development of the proposed system’s design.

Some of the choices in voltage regulators to take into consideration are the LM78xx series. These voltage regulators are easy to install and are low in cost. The LM7905 voltage regulators can supply an output current up to 1.5A and can take in a range of 7V to 25V.

3.11.2. Switching Voltage Regulators

Switching voltage regulators, like the one seen in Figure 3.11, are also an important device in power and voltage regulation in electronics. They can regulate an input voltage by switching on and off a series element to maintain a constant voltage. These devices can operate very much like a linear voltage regulator, but can be used for higher voltages in order to avoid wasting a lot of power. Linear voltage regulators are beneficial to lower voltages therefore switching voltage regulators will be used in keeping a steady voltage amount the motors. Switching voltage regulators are fully conductive devices when turned on or completely shut off, so no power is dissipated, allowing these devices to be high in power efficiency. They can also convert DC to DC power more efficiently than that of a linear voltage regulator.

There are many switching voltage regulators on the market. The choice of a voltage regulator can vary based on how much frequency is needed for switching as well as what range of voltage regulation is required. These frequencies range from 300 kHz up to 4 MHz, allowing full flexibility for this project. The choice of using switching voltage regulators will be determined if, during testing, there is a high dissipation of heat within linear voltage regulators or if a large heatsink would be undesirable.

A common switching voltage regulator would be the DE-SW0XX family of switch voltage regulators shown in Figure 3.11. These devices allow for taking in any voltage and stepping it down to the 5V voltage requirement suitable for microcontroller. It can step down these voltages in an efficient manner with no power dissipation, up to 87% efficiency. There is a drop out voltage in which the regulator will not be able to regulate power and that is at 1.3V. It drives a continuous output current of 1A and has the ability to drive inductive loads. It can also work off a breadboard for easy prototype testing and hardware designing.

3.11.3. DC to DC Converters

While there are voltage regulators that can step down voltage, the standard LM78XX do not supply enough current for the NERF blasters and TK1. Therefore, DC to DC convertors must be used. While there are step up and step down DC to DC convertors, this project implements variable step down. The table below compares the different type of convertors considered.

Model	Voltage Range	Max Current	Short protection	Working frequency	Price
DROK DC Car Power Supply Voltage Regulator	5-40V	12A	Yes	300kHz	$10.99
DROK 5 Pcs DC Step Down variable Voltage Regulator	4.5-40V	3A	Yes	150kHz	$19.99
DZS Elec DC-DC Step-down Constant Current Regulator	4-38V	5A	Yes	180kHz	$9.49

Table 3.17 Comparison of DC-DC convertors

3.12. Chassis

The chassis of the robot will be the outer case to protect the internal components and provide a protective layer against incoming projectiles from enemy robots. The chassis will be designed and provided by the Mechanical team.

3.12.1. Preliminary Chassis Design

The design is focused around the design of a military tank. Following the specifications of the size, which is limited to 3ft x 3ft x 3ft, all the electrical components will be able to fit within the case of the robot with the turret and sensors elevated by a tripod structure from the center of the robot's chassis. The material for the chassis will be manufactured by Lockheed Martin and is planned to be 3D printed. The material for this chassis is to be made of ULTEM resin, which provides incredibly high thermal resistance and high strength and durable stiffness.

Figures 3.12-3.15 illustrate how the body of the robot is built. Figure 3.12 shows the front view of the chassis. As seen in the figure, there is a platform with 4 walls inclining inward for stability. All the circuit boards will be placed on the platform and will be protected by the walls of the chassis.

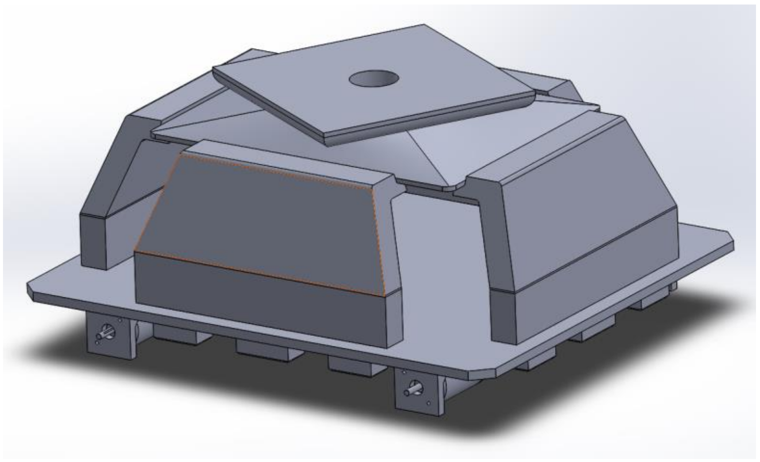


Figure 3.11: Front view of chassis
(Team Designed)

In Figure 3.13, there is the bottom view of the chassis. This is where all the motors will be mounted with any necessary wires to be placed along the perimeter of the robot. The DC motors will be connected and wired through holes that will be drilled into the platform so that the wires will remain organized and secured throughout the competition.

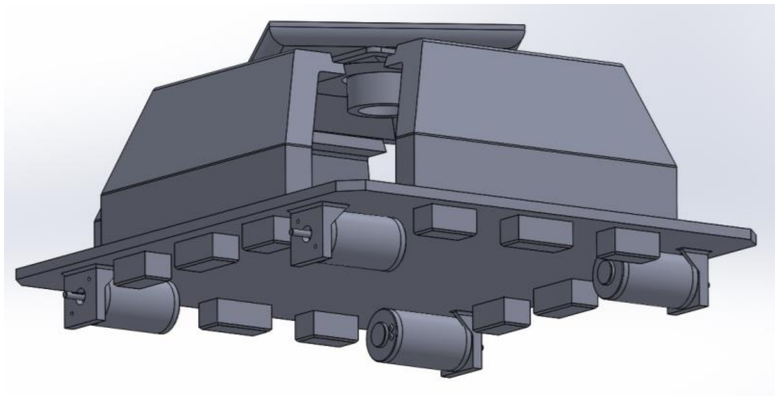


Figure 3.12: Bottom view of chassis
(Team Designed)

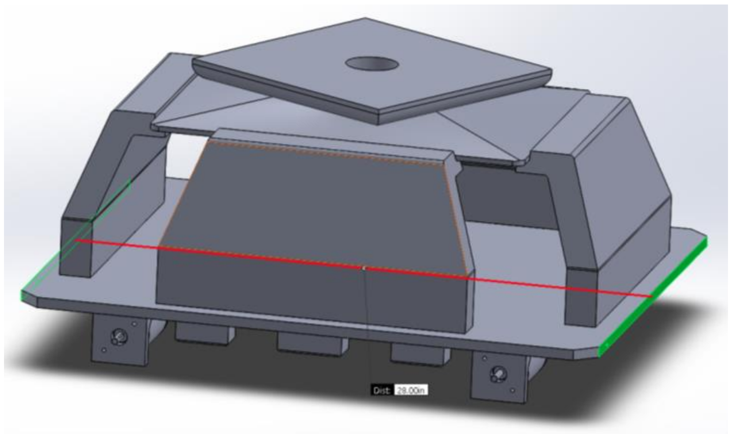


Figure 3.13: Side view of chassis
(Team Designed)

Figure 3.15 shows a model of the desired look for the battlebot robot. The two Nerf-blasters will be mounted in the middle and will rotate freely from the platform of the robot. There will be two sensors in the middle of the two Nerf-blasters to produce the firing mechanism subsystem. This subsystem will be mounted onto a rotating tripod structure.

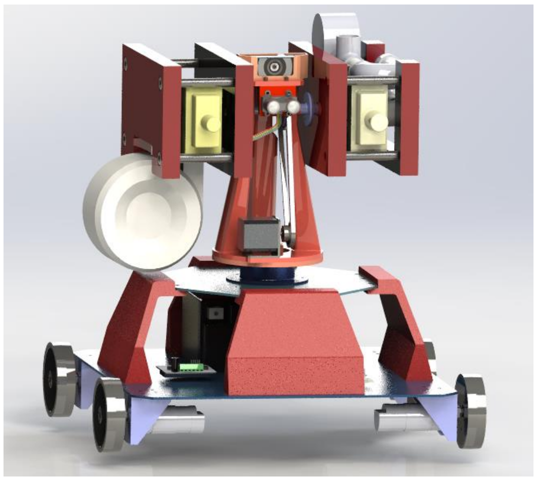


Figure 3.14: Completely assembled robot
(Team Designed)

3.12.2. Final Chassis Design

The final design of the Nerf enabled battlebot, while highly comparable to the preliminary design, features a multitude of minor changes. The structure of the chassis was not manufactured by Lockheed Martin or even 3D printed with ULTEM resin. For cost reductions and faster manufacturing times, the structure of the robot was designed with multiple layers of laser cut MDF wood. The material is highly durable, extremely affordable, and easily modified by tools such as a power drill. The sidewalls, however, maintained the same design with the caveat of being manufactured by a lower quality 3D printer - still the 3D printer material is able to endure the weight of the top turret system.

Figures 3.12-3.15 illustrate how the final body of the robot is built. Figure 3.12 shows the front view of the chassis. The changes with the MDF wood can be seen in the figure.

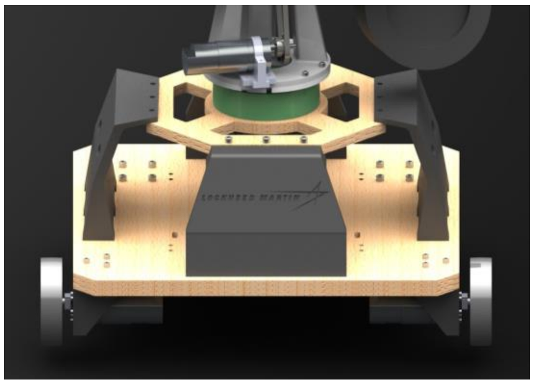


Figure 3.15: Front view of chassis
(Team Designed)

In Figure 3.13, there is the bottom view of the chassis. The DC motors are still connected and wired through holes that are drilled into the platform so that the wires will remain organized and secured throughout the competition. Most of the power wiring is maintained under the robot for easy recognition of power distribution. As seen in the figure, the battlebot now features only two DC motors at the front of the robot, while caster wheels take their place at the back of the robot. This modification was done because of budget cuts, reducing the cost of manual navigation directly whereas the motors are still providing enough force to drive the robot.

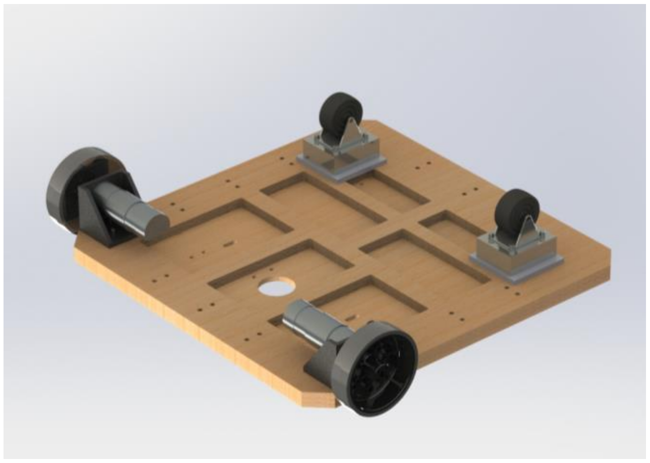


Figure 3.16: Bottom view of chassis
(Team Designed)

Figure 3.14 provides a better view of how the caster wheels are seen from the side of the robot. They are tucked in slightly to withstand the weight of the robot without buckling inwards or outwards.

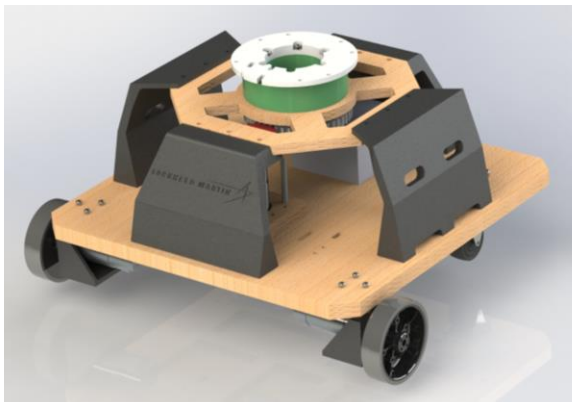


Figure 3.17: Side view of chassis
(Team Designed)

Figure 3.15 shows a model of the final look for the battlebot robot. The two Nerf-blasters are still mounted in the middle and will rotate freely from the platform of the robot. The

Nerf-blaster positions have been swapped because of a small error having to do with the orientation of the extra ammunition cartridge of the nerf dart gun; the gun would not mount properly on the opposite side. The two sensors remain placed in the middle of the two Nerf-blasters to produce the optimal viewing angle. The turret system is situated onto the original rotating tripod structure.

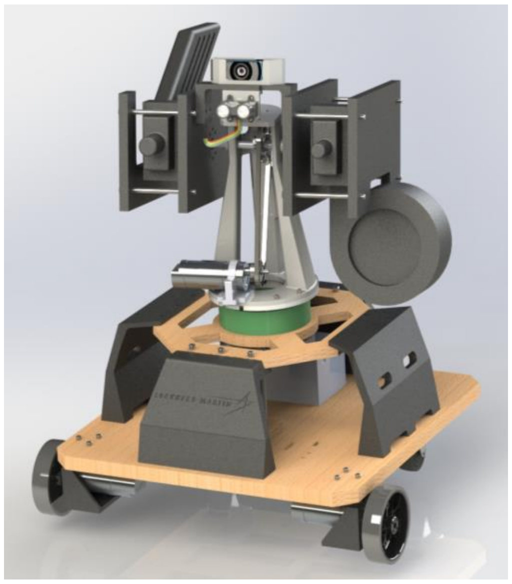


Figure 3.18: Completely assembled robot
(Team Designed)

3.14. Nerf-Blaster

The firing mechanism for this project will be using Nerf-blasters. The rules of competition allow a minimum use of one Nerf-blaster that can shoot Nerf-balls and a maximum use of two. If a second Nerf-blaster is used it would be required to shoot Nerf-darts.

3.14.1. Nerf-Blaster Selection

For choosing Nerf-blasters for this project, it was important to take in the consideration of how far the ammo could reach. It was required that up to 45 feet was needed to ensure that the dart and ball blasters would reach impact. Table 3.14. contains a full run down of the different types of NERF blasters considered.

Name	Ammo Type	0-Angle Range	Velocity	Price
Rival Zeus MxC-1200 Battle Gun	Ball	65-75 feet	100 feet/second	$39.99
Rival Khaos MXVI-4000	Ball	65-75 feet	100 feet/second	$62.99
N-StrikeElite Rampage	Dart	50 feet	75 feet/second	$39.9
Rapidstrike CS-18	Dart	55 feet	75 feet/second	$39.99

Table 3.18 NERF Blaster Comparison

For the Nerf-blaster that will be firing Nerf-balls, the initial choice was the Nerf Rival Zeus MXV-1200 blaster (seen in Figure 3.18). This product uses a motorized blaster with the capability of firing rounds up to 12 high impact rounds at 300mps. The cost of this blaster is $49.99.

Figure 3.19: Nerf Rival Zeus MXV-1200
(Permission Requested from Hasbro)

The second Nerf-blaster that will be used is the CS-18 N-Strike Elite Rapidstrike (seen in Figure 3.19). This Nerf-blaster will be firing the more common NERF dart ammo. These darts can fire up to 75 feet on this weapon. This gun will require 6V to fire the darts as it uses a powering motor to increase the rate of fire. It has been planned to alter the firing mechanism for a faster rate of fire and install a larger ammo magazine. The CS-18 N-Strike Rapidstrike is regularly priced at $39.99.

Figure 3.20: Rapidstrike CS-18
(Permission Requested from Hasbro

3.14.2. Nerf-Blaster Research and Development

A NERF-blaster’s function is simple. It acts on a plunger mechanism, as the spring in the barrel of the blaster is pulled back, a dart or ball is loaded into it. Once the spring is released, air is pressured into releasing the ammo. Since NERF ball and darts are made of foam, they can travel at a high velocity.

The NERF blasters were opened up to discover that each motor had a motorized flywheel and a motorize triggering system. The implementation was to power the flywheels to be on at all times and for the darts to be fed into the flywheel from the remote triggering.

For the Rapidstrike CS-18, all the safety switches were bypassed and wire extensions were made from the flywheel motors and the triggering system. This NERF blaster operated on 6V and needed a 3A start up for the flywheels. There was also a consideration in delay time to ensure that the trigger motor would not jam with the darts. Through testing, it was found that in order to keep the darts from jamming, the fire blaster had to shoot in intervals of 2 to 3 darts per activation. This was to allow enough time for the pushing trigger to complete a full cycle and not interfere with the dart ammo reloading mechanism.

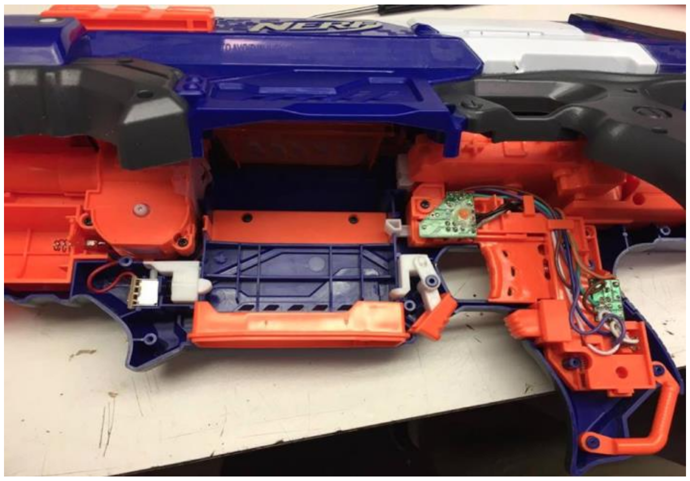


Figure 3.21 Insides of Rapidstrike CS-18

Our second NERF blaster system was initially the Rival Zeus MXV-1200. This NERF blaster could shoot a magazine of only 15 balls. It contained a flywheel but no triggering system like the dart NERF blaster had. Therefore, with collaboration with the mechanical team a modification was made to extend the magazine to 50 balls and use a servo motor to feed the balls into the flywheel.

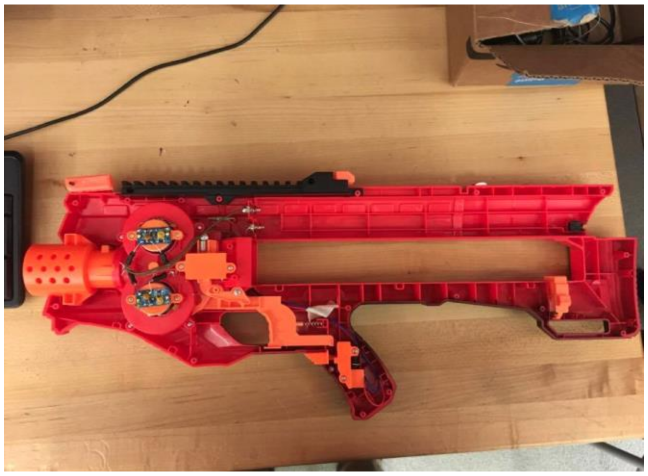


Figure 3.22 Insides of Rival Zeus MXV-1200

This modification was based on a design from a Youtube channel, “Out of Darts.” A 12V fan was used to push the balls through the tubing and the servo motor was going to act as a switch to let the balls feeding into the flywheel. Unfortunately, there was not enough time to make this modification work properly, since there were difficulties with the fan not supplying enough force to push the ball.

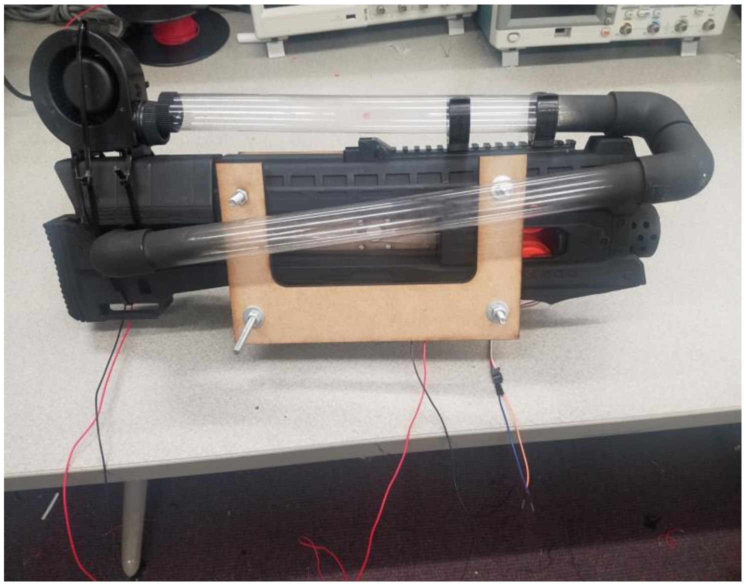


Figure 3.23 Rival Zeus MXV-1200 modification

Since two blaster systems were still needed, the final design implemented a Rival Khaos MXVI-4000. This NERF blaster utilized 44 ball magazine and was able to shoot at far range. This $70 NERF Blaster operated on 9V.

Figure 3.24: Rival Khaos MXVI-4000
(Permission Requested from Hasbro)

This NERF blaster was taken apart and all the safety switches were bypassed. This blaster contained a flywheel and a conveyor belt to feed the darts through to the flywheel. The flywheels remained on at all times and the conveyor belt was used to be turned on through the microcontroller.

Figure 3.25: Insides of the Rival Khaos MXVI-4000

3.15. Motor Controller

While the microcontroller is responsible for sending the controls for direction and speed to the motors of the robot, it still does not contain enough power to drive all the motors. A motor controller, displayed in Figure 3.20, will be able to work in connection with the microcontroller and also connect to the power supply in order to drive the motors. The robot will be driven by four DC motors, two stepper motors, and a servo motor. With a total of seven motors for powering, there are considerations in choosing the right motor controller to do the job.

For choosing a motor controller for the DC motors, the nominal voltage must be known. Once the voltage is known, the continuous current is determined. It is important to know these two values for balance and stability. Too much current can fry the microcontroller and not enough current will not turn the motors. There is also the type of control these motors will use. These controllers can be pulse width modulation, analogue vogue, UART, or R/C.

In section 3.9 Motors, several motors were researched and a motor controller can now be chosen. The nominal voltage for the DC motors and stepper were 12V and contain a continuous current of 2.875A. The servo motors do not require the same amount of voltage and current of a DC motor therefore it is not suitable to connect them in parallel them. Figure 3.20 shows a setup of how the motors would be connected to the microcontroller and driven by the motor controllers.

There are thousands of motor controllers available and there is one motor controller that will be able to power all seven of the motors. A motor controller to consider for this project is the called the Adafruit Motor/Stepper/Servo Shield for Arduino v2. This motor controller uses MOSFET technology and contains its own pulse width modulator driver chip for easy power control. It has 2 connections for the servo motors, can drive up to 4 DC motors and supply power to 2 stepper motors. The price for this motor controller is only $19.99 and if more motors will be required later on, this motor controller can stack 32 times, supplying up to 128 DC motors.

Another motor controller to consider for this project is the 10A 5-25 Dual channel motor shield by Crypton. This motor controller is capable of driving up to two bidirectional motors. These motor controllers support both locked-antiphase and sign-magnitude PWM operation. Each motor controller is equipped with activation buttons for quick power testing. They sell for $23.48 USD per board and the integration step up to this project is easy.

3.16. Manual Navigation

Manual navigation is imperative to the development of the proposed system. Manual navigation will be the main means of controlling the robot’s movements. This section will highlight the research gathered regarding manual navigation. It will include sections discussing PID control, odometry model, and how the robot will transfer data in order to allow the user to successfully manually navigate wirelessly.

3.16.1. PID Control

There are various movements of which each of our different motors will have a responsibility for. The four DC brushed motors will be used to drive the wheels of the robot around the course like a car. The two stepper motors will be used to aim the Nerf-Blasters in synchronized fashion. One to pan or move them horizontally and the other to tilt them or move them vertically. The last motor or the servo, will be used to tilt the camera to keep the target centered in the field of view. In order to direct all these movements, we will design a PID controller. A PID controller incorporates a feedback loop between an input signal sent to the motors and their outputted physical movement, to smoothly reach its goal.

PID controller stands for proportional integral derivative controller. Each of the names in the abbreviation are different controllers; each uses a particular model of computing a gain to reduce error. Error compares the difference between an input at a given time to its desired output or goal. The different functions of these controls are shown in Table 3.17 below.

Proportional	$o = K_p * (V_d - v) = K_p * i = K_p * error(t)$
Integral	$o = K_i * \int i(t) * dt = K_i \int error(t) * dt$
Derivative	$o = K_d * \frac{di}{dt} = K_d * \frac{d(error(t))}{dt}$

Table 3.19: PID Control Functions

In the table shown above, "o" refers to the measured output of the motor. The "K" terms are constant parameters, which are tuned together. The "i" terms are the error measurement, which is the difference between a current position or speed to the desired position or final speed. The proportional controls the motor in a manner proportional to the instantaneous error. The derivative controller avoids the motors overshooting by acting in proportion to the rate of change of the error. The integral controller acts in proportion to the accumulated error, which is useful for eliminating steady-state errors.

3.16.2. Odometry Model

We are using encoders in our wheels as instruments of measuring our robot's movements to determine its location in the course. In order to do this, we will need to use an odometer model to convert the information read from the encoders to real world coordinates. The model we intend to use is detailed below.

- Begin with the following equations relating the change in position of the wheels to the circular arc of the turning axis as seen in Figure 3.21.

  Eq(1): $\Delta sl = R\alpha$

  Eq(2): $\Delta sr = (R+2L)\alpha$

  Eq(3): $\Delta s = (R+L)\alpha$

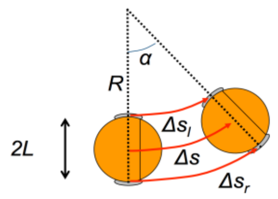


Figure 3.26: Odometry p1- Example showing the relations of variables (Permission Requested from Princeton)

- Use Eq(1) and Eq(2) 2 to solve for the distance from the right and left wheels to the axis of the turn.

  Eq(4): $R\alpha = \Delta sl$

Eq(5): Lα = (Δsr - Rα)/2 = Δsr /2 – Δsl /2

- Substituting Eq(4) and Eq(5) into Eq(3) we represent Δs as a function of Δsl and Δsr.

  Δs = (R+L)α = R α + L= Δsl + Δsr /2 – Δsl /2 = Δsl /2 + Δsr /2
  Δs = (Δsl + Δsr) /2

- To calculate the change of the angle of the robot's orientation Δθ, is equal to the circular arc's center point as seen in Figure 3.22.
  Δθ = α

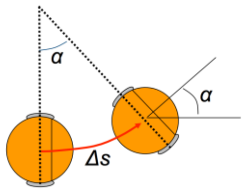


Figure 3.27: Odometry p2- Example relating circular α to the orientation (Permission Requested from Princeton)

- Then solve for α by equating α from Eq(1) and Eq(2).
  Δθ = (Δsr - Δsl) /2L

- With Δθ and Δs we are able to calculate the position change in real world coordinates. Using trigonometry to model the Δx and Δy as functions of Δd.

  Δx = Δd cos(θ + Δθ/2)
  Δy = Δd sin(θ + Δθ/2)

- As seen in Figure 3.23 the paths Δd ≈ Δs are slightly different. The measurements will be made fast enough to approximate that Δd ≈ Δs.
  Δx = Δs cos(θ + Δθ/2)
  Δy = Δs sin(θ + Δθ/2)

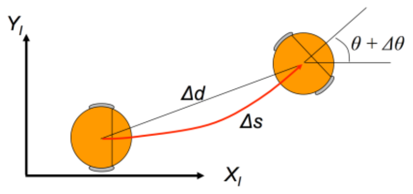


Figure 3.28: Odometry p3- Example denoting difference of Δd and Δs (Permission Requested from Princeton)

Using the above model, we will be able to approximate our robots location on the field. The coordinates will not be exact because of the assumptions that were made. Also there will be noise from the encoders that will impact the calculations. Other causes of inaccuracy to the model can arise from the wheels slipping, hitting a bump, or driving on carpet.

3.16.3. Wireless Data Transfer

In order to successfully develop a plan for functional manual navigation of the proposed system, different methods of transferring data to a control device must be considered. The desired method will have a great impact on how the robot will be programmed. Since a requirement of this project is to wirelessly provide video footage of the robot’s autonomous tracking capabilities in real time, it is logical that both the video footage and the manual navigation will be wirelessly operated through a laptop. One can then control the robot solely by viewing the streamed video on the laptop and using the arrow keys to navigate accordingly.

Generally, there a three ways that the robot will be able to interact wirelessly with laptop in order to successfully navigate through the course environment. The first way that can be chosen is using radio frequencies, RF, which uses a transmitter and a receiver to send data across a specific frequency. One benefit of using radio frequencies is that they allow for an extremely long range. Radio frequencies also allow for a straightforward setup of the system.

Two different wireless protocols that can be explored that utilize radio frequencies are ZigBee and Xbee. Zigbee allows for a low-cost and low-power wireless connection between devices. It can operate at many different frequencies, such as 900MHz, 868 MHz, and even 2.4 GHz. There is a low latency while using Zigbee, which can be beneficial to the proposed system. The data rate of Zigbee is about 250 kps, which is much lower than that of Bluetooth and Wifi. Xbee is very similar to Zigbee, as it includes all the features of Zigbee, but has new added features. Zigbee’s features already prove to be efficient, but lacks in data rate transfer. Unfortunately, Xbee does not provide a higher data rate than Zigbee.

Although radio frequencies prove to be robust enough to handle manual navigation of the system, transmitting the video footage as well with the manual navigation is seemingly close to impossible with radio frequencies. Transmitting large files such as video over a radio frequency requires a very high bandwidth, which may not be available within the region of the competition destination. As stated earlier, a goal set in place is to obtain the information for both the video footage and manual navigation through the same means. Therefore, other options must be explored.

Another more specific RF option for the manual navigation of the robot is to use Bluetooth in order to wirelessly obtain data. Bluetooth will allow for two-way communication of data from the laptop to the robot via the Raspberry Pi. Even though Bluetooth is a RF, Bluetooth allows for higher data rates than standard RFs, since it follows specific protocols for communication.

Although Bluetooth provides a great way to both send the video stream and data from the motors for manual navigation, Bluetooth only allows for a maximum range of 10 meters in most cases. With the full length of the field being roughly 40 feet, another option may prove to be more beneficial. Even though the robot will most likely travel no farther than 20 feet away from the sidelines, since the robot should be staying in its required zone, there could be a possibility where the robot will be more than 10 meters away from the controller. If this proves to be a considerable trade off, it may be better to choose another option for manual navigation of the robot.

The last option explored for the robot to wirelessly send data for both the manual navigation and the video stream is to use a Wi-Fi network. Wi-Fi will allow the user to connect to the proposed system from anywhere in the world. Wi-Fi will also provide a significantly high data rate that should prove to be sufficient for transmitting both the data for the video stream and the manual navigation. However, the competition location will not provide Wi-Fi that the robot will be able to connect to. Therefore, a local network may be considered a solution to this problem. By establishing a connection between the Raspberry Pi and the laptop via a router, the proposed system will be able to indeed transmit the necessary data in theory to the laptop. This can be considered the most reliable option to providing a secure connection between the proposed system and the laptop.

After all options have been explored, radio frequencies, Bluetooth, and Wi-Fi, these options were compared thoroughly in Table 3.18

	Data rate	**Maximum Range**	**Power Comparison**
Radio Frequencies (Zigbee)	250 Kb/s	7 meters	Low
Bluetooth	25 Mb/s	Normally 10 meters	Medium
Wi-Fi	54 Mb/s	Normally 100 meters, but can be unlimited	High

Table 3.20: Comparison of Data Transfer Wirelessly

3.17. Component Selection Summary

The Part Selection Summary describes the components that were selected at the time of drafting the project and components that were selected while testing and finalizing the battlebot design.

3.17.1. Preliminary Component Selections

Table 3.19 below contains a list of the main functional computer/electrical components that are to be implemented in this project's design.

Component	Name	Quantity
Image Sensor	Logitech HD Pro Webcam C920	1
Rangefinder	LIDAR-Lite 3 Laser Range Finder	1
Processing Unit	Raspberry Pi 3 Model B	1
Microcontroller	ATmega328p	2
Motorshield	10A Dual Channel Bi-directional DC Motor Driver	2
DC Motors	NeveRest 40 Gear Motor	4
Encoder	*Attached to NeveRest 40 Gearmotor	4
Servo Motor	Futaba S3004 Standard Servo Motor	2
Stepper Motor	3V 1.7A 68oz-in Stepper Motor	1
Nerf-blaster (Dart)	CS-18 N-Strike Rapidstrike	1
Nerf-blaster (Ball)	Nerf Rival Zeus MXV-1200	1

Table 3.21: List of main computer/electrical components selected

The Logitech HD Pro Webcam C920, seen in Figure 3.24, was selected as the primary vision sensor. This camera provides for better picture quality and more than enough frame rate needed. It is highly compatible with other systems as it is attached by a simple USB cable, which is included in the packaging. Additionally, the USB cable has a length of six feet; about 60 inches longer than the ribbon cables that come with the other camera alternatives. A lengthy cable is necessary as the vision sensor will sit at the highest point of the robot and may need to twist and turn.

Figure 3.29: Logitech HD Pro Webcam C920
(Permission Granted by Logitech)

The selected rangefinder was the Lidar-Lite 3, shown in Figure 3.25. While there are many alternative products that are capable of finding the range of an object, not many seem to provide the desired range and accuracy as does the Lidar Lite 3 at its price point. At a regular price of only $150, this rangefinder will be able to detect objects at more than three times the length of the competition course and with comparable accuracy of higher end models. The use of laser range detection was also attractive as it is more likely to come across sound or radio interference than it is to come across light interference since the course is set indoors.

Figure 3.30: LIDAR-Lite 3 Laser Range Finder
(Permission Granted by RobotShop)

For image processing, the Raspberry Pi 3 Model B (Figure 3.26) was selected because of its affordability and popularity on the market. This microprocessor comes with enough memory to allow for image resolution flexibility when running detection algorithms. It is a quad-core system which, when threaded properly, should have no trouble processing higher frames per second, thus potentially increasing aiming accuracy of moving targets. Amongst other features, it includes multiple USB ports, for an easy camera connection, and Wireless LAN, for required wireless data transfer.

Figure 3.31: Raspberry Pi 3 Model B
(Permission Granted by RobotShop)

To compress the amount of subsystems in place and complete the requirement of implementing a PCB board, it was decided to combine the microcontroller with the PCB board. The selected microcontroller was the ATmega328p (Figure 3.27), which is used on the Arduino UNO board. The driving factors for this selection were the very low price of this component (inclusive that it comes in pairs of three's - possible backups) and that the amount of pins included with this microprocessor is sufficient for the number of inputs and outputs required by the robot.

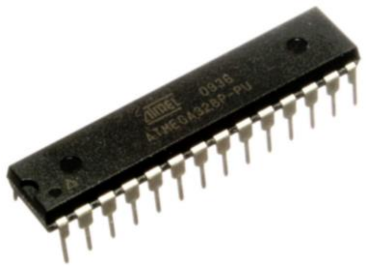


Figure 3.32: ATmega328p
(Permission Requested from oomlout)

The motor driver seen in Figure 3.28 was chosen because of its ability to handle the robot's estimated power system. The other motor drivers that were considered were found not to be able to handle the amount of current necessary for the motors in the system.

Figure 3.33: 10A 5-25V Dual Channel DC Motor Driver
(Permission Granted by RobotShop)

As previously discussed, encoders can be found packaged with gear motors, which simplify purchasing and are more affordable as a bundle. The selected gear motor, the NeveRest 40 Gearmotor (Figure 3.29), is one such gear motor that comes with an attached encoder.

Figure 3.34: NeveRest 40 Gearmotor with attached encoder
(Permission Requested from AndyMark)

It is not intended for the camera to be rotating a full 360 degrees, but to pan slightly left and right to accommodate the change in direction of the robot (so long as the robot does not turn completely around). The Futaba S3004 Standard Servo Motor (Figure 3.30), based off reviews, appears to transition smoothly and its packaging already includes a bracket for mounting objects. This motor is also very affordable, at a little under $15.

Figure 3.35: Futaba S3004 Standard Servo Motor
(Permission Granted By RobotShop)

The stepper motor in Figure 3.31 was selected for its size relative to the torque it provides. This motor will be rotating the upper portion of the robot holding other components such as the Nerf-guns, camera, and rangefinder so it must be able to support the weight of those combined objects.

The Nerf-blasters pictured in Figure 3.18 and Figure 3.19 were selected because of their size, loading and firing style, rate of fire, and firing distance. These blasters are within requirement size of the robot without having to modify the weapons themselves.

The firing mechanism for these weapons have the assistance of electronic components. As long as the trigger is held and the blaster is fed ammo the blaster will keep on firing and, added, at a fast rate. Finally, the shot range for both weapons is more than 40 feet, which is the full length of the course.

3.17.2. Final Component Selections

Table 3.19 below contains a list of the main functional computer/electrical components that are a part of the final battlebot design.

Component	**Name**	**Quantity**
Image Sensor	Logitech HD Pro Webcam C920	1
Rangefinder	LIDAR-Lite 3 Laser Range Finder	1
Processing Unit	NVIDIA Jetson TK1	1
Microcontroller	ATmega328p	1
DC Motors	NeveRest 40 Gear Motor	4
Motorshield	10A Dual Channel Bi-directional DC Motor Driver	1
Motorshield	HiTechnic DC Motor Controller	1
Encoder	*Attached to NeveRest 40 Gearmotor	4
Nerf-blaster (Dart)	CS-18 N-Strike Rapidstrike	1
Nerf-blaster (Ball)	Rival Khaos MXVI-4000	1

Table 3.22: List of main computer/electrical components selected

A comparison of the components that were seen in section 3.17.1 with the components that are seen in this section reveals that the following components were:

Removed:

- 3V 1.7A 68oz. Stepper Motor (2)
- Futaba S3004 Standard Servo Motor
- Raspberry Pi 3 Model B
- Nerf Rival Zeus MXV-1200
- ATmega328p (1 of 2)

Added:

- NVIDIA Jetson TK1
- HiTechnic DC Motor Controller
- Rival Khaso MXVI-4000

Refer to section 5 Hardware for more detailed information about the components that are listed above.

3.17.3. Part Acquisition

Fortunately, since the proposed system's design has been sponsored by Lockheed Martin with an overall budget of $2,000, we are able to pick out quality parts for the build of our system. We chose to search for parts only from reliable retailers, since we wanted to guarantee that the parts would arrive on schedule. Listed below are all the retailers we have acquired components from thus far for the proposed system.

3.17.3.1. Amazon

Although Amazon is known for many home products, Amazon also carries many robot parts for a reasonable price. Amazon also provides free two-day shipping for Amazon Prime customers, which allows us to stay ahead of schedule with the development of our project. With a wide variety of products, reasonable prices, and fast delivery, Amazon has become our top contender for part acquisition.

3.17.3.2. Robotshop

Another retailer that carries a very large supply of robot parts is the Robotshop. The Robotshop is a very well-known supplier for robot parts. The website URL was provided in the prompt of this project as a great guide for research and part acquisition. Although the Robotshop prices are often not as low as Amazon's, Robotshop does provide sales at certain times of the year, which our team has already taken advantage of. Overall, the Robotshop has been our team's main source of knowledge for researching different parts that are currently on the market in our price range.

3.17.3.3. Sparkfun

The third main retailer that we have used to acquire parts for this project is Sparkfun. Sparkfun has similar pricing to Amazon, but does not offer free-two day shipping as an option. However, Sparkfun has a wider variety of robot parts than that found on Amazon. Therefore, we have used Sparkfun for certain parts that could not be obtained from Amazon or for parts listed on Robotshop that are priced too high.

3.17.3.4. Dimension Engineering

Dimension Engineering has only been utilized for some specific parts that cannot be purchased at other retailers. For example, the 5V switching regulator was a piece that was specifically developed by Dimension Engineering. Therefore, the only place to order this product was on their website. This was the best regulator that was found within our price range for the proposed system.

4. Related Standards and Design Constraints

In this section, there are several different standards that will be used in order to make this project successful. There are some related standards out in the market, it is important to compare these standards in order to be able to prove that the project is safe, usable and stable. This section will explore the different types of standards that are related to our project as well as discuss the possible design constraints that this project may face.

4.1. Standards Research

For related standards research, there are plenty of standards that can be compared to this project. In this section, one standard is researched through details and analyzing so there is a better sense of how to read and use a standard and a table is provided for the other standards that can be researched and used.

The first related standard for our project is an IEEE standard denoted “1873-2015 - IEEE Standard for Robot Map Data Representation for Navigation.” One of the methods that the battlebot robot in our project will implemented is the use of 2D mapping. Therefore, this related standard can be used and compared to the robot project. Although this topic is very broad, the standard provides a scope of what exactly is being standardized. This standard is used for defining terminologies related to 2D robot maps of navigation in indoor and outdoor environments. It also specifies a data model for each element and defines a format used to exchange this data among other computers and robots.

The importance of this standard is to set a common representation for robot map data in a way that can allow an accessible use of software exchange among other robotic systems. This standard is used in order to expand the range of application and operational use among robots. There is also a section of definitions containing word definitions and mathematical definitions in order for the reader using this standard to understand the terms being presented as well as define specific words that makes following the standard less confusing.

There are several tables in the standard, labeled M/O which defines how often a variable or element is to be used. M is for mandatory and O is for optional. There are also detailed descriptions for how to format the data between the different type of mapping such as metric and topological mapping.

The standard then describes the basic mathematical models that shall be used in representing an environmental map as robots navigate and then goes into defining specific data formats are introduced. There are a lot of graphs and tables to follow, but it is important in being able to make 2D mapping definable and if every 2D mapping system uses this, the compatibility of sharing this data set among other systems will be able to advance forward in the technology. This standard sets the language for data to be transferred among other robot systems with XML. The standard explains that XML is a platform-independent language, many operating systems use this language and stores the files in human readable format. Table 4.1 contains other relatable standards with a scope, the publish date and a short description.

Standard Number	Scope	Publish date	Description
IEC 62680-1-2 Ed. 1.0 en:2016	USB	21 Oct. 2008	Universal serial bus interface for power and data.
CISPR 14-1 Ed. 6.0 b:2016	International Electrotechnical Commission	22 Nov. 2016	Requirements for household applications, electric tools and similar apparatus that uses DC motors.
802.11-2007	Wireless communications	12 Jun. 2007	IEEE standard for information technology, telecommunications and information exchange between systems.
1118.1-1990	Microcontroller	31 Jan. 1991	IEEE Standard for microcontroller system serial control bus
P2700/D1.00	Sensors	12 Aug. 2014	Standard for sensor performance.

Table 4.1: Table of related standards and their descriptions

Each standard follows the same format as to what needs to be included and implemented. After looking through one standard in detail, there are several related standards that can be considered and followed in the same manner. Since this project contains several features and components that will be used to make the robot functional and each component can follow this standard.

4.1.1. Design Impact

The standards researched and found in Table 4.1 above, although relevant, will not have a great impact on the overall design of the Nerf-enabled Battlebot system. This is due to a number of reasons.

The computer algorithms for autonomous detection will be built upon open source libraries such as OpenCV. This means that a bulk of the algorithm development process will have already been predefined.

Connections between system components will not go beyond the scope of what has been specified in the corresponding instruction manuals – specific pin connections will not be altered (regarding the microcontroller) and USB connections will only be plug and play (such as the USB connection to the camera). Vision and range finding sensors will not need to be adjusted as their default operations should provide sufficient enough data for the requirements of this project.

4.2. Design Constraints

There are several design constraints of the robot that played a role in determining the project's feasibility. This chapter analyzes each type of constraint that this project may contain from each area of the design. These constraints include consumer, ethical, environmental, power, weight, economic, time, and safety.

4.2.1. Size Constraints

An important constraint affecting the design of the robot is its dimensions must not exceed 3ft x 3ft x 3ft. This is a specification imposed by Lockheed Martin, in order for the robot to participate in the Battlebot competition. For this reason, special considerations had to be made with the evaluation of electrical components for the system. Also, a custom platform was designed by the mechanical team to ensure its volume were efficiently used and properly enclosed the components.

Minimizing the dimensions of the robot to meet constraints was a job that began with the selection of the electrical components. We had to stray away from choosing desktop-sized computers to handle the onboard processing. Components such as the motherboard, power source, and hard drive would require an enclosure that would put our robot near the limit. Instead we focused on finding the most powerful SoC (System on a Chip) computer that would be a fraction of the size of a desktop.

Another volume saving driven customization was a modification made to the Nerf-Blasters. The Nerf-Blasters were stripped of their original housing to be included on the turret. By stripping the housing off the Nerf-Blasters, and only keeping the bare parts needed by the shooting mechanism, the volume of each Nerf-Blaster is reduced by 5 inches in height and two inches in width.
Another component affected by the size constraint was the camera. The camera had to be near the guns to facilitate the aiming mechanism, but high enough to capture as much of the scene as possible without being blocked. Thus the camera was mounted at an elevation of about 20 inches above the ground with the Nerf-Blasters sitting on either side.

4.2.2. Ethical Constraints

Ethical constraints apply to this project since there will be a competition between multiple groups. The first ethical constraint that must be abided by is to engage in friendly combat with other enemy battlebots. This means that any countermeasure used to destroy one of the enemy team’s robots can be considered unethical. A few examples of this practice could be the use of ammunition other than nerf balls or nerf darts, setting the enemy’s robot on fire during the competition, or even purposely manipulating the enemy’s robot before the competition.

Another form of a countermeasure that may be considered unethical is somehow tampering with the enemy’s capability of sight. This means that blinding the enemy robot’s vision may be considered unethical, since the enemy team will not be able to demonstrate their working algorithm accordingly. A way to abide by such ethical constraints is to leave out any bright lights in the build of the robot so the competition is fair for all individual participants.

Since every group participating in this project is being given similar objectives, there is a chance that work from other groups will be fairly similar to the work provided by the Red Team. Although hearing information from other groups can be considered a fair practice in this competition, it would be unethical to copy work directly from another group. This can include code developed by an enemy team, research written by an enemy team, or even schematics drawn by an enemy team.

As more information is obtained from further research, it is imperative that all references and sources found of similar projects, guides, and information relevant to the project are cited at the end of this paper. Any information that is left uncited can be considered an unethical practice. Such ethical constraints can be considered standard for research papers that are developed.

4.2.3. Environmental Constraints

The environment is defined as the physical surroundings and conditions that will influence the performance of the design. The environment of this robot will be an indoors facility such as a conference room, located on either the premises of Lockheed Martin or of the University of Central Florida. The exact environment will be determined at a time closer to the competition date due to a possible conflict with the reservation of a room.

The uncertainty of the course means it is necessary to develop a robot capable of operating smoothly under different conditions. Conditions such as flooring can have an effect on mobility of the wheels, and accuracy of the encoders. Different sets of lighting can affect the vision algorithms which are based on tuning parameters to pixel intensity values.

Another condition set for the competition is that each robot must stay within the confines of its designated zone. The designated zones of each robot is a rectangular 10ft x 20ft perimeter set on opposite ends of encompassing 40ft x 20ft rectangular perimeter. Between the two zones is an obstacle zone, where robots are docked points for crossing into. This has influenced about every main component. The camera had to be chosen to provide a clear image of objects that would be meters in distance away. The Nerf-Blaster’s shots had to travel ranges without dropping. The range detector would be used to determine the likelihood a detected target is in an expected zone and not within its own zone or outside of the course.

A condition set by Lockheed Martin is to have live video of the robot’s point of view, with overlay of the automated target detection. This absolutely had an influence on our setup. Specific solutions had made just to transmit this amount data wirelessly, taking into account that an internet connection would not be made available to us. An important factor we must look out for is the possible interference between the wireless transmission protocol chosen and nearby electrical devices or the opponent's robots wireless transmitter.

4.2.4. Power Constraints

As this Nerf-enabled battlebot will be used in competition against other battlebots for two rounds, each ten minutes long, it is important to supply the battlebot with enough power to run the full duration of each round.

While it may not be necessary to provide the robot with a power supply strong enough to last two consecutive rounds (or 20 minutes), it would be beneficial to place the power supply in an area that is easy to access and modify. In this manner, the battery could be swapped out in-between rounds, reducing the minimum power usage length by 50% or ten minutes.

The weight and size of the power supply is also a limitation. It must be light enough for the chassis of the robot to support so that the robot can move about and do so without also wasting power. Depending on how power is regulated, additional power could be drawn to the motors controlling the wheels if the wheels need more torque in order to move the robot’s own weight. The power supply must also be small enough to fit the size constraints of the robot.

Positioning of the power supply is another factor to consider. The power supply unit must be placed on the robot in a such a way that it does not obstruct other components and/or wiring. It would pose a threat if the power supply was restricting the movement of a mechanical component (e.g. a rotating platform for the camera). It may also be necessary (if heating becomes an issue) to provide spacing enough for an open area to account for any possible heat transfer methods such as open airflow for a fan and heatsink.

4.2.5. Weight Constraints

One of the constraints that the team decided to follow for this project is to consider the weight of the robot. Listed in section 2.3, there are requirements listed for both size and mobility of the proposed system. The size constraint, S1, states that the robot dimensions shall not exceed 3ft x 3ft x 3ft. Keeping this size in mind while also considering requirement M1: being able to remotely control the Battlebot, it is very important that weight becomes a constraint to be considered throughout the course of the proposed system’s design.

Weight indeed becomes a constraint for both extremely light and heavy designs. However, for the proposed system, it is more likely that a heavy weight limit will be a more pressing concern. If the robot is too heavy, the mobility of the robot may be sacrificed. The robot may not be able to move and dodge enemy attack with weight slowing down the robot’s movements. The robot’s weight may affect the motor’s ability to move the robot. More power may need to be drawn in order to successfully move the motors at the speed necessary to move the system.

The proposed system’s overall weight is not the only concern that must be considered. There are constraints as to the equal distribution of weight throughout the proposed system design. If the system does not have an equal distribution of weight, certain parts of the system become more vulnerable. This may lead to the robot tilting a certain way, or even cause the robot to topple over. With the scope of this project being showcased at a competition, there leaves little room for error of weight design flaws in the system. Weight for every component then becomes a major consideration as well as where these components are placed throughout the proposed system.

The equal distribution of weight does not only apply to the Electrical team’s components and where they are placed. Major consideration must be done as to how the Mechanical team decides to design the robot. If a certain part is their design is already equally distributed, the Electrical team may not be able to place a component where originally desired. One the other hand, if a certain part of their design is not equally distributed, there may need to be considerations done as to whether the Electrical team should change its original placement of a sensor or whether the Electrical team should move their part to compensate for the unequal distribution of weight.

Overall, every part of the proposed system’s design has weight constraints that must be considered in order to build a successful system. Every piece must have its weight taken into consideration for an organized approach to solving the proposed system’s presented problem. Without such considerations being taken into play, the system could have mobility issues, could potentially topple over, or could suffer from lack of an equal distribution of weight. Weight is one of the crucial constraints that cannot be ignored throughout the course of this project, since it is one of the leading forces that affects development of the project design.

4.2.6. Economic and Time Constraints

The battlebot will span a design and development time of about 30 weeks - from September, 2016 to April, 2017. This is calculated from the length of a typical school semester whereas this project will span two semesters inclusive a one month break in between.

Within this timeframe, the battlebot will need to meet other time constraints, such as the competition between all involved teams and their battlebots as was the intent of the sponsor, Lockheed Martin. This competition is undated, but is assumed to be set for early April. It is also around this time that the project will need to be presented to the school panel for an assessment of the project design and performance and if it is worthy enough to pass the Senior Design course.

Financially, the sponsor has limited project expenses to a maximum of $2,000 of which only $1,000 can be used for the final battlebot design. This leaves at least $1,000 towards component backups and testing.

4.2.7. Safety Constraints

Since the robot will be showcased at the competition and at the senior design fair following the project’s development, there are certain safety constraints the Red team members must abide by in order to ensure that the robot will not be harmful to individuals interacting with it.

One safety issue that will be considered is the possibility of the robot firing at individuals as soon as the robot is turned on. In order to compensate for this issue, a safety constraint will be put in place that will allow the user to switch the robot between a “friendly” mode and an “attack” mode. The “friendly” mode will allow the user to manually navigate the robot without using the autonomous aiming, firing, and detecting features that will be present on the robot. This will ensure that when showcasing the robot, others will not be harmed.

Another safety issue could be using hazardous materials in the design of the system. This can include using an ammunition other than nerf darts or balls, sharp objects present on the robot, or not doing proper checks for faulty wiring. It is important that all team members abide by these constraints to prevent harmful interaction with the user. This will ensure that the robot can be used appropriately for its intended purpose within the scope of this project.

While choosing a laser rangefinder, it is important to keep in mind safety concerns that may arise. Some laser rangefinders can be damaging to the eyes, so it is important to make sure that these rangefinders are not pointed directly at someone’s face. However, the rangefinder chosen for this project, Lidar Lite 3, has a protective capping that may prevent such damage to one’s eyes. Therefore, individuals participating in the project do not have to worry during testing and individuals that will be interacting with the final product will not be harmed.

With the proposed system firing off many rounds of both nerf balls and nerf darts, it has been decided by Lockheed Martin that protective eyewear should be worn throughout the competition. All team members will need to utilize this protective eyewear in order to ensure safety among each group. This will prevent any stray nerf balls or darts from coming into contact with one’s eye, which may cause serious lifelong damage as a result.

5. Hardware Design

Chapter 5 contains the procedure for the hardware design necessary to power and operate the robot. It is separated into different subsections that will cover the power, the microcontroller design, the sensors, the turret and various motor shields that will be used.

5.1. Power for Preliminary Design

Power is the most important factor of the robot. Without power, the robot will not be able to detect objects, navigate around the field or fire the Nerf-blasters. The entire robot is planned to be powered by a single 12V battery. Therefore, the power must be able to distribute to all the different components of the robot. There are motors that will need direct power from the battery while some sensors will be powered through the microcontrollers and Raspberry Pi.

The following components that will need to be powered by the 12V is found in Table 5.1. Powering Components. This table shows the component that needs to be powered with power consumption specifications.

Component	**Quantity**	**Current (A)**	**Operating Voltage(V)**	**Mostly Off/On**	**Power(W)**
Microcontroller	2	0.0465	5	ON	0.466
DC motors	4	1.1700	12	OFF	57.60
Stepper Motors	2	1.7000	3	OFF	10.2
Servo Motor	1	2.0000	6	OFF	12
Nerf-blaster	2	-	6-9	OFF	-
Raspberry Pi	1	1.2000	5	ON	6
				Total Maximum Power	86.266

Table 5.1: Powering Initial components

The total maximum power is if everything in the robot was turned on at once. There are no indications or planning where everything will need to run at once but this is for consideration how much battery power the robot will require to run a total of two 10 minute rounds.

Since the battery is 12V and some components only require a lower voltage, there will be a use of voltage regulators to step down the voltage. The use of fuses and diodes will also help in preventing overloading the components as well as preventing voltage feedback into the power supply.

Figure 5.1 shows a diagram for the power flow that will be used to make sure each component can share and use each power safely and with stability. The 12V battery will be connected to each device so a high mAh rating will be used to ensure the system can stay running for longer than 20 to 30 minutes without charging.

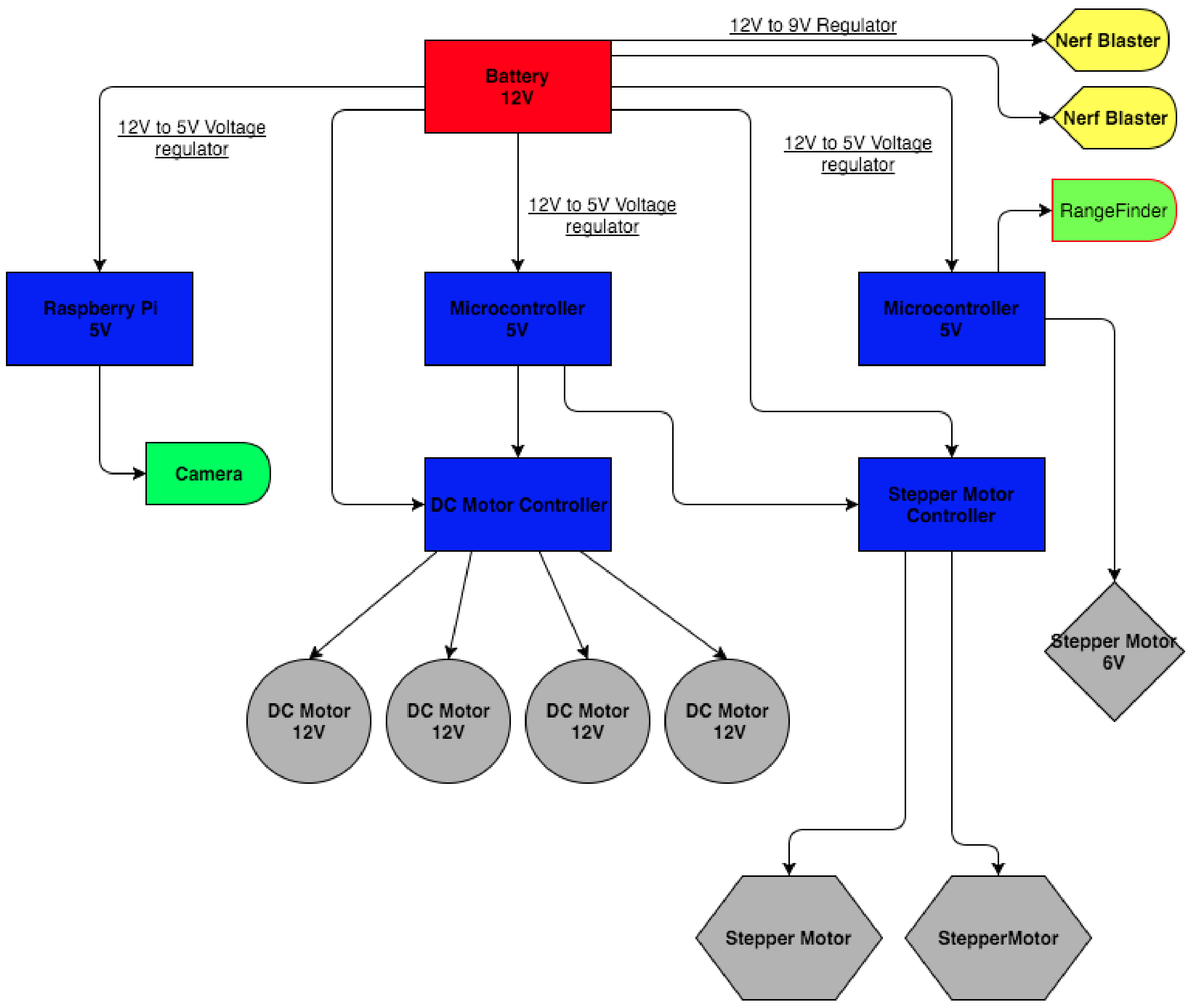


Figure 5.1: Power Flow diagram
(Team Designed)

5.2. Power for Final Design

As components changed from the preliminary design of the robot, so did the power distribution calculations. In order to select a battery for the proposed system, it was important to find the right capacity that would power all of the components at the correct ratings. Table 5.2 lists the power calculations for the final design of our system.

Component	Quantity	Current (A)	Operating Voltage(V)	Mostly Off/On	Power(W)
Microcontroller	1	0.0465	5	ON	0.23
DC motors	4	4.800	12	OFF	57.60
Stepper Motors	2	1.7000	3	OFF	10.2
Nerf-Blaster (B)	1	1.5000	6	OFF	9.0
Nerf-Blaster (D)	1	1.5000	9	ON	13.50
Jetson TK1	1	2.5000	12	ON	30.000
LIDAR Lite	1	0.130	5	OFF	0.65
				Total Maximum Power	110.98

Table 5.2: Powering Final components

Most of the current draw from our system comes from the DC motors that are used for manual navigation and the pan and tilt of our turret system. Therefore, to provide sufficient amount of current, a sealed lead acid battery was chosen. Sealed lead acid batteries contain the best chemistry type for high current and does not require complex charging. The downside to the sealed lead acid battery is its heavy weight and bulky size which was accommodated for the design. In order to calculate the right battery capacity, Equation 5.1 was used. Therefore at 12V, 5aH this battery will be able to last for 15 minutes per full charge only costing the project $15 per battery.

*Battery Life = [Battery Capacity(mAH) / Load Current(mA)] * 0.70*

Equation 5.1: Battery Life Equation

Although one battery is enough to power the entire system for the length of the competition, the final design features two batteries for testing and demonstration purposes. One battery is used solely for the DC motors, which takes up most of the current draw from our system. The other battery is used for the rest of our components, including but not limited to our PCB design, the Jetson TK1, and the Nerf Blasters. Many switches were added to the system to minimize power consumption during testing from components with high Wattage. Also, the switches added convenience to the system, allowing components to be powered on and off quickly and efficiently.

5.3. Microcontroller

The microcontroller is the main central processing unit and will be able to communicate with other components. The robot's rangefinder, motors, and Nerf-blasters will be controlled through one microcontroller board. This section will go through the design and pinout of the ATMega328p chip

5.3.1. ATMega328p

The Atmel ATMEga328p is a 28-pin microcontroller with 20 I/O. There are 14 digital pins for input and output pins, which 6 of them are pulse-width modulation pins and 6 analog pins for input. Figure 5.2 contains the full pin out diagram for the Atmega168/328p microcontroller.

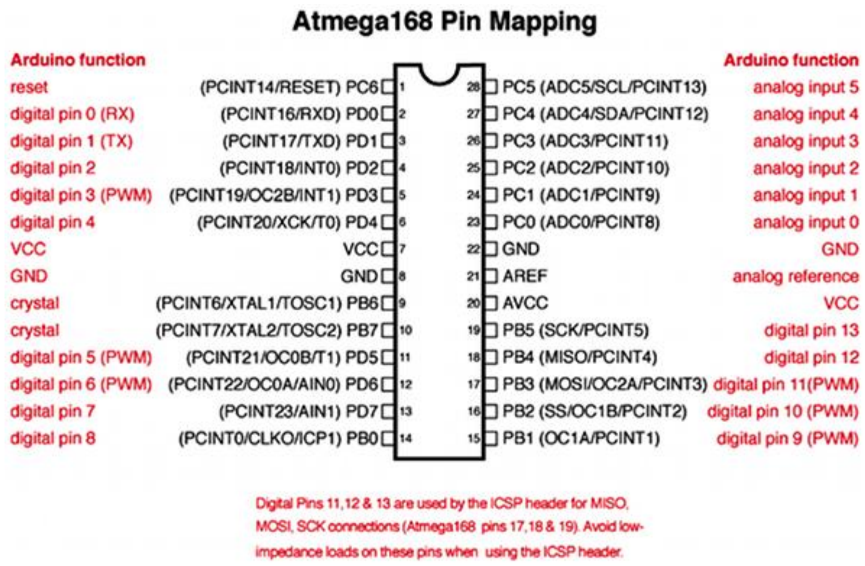


Figure 5.2: ATmega168/328 pin mapping
(Permission Granted by Arduino)

In order to supply power to the chip, Pin 7 also known for VCC will be connected from the main power supply using a voltage regulator. To supply a clock crystal to the device, the clock crystal will be connected between pin 9 and 10 using 22pF capacitors for stability. A tactile switch will be connected to the reset pin, PC6. There will be LEDs connected to the power and to pin 19 for debugging and troubleshooting to make sure the chip is powered properly. Table 5.2. Contains the label value of each pin and what component will be connected to the respective pin. This project is using two microcontroller chips with the same pin out configuration. This table assigns each component to a pin of the chip and will distinguish which microcontroller will go with which component.

Component	Number	Microcontroller Number	Pin Numbers (I/O, PWM)
DC Motor	1	1	PD2, PD3,
	2	1	PD4, PD5
	3	1	PD7, PD6
	4	1	PB0, PB3
Encoder	1	2	PD2 PD4
	2	2	PD7 PB08
	3	2	PB4 PB5
	4	2	PB3
LIDAR Lite	1	2	PC5
Nerf-blaster	1	2	PB5
	2	2	PB4

Table 5.3: Pin assignment of each component

The DC motors, stepper motor and servo motors all require a PWM pin and will not be directly connected to the microcontroller. Since each pin can only supply up to 500mA, the motor controllers will be used to drive each motor. The pinout to the microcontroller will still be the same, but the current and voltage will be supplied by the motor controllers. Lastly, for communication between other microcontrollers and for general programming, the ATMega328p will use a USB to serial breakout board and will receive power from the same VCC pin. The RX pin of the USB breakout board will connect to the TX pin of the microcontroller. The TX pin of the breakout board will be connected to the RX pin of the microcontroller. All components are expected to share a common grounding pin.

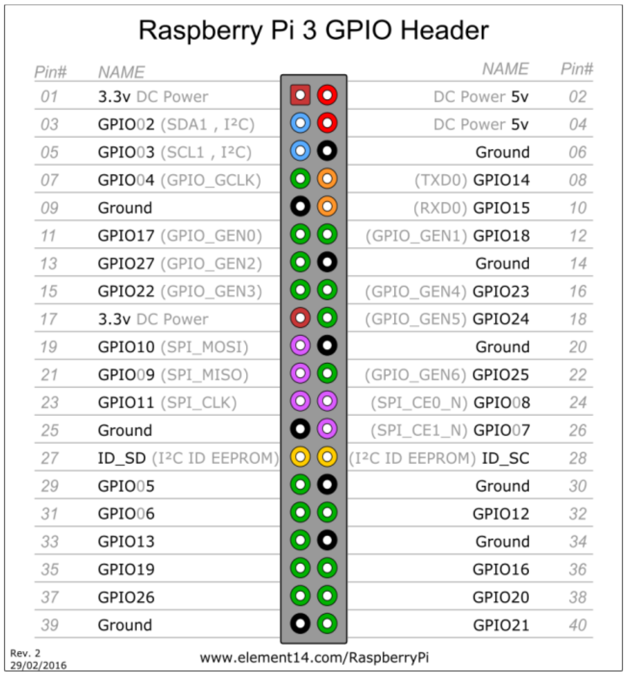
5.4. Microprocessor

The microprocessor will be used for object detection, image processing, and other computing algorithms for the robot and for controlling the robot's camera sensor.

5.4.1. Raspberry Pi 3 Model B

The Raspberry Pi 3 was originally the main unit for the automated object detection and image processing. The hardware required for the Raspberry PI to operate is an SD card with a preprogrammed operating system, external power supply connection, and a USB cord for communication between the microcontroller. The camera was to be connected to the Raspberry Pi through USB.

Figure 5.3 shows the full pinout for the Raspberry Pi 3 GPIO header is at full disposal in case more sensors or components will need to be added.



Raspberry Pi 3 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 2
29/02/2016
www.element14.com/RaspberryPi

Figure 5.3: GPIO pinout for Raspberry Pi 3
(Request Pending from Element14)

5.4.2. NVIDIA Jetson TK1

The final design for the nerf-enabled battlebot uses an NVIDIA Jetson TK1 board dedicated for computer vision processing and sending control signals to the microcontroller through Bluetooth, mainly manual navigation input and detected target coordinates.

The TK1 comes with 192 GPU CUDA cores and 2GB of RAM, of which are an importance for this robot because of the heavy image processing that it does. This is a vast improvement over the hobbyist level microprocessing board that was previously chosen, the Raspberry Pi, which in most cases is found to have a visible delay between image processing and a physical response.

The TK1, for the purposes of this robot, will not be used for its pinouts because the microcontroller will already be connecting all the other components. This also reduces the strain on the TK1 so that all of its resources will go to the computer vision algorithms. The added benefit is that failure of the TK1 will not cause failure of the entire system as other devices are capable of controlling the embedded robotic system.

This NVIDIA board is capable of consuming a large amount of power from the batteries as it is rated for 12V and 4.8A.

5.5. Sensors

The two types of sensors to be used for this project are the camera module and a range finder. These two sensors will be integrated into our chip since due to the time and money constraint, will not be designed by hand. This section explains how these two devices will be connected to the microprocessor as well as how each sensor can be mounted to the robot.

5.5.1. Lidar-Lite Laser Rangefinder

The Lidar-Lite Laser Rangefinder is simple to integrate to the robot system. To connect the Lidar-Lite to the ATMega328p chip, there is a 6 wire cable connector. Each pin of the cable connector contains a value. Table 5.3 shows the pin with the corresponding name value.

Pin	Name	Connection
1	Ground	GND
2	SDA	PC5
3	SCL	PC4
4	Mode	-
5	Power Enable	-
6	5V	5V Input

Table 5.4: Pinout for LIDAR-Lite

The pinout required for the LIDAR to communicate with the ATmega328p, pin 2 and 3 will be connected to two analog pins of the chip. Pins 1 and 6 will be connected to 5V power supply. The pins will be soldered into male jumper cables to avoid any of the wires from slipping out of the nodes of the microcontroller. The LIDAR will be mounted between the two Nerf-blasters to ensure precise range is captured. This way the Nerf-blasters will be able to fire the ammunition in the same direction.

5.5.2. Logitech HD Pro Webcam

The camera module will be mounted on the servo motor and will move independently of the turret and robot's turning directions. This is so that the camera can have sight of the targets at all times during the competition. The camera utilizes a USB cord which will be connected to the Raspberry Pi with pre-installed drivers to support it.

5.6. Aiming and Transportation

This section will be discussing the strategies of the movement of the robot and how the robot will be able to position the Nerf-blasters to fire upon targets. The robot will be utilizing motors to be able to move the Nerf-blasters in position to fire at targets, both moving and stationary. The following subsections go into details of how each will function.

5.6.1. Preliminary Design

The preliminary design section discusses the initial draft design of the battlebot's firing and manual navigation system.

5.6.1.1. Turret

In the initial draft of the battlebot it was decided to use stepper motors for the turret system:

“The turret will be driven by the two stepper motors for lining the Nerf-blasters at targets. The Nerf-blasters will be placed parallel to one another facing the same direction. With the two stepper motors, the first stepper motor will be placed so it can rotate the Nerf-blasters about the horizontal axis. The other stepper motor will be placed so it can elevate itself about the vertical axis. This allows a 2D frame for the Nerf-blasters to be aimed given the coordinates from the software. The Turret will be responsible for holding the extra ammunition that will be mounted on the Nerf-blasters in order for it to fire.”

This did not fare well in testing because of the actual weight of the turret system and the power that could be supplied through the stepper motor drivers..

To fix this in the time remaining for the project completion date, the stepper motors and drivers were swapped with the extra two DC motors that were originally cut because of budget. This process is highlighted in section 5.6.2.1.

5.6.1.2. Movement

The original manual navigation system called for 4 DC motors to drive the battlebot:

“The robot will need to be able to move across the field to scope out for targets to fire upon. There are maybe ways to drive a robot using 4 DC motors. The simplest and efficient design is to allow the four DC motors to be paired into pairs and implement a tank drive. This is done by driving one pair of wheels at the same speed and rotation and driving the other pair at the same speed. When it comes down to turning or rotation, one pair of wheels will move together in a different speed while the other pair will drive in an alternative speed to create the rotation.”

Modifications to this movement system are highlighted in section 5.6.2.2.

Figures 5.4 - 5.5 shows a visual example of how the robot’s tank drive would have worked.

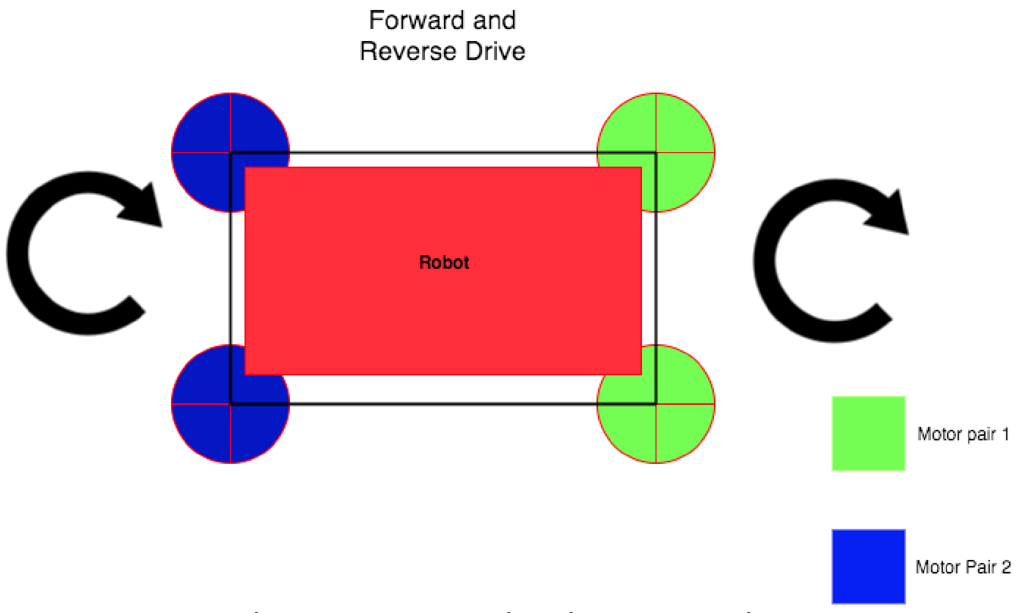


Figure 5.4: Forward and Reverse Drive
(Team Designed)

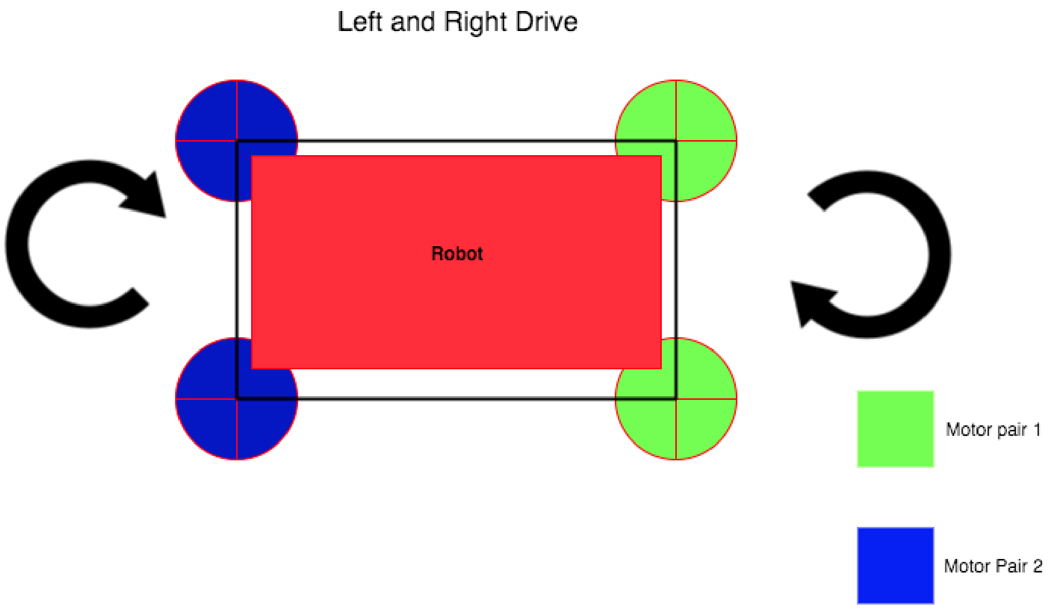


Figure 5.5: Left and Right Drive
(Team Designed)

5.6.2. Finalized Aiming and Transportation

This section discusses the Nerf-enabled battlebot’s final firing and manual navigation system.

5.6.2.1. Turret

As previously stated in section 5.6.1.1, the turret system switched from using two stepper motors to using two DC motors. The stepper motor drivers could only supply half of the rated power consumption of the stepper motors and therefore they could not compensate for the weight they were handling. Because of time constraints, it was faster to replace the stepper motors with DC motors than to wait on shipment of new stepper motors and drivers from the vendor.

DC motors, however, are not as accurate as the stepper motors. Luckily, the DC motors come equipped with individual encoders, thus effectively making them act as stepper motors. To do this, a compatible DC motor controller was used, the HiTechnic DC Motor Controller. The DC motors and their encoder cables attach directly to this motor controller. Then, the DC motor controller is connected to the microcontroller’s analog pins A4 and A5 (the I2C ports) where heavy software development was implemented to get accurate position control of the motors.

This controller was able to provide enough power to the DC motors so that they could move the turret system as flawlessly and as effortlessly as what was imagined with the original stepper motors.

5.6.2.2. Movement

Because of budget constraints due to upgrading the microprocessor from a Raspberry Pi to an NVIDIA Jetson TK1, a price increase of over $100, it was decided to remove two of the DC motors from the navigation system and replace them with simple caster wheels. The remaining two DC motors would move the battlebot using a differential drive method. To move backwards or forwards, both DC motors will turn in the same direction whereas turning left or right requires turning both DC motors in opposite directions (one forward and the other backward and vice versa).

This method balanced the budget at the cost of a lesser accurate navigation control.

5.7. Nerf-Blasters

The final step in the algorithm are the Nerf-blasters firing at stationary and moving targets. The two Nerf-blasters of choice will be stripped from its original case to properly mount onto the chassis of the robot. All the battery packs will be removed and connected to the main power supply. Each Nerf-blaster is operated with a pull of a mechanical trigger. Since these Nerf-blasters will need to be fired automatically, they will be taken apart and examined for the main switches that turn on the power to the blaster to fire nerf darts. Figure 5.6 shows the insides of a Nerf-blaster revealing all the mechanical parts and electrical wiring.

A Nerf-blaster consists of many parts both mechanical and electrical. If the user of the Nerf-blaster holds down the trigger of the blaster, the blaster will connect the circuit that allows the battery to connect to the motor and shoot the darts in multiple rounds until the magazine runs out of ammo. Therefore, the plan is to open up each Nerf-gun and discover what turns on the blaster and how to modify that to do so digitally.

Most Nerf-blasters contain a motor that drives the Nerf-darts through the chamber of the blaster and reloads the next dart into the barrel. If the mechanical trigger is pulled, the motors will activate by using the power of the battery to allow the mechanism to flow through.

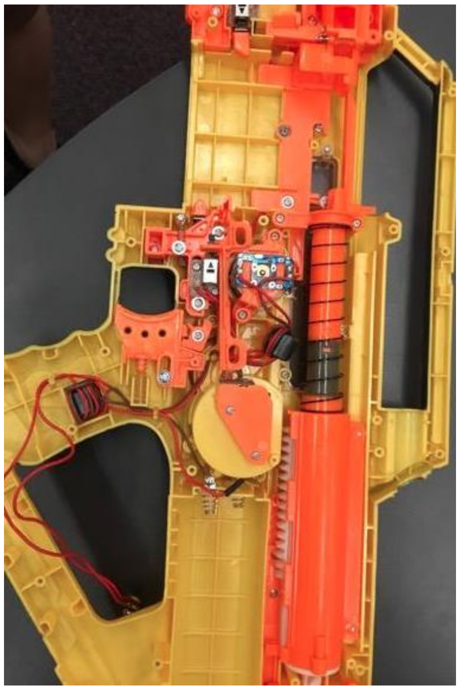


Figure 5.6: Inside of a Nerf-blaster
(Team Supplied)

Figure 5.7 shows a switch that sends the voltage to the firing mechanism to release the dart. This blaster has several mechanical switches that must closed for the blaster to fire the darts. The final switch that completes the full circuit is manually set on through the trigger of the Nerf-blaster. Since the robot must be able to also fire the Nerf-darts autonomously, the mechanical switch will need to be connected an I/O pin on the ATMega328p and will be programmed to fire. Although in order to achieve this feat, a MOSFET switching device will need to be implemented to turn on the Nerf-blaster and fire the darts.

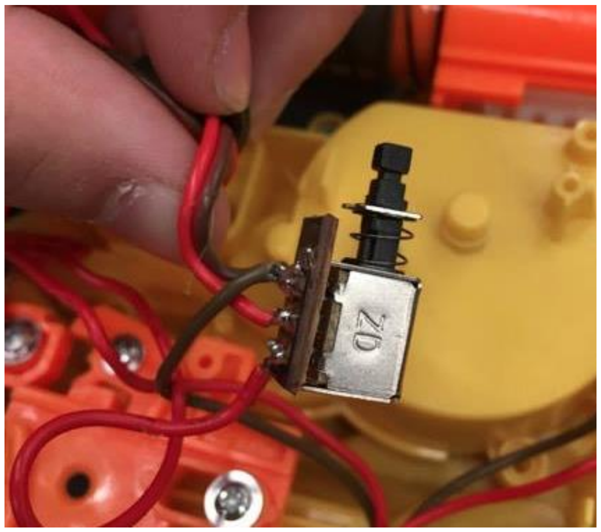


Figure 5.7: Mechanical switch of Nerf-blaster
(Team Supplied)

This mechanical switch in Figure 5.7 will be removed and replaced to operate digitally. The most common use of a digital switch in consideration are MOSFETs, which use voltage to control the voltage of another source (an example circuit is seen in Figure 5.8). The MOSFET gate source will be connected to a digital pin of the ATmega328p microcontroller and will allow the robot to remotely and automatically fire the Nerf-blaster.

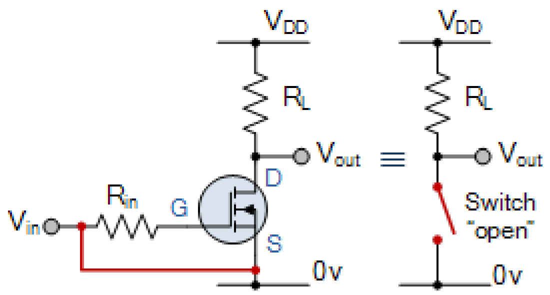


Figure 5.8: MOSFET as a switch
(Permission Requested from Electronic Tutorials)

This will allow easy programming for the Nerf-blasters to be fired from a digital voltage input. If the I/O pin is set to off, the Nerf-blaster will not be able to complete its circuit. As the MOSFET receives signal from the I/O pin, this will allow the switch to close, powering on the Nerf-blaster. The way the only way to turn off the Nerf-blaster from firing is if the I/O pin no longer sends a signal.

Another design idea for firing the Nerf-blasters digitally would be the use of a relay switch circuit. These devices are electromechanical and use an electromagnet that can operate a switch to open and close physically. It would only take a small amount of

power to operate the relay coil and could be used to control things such as motors, lamps or AC circuits.
The most common type of relay switch driven by a NPN transistor switch. Like most transistors, if there is no current supplied to the transistor, the circuit is an open switch and will not connect the power. Once the current and voltage is supplied to the transistor, the circuit will connect causing a flowing current from the battery to the component needed for powering.

With this device, it will be possible to controller the Nerf-blasters with a transistor and allow the output pins of the microcontroller to turn on and off the Nerf-blasters. The microcontroller will be programmed through the code and will be able to digitally connect the power of the Nerf-blasters to the firing mechanisms.

6. Software Design

The following Software Design section discusses the input and output, or dataflow, of the selected processes and components for each subsystem. This section also discusses how each subsystem communicates to come together as a whole battlebot system.

6.1. High Level Software System Architecture

In Figure 6.1, a software breakdown of the system components is visualized at a high level. This system is broken down into the subsystems: object detection and firing, panning and tilting control, navigating the robot, input and output data from the user, and processing via the microprocessor and microcontroller.
Discussion of these subsystems will not be as straightforward as they are illustrated. There are several data connections that will be webbing back and forth as certain components may have roles in multiple subsystems. For example, wireless remote control input from a user device, such as a laptop, will need to go through the processing subsystem before it reaches the manual navigation subsystem wherein the movement feedback from the encoders will be sent back to the processing subsystem for analysis.

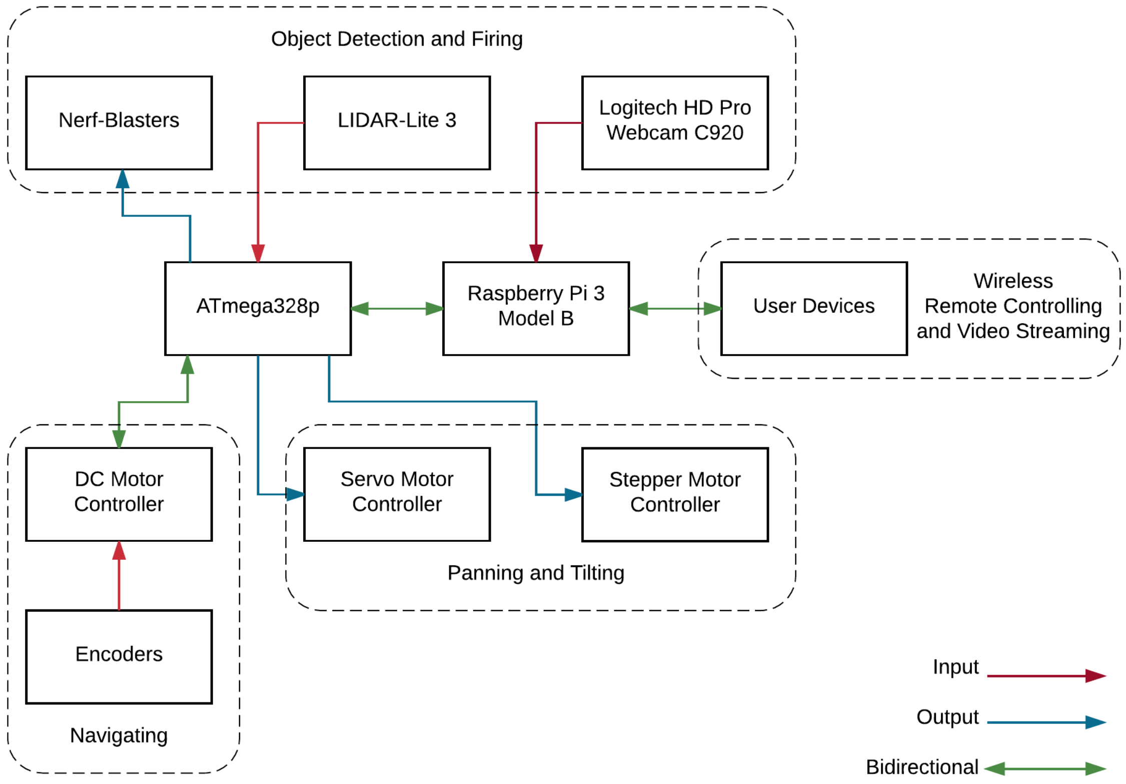


Figure 6.1: High Level Software Diagram (Team Designed)

6.1.1. Modes of Operation

To prevent the battlebot from attempting to target allies (by facial detection) immediately after being placed on the field and powered on or during repairs, two modes of operation will be configured - interchangeable by the push of a button - Targeting Mode and Neutral Mode.

Therefore, the high level architecture model found above in Figure 6.1 is true only when the battlebot is in Targeting Mode. This targeting mode is to be the primary mode of operation, where all aspects of the robot are fully functional.

The other mode of operation is to invoke a more neutral functionality of the robot. Only some subsystems, mainly the navigational system, of the robot will be running during startups, repairs, or other reasons, perhaps for defensive tactics to conserve power. Figure 6.2 presents the subsystems that will be active in Neutral Mode.

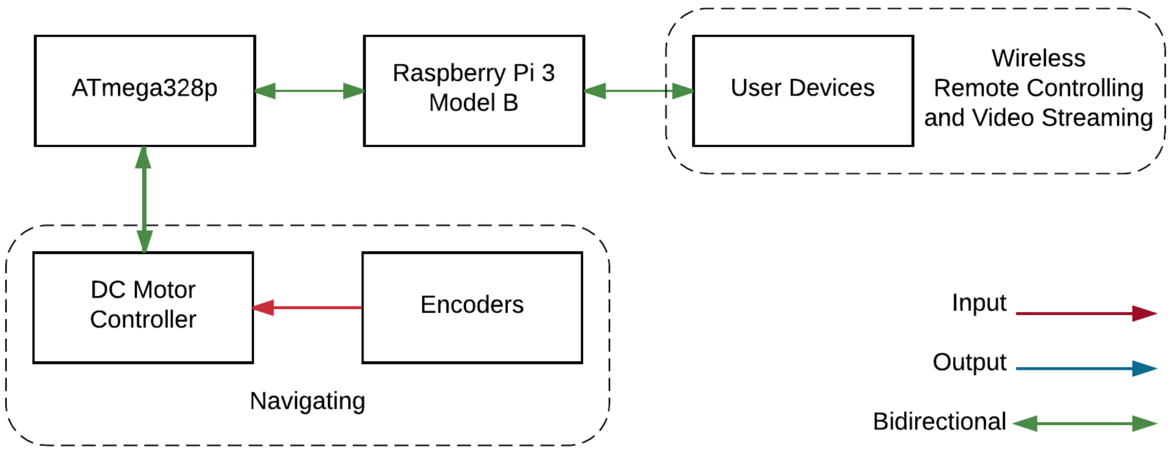


Figure 6.2: High Level Software Diagram in Neutral Mode
(Team Designed)

6.2. Software Development Life-Cycle

After much consideration for what type of Software Development Cycle is best for this project, it was decided that the Agile Development Life Cycle will be used. As opposed to traditional software development approaches, the Agile method consists of multiple iterations of analysis, design, development, and testing. Initial planning is kept generally high-level in order to allow for more iterations of the Life Cycle process.

Figure 6.3 below highlights the key stages of the Agile Method. Each iteration could last anywhere from a week to a few weeks at a time. These iterations can occur any number of N times, which allows for constant updating of the system until the final product must be delivered.

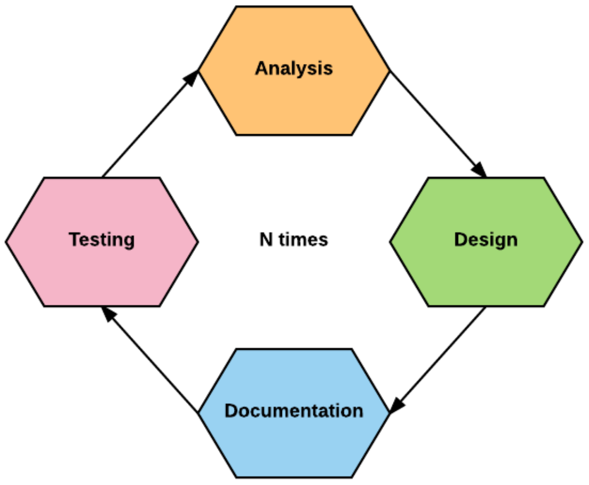


Figure 6.3: Agile Development Life Cycle
(Team Designed)

The reason the Agile method was chosen as the best software approach for this project is because the Electrical team will be working with two other groups; the Computer Science team and the Mechanical team. The Agile method allows the Electrical team to account for necessary changes that may occur throughout the course of this process. It also allows for constant up to date information between all group members in each team discipline.

A great example as to why the Agile method has been chosen for this project involves all members from each discipline. To explain, if the Mechanical team decides to change a specific part in their design that may affect the way the Electrical team chooses to program the Nerf-blasters, it will be imperative to make necessary changes to the proposed system as quickly as possible. This change may also affect the Computer Science team's algorithm which indeed can change the system design immensely. By constantly providing new software to ensure the system is constantly up to date for all team members is the best approach for this given project.

6.3. Sensor Processor

This section, Sensor Processing, will detail the setup that is required for the sensors. To explain in further detail, the data flow of the sensors readings is graphically displayed in Figure 6.4.

The Logitech C920 webcam will be plugged directly to the USB port of the Raspberry Pi. The C920 is a plug-and-play device, therefore it will automatically be recognized by the Raspberry Pi, assuming the operating system installed is Raspbian Jessie. The camera will be available to any Python code that needs to access it.

The Logitech C920 is a 15-megapixel webcam that is capable of recording video at 1080p. If this resolution proves to be too much for the system to process in real-time, we may need to downgrade it. Lowering the resolution can be done different ways. One way it can be done is through the terminal. Installing the fswebcam package on the Raspberry Pi allows us to access the device driver and directly specify the resolution. Another way of doing is through the OpenCV Python function VideoCapture. You create a "VideoCapture" object to receive video from the default camera. Then you can call the class method "Set" on the VideoCapture object with two arguments. The first argument is a flag to set the horizontal frame size with CV_CAP_PROP_FRAME_WIDTH, or vertical frame size with CV_CAP_PROP_FRAME_HEIGHT. Then the second argument takes an integer value as the resolution size.

Once the camera frames are being received they will each be analyzed by the autonomous target detection pipeline. With the camera feed alone, the system will not know its position in the field nor the distance to the objects detected. Therefore, the range finders and encoders must be frequently polled too.

The range finder we will use is the Lidar Lite v3. It will either be read as input directly by the Raspberry Pi or indirectly with the Arduino playing the middle role. The Lidar Lite can be interfaced via Pulse Width Modulation (PWM) or I2C interface. Using the Lidar Lite v3 Arduino library we can call the Lidar with a line of code. If the LiDAR is interfaced via PWM, then we can call GetDistancePWM() to read its distance measurement. If instead it is interfaced via I2C, then we simply change the call to GetDistanceI2C().

The last sensors we will use are the encoders embedded on the wheels. To read the encoders we will simply need to ensure the pins they are connected to are set as inputs. To do this we can use the Arduino function pinMode(). With the encoders pins set to inputs we can then use the digitalRead() function to take measurements.

Though the Raspberry Pi needed to be swapped for the Jetson TK1, the sensor fusion stayed the same. The Jetson TK1 used USB to interface with the camera, and the microcontroller interfaced with the LIDAR-Lite. The Jetson TK1 uses Ubuntu 14.04 and OpenCV packages to read frames from the camera.

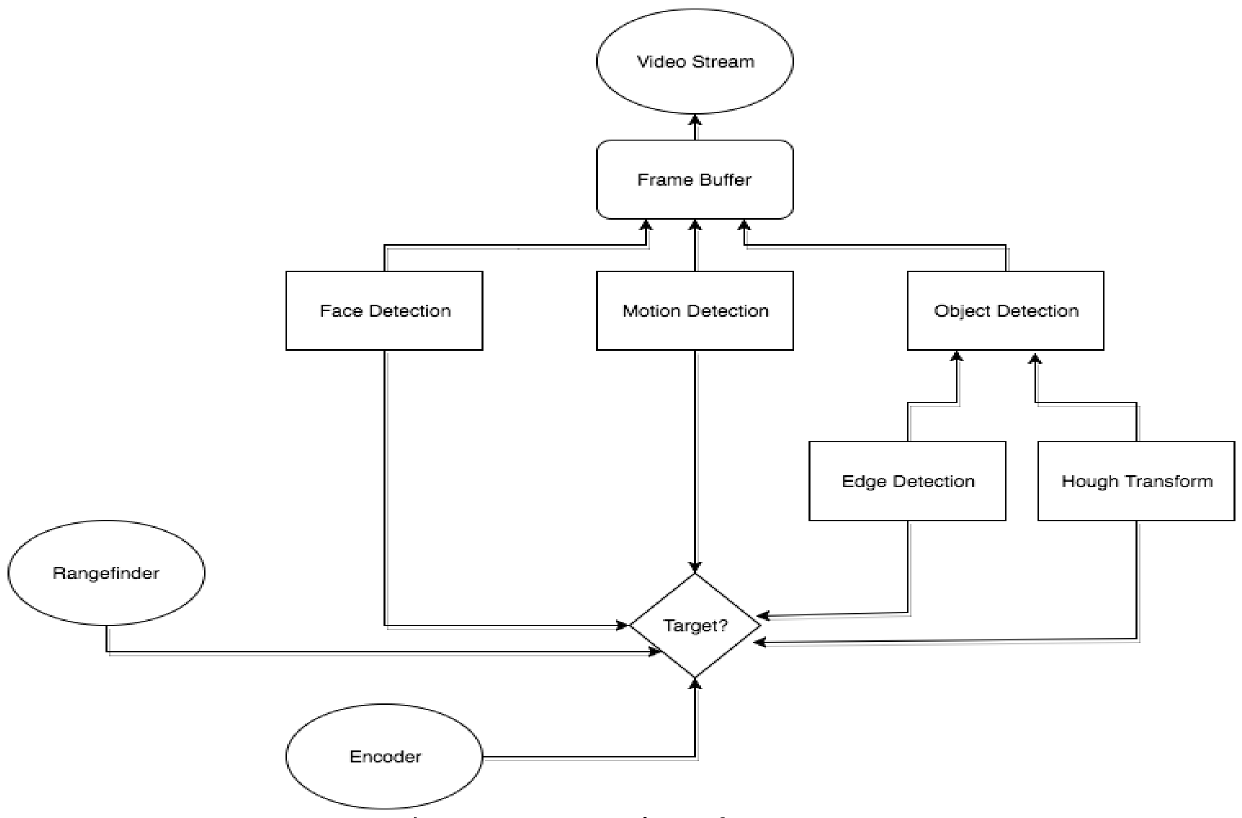


Figure 6.4 Data Flow of Sensors
(Team Designed)

6.4. Autonomous Detection

It has been decided that the autonomous detection for this project will be carried out based on concepts of computer vision. Although the Electrical team will not be responsible for the autonomous detection of the proposed system, the Computer Science team's autonomous detection drastically impacts the programming that needs to be carried out by the Electrical team regarding input and output from different devices and the moving parts of the robot. This section will briefly discuss the basic concepts that will be carried out by the Computer Science team as well as how these aspects will affect the Electrical team's coding plan.

The first concept that must be considered regarding the autonomous detection of the proposed system is the different aspects that will be present on the given field. An important aspect of the field to remember is that there will be both stationary and moving targets that must be autonomously targeted. Another aspect of the course will be the obstacles that must be avoided when shooting at a target. Since not all information of the course has been provided, the solution to the proposed system's problem becomes more intricate. In order to solve the proposed system's problem of being able to detect, aim, and fire at stationary targets and moving enemies, there will be multiple autonomous detection algorithms combined. The main algorithms that will be used to program the autonomous detection of the robot will deal with object recognition, motion detection, and facial detection.

The Computer Science team plans to use object recognition as a means to recognize certain features from the enemy robot's design and use these features in order to identify that object. Based off a database that already contains data, this data will be matched to the features being obtained from the camera. The information provided by the camera sensor will be programmed to match with the information provided in the database. If the objects are similar, the Nerf-blaster will be able to recognize the object as a "friend" or an "enemy".

Another key development in the programming of the robot's autonomous capabilities will be motion detection. Motion detection will allow the robot to find any object in motion within the frame of the camera. This will be a major aspect of the system's performance, since all the objects that will be in motion during the competition will be enemies. As the system identifies whether an object is in motion, it will be able to label an object an enemy automatically if that object is found to be in motion.

The final factor that will play an important role in the development of the autonomous capabilities of the proposed system its ability to use facial detection. Facial detection will be a major aspect in the design of the robot because information has been released that the stationary targets will have faces on them. With the robot being able to detect faces, the robot can automatically label this object as a stationary "enemy". Thus, the

Computer Science team will ensure that facial detection is a crucial aspect in their programming design of the proposed system.

The current plan for programming autonomous detection for the proposed system has a great impact the Electrical team’s decisions of the robot’s design and functionality. The image processing will be done through the Raspberry Pi, as the camera will be directly connected to the Pi. The Electrical team will be in charge of sending the output from the camera to the Raspberry Pi. The Electrical team will also be in charge of sending input to the motor controlling the camera so the camera will pan. This will allow the proposed system to search for these objects that will be recognized by the Computer Science team’s algorithm.

After some algorithm development it was decided the Raspberry Pi processing was not sufficient therefore the single board computer was upgraded to a Jetson TK1.

6.5. Manual Navigation

As mentioned in section 3.15, the programming regarding manual navigation of the proposed system relies heavily on how the data from the motors will be sent to the controller. It has been decided that a laptop will be used as the main source of controlling the navigation of the robot. As a secondary source, a separate controller, such as an Xbox controller, may also be programmed if it is possible during the time given to complete this project. Both controls will be operated wirelessly in order to complete a requirement given by the competition and to also allow for optimal use of the robot in battle.

Since different means of how the robot can wirelessly communicate with the laptop have been explored in section 3.15, it has been decided that the robot will preferably connect to a local network via a router in order to establish a connection with the laptop. This proves to be the most optimal option, since this allows for a sufficient range as the robot will be in the same room as the router and laptop during the course of the competition. This will also provide a reliable way of transmitting the live video feed to the laptop while being able to manually control the robot at the same time.

A program will be developed by the Computer Science team to house both the controls for the manual navigation and the video stream for the robot. The Electrical team will provide adequate connection from the motors to the PCB, which will house an Arduino chip used for processing the data for the manual navigation. Once connection has been established, the Electrical team will program the Arduino to control the motors.

The Arduino will send all the data to the Raspberry Pi in order to establish connection to the laptop and allow the Computer Science team to use a python program to interface with the data. If preferred, the Arduino may instead establish its own connection to the laptop if proven to be a better strategy for the proposed system. Finally, the Electrical team will be able to control the robot's movements wirelessly via the arrow keys or the WASD keys on the laptop for intuitive controls.

The first consideration for establishing manual navigation is the libraries that may need to be imported when transferring the data from the Arduino to the Raspberry Pi. It is likely that there will be a serial communication between the Raspberry Pi and the Arduino by simply connecting the two via a USB cable. One way to send the data from the Arduino to the Raspberry Pi is by simply importing the serial library to both the Arduino and the Raspberry Pi to assist in the transfer of the data. By calling the serial library and reading each line of data, the code written to the Arduino for the manual navigation can be transferred to the Raspberry Pi which will connect wirelessly to the laptop using a local network connection.

The Jetson TK1 wound up being the core processing unit used. There was serial communication with the microcontroller, like mentioned before with the Raspberry Pi but it was done using a Bluetooth HC-06 module.

Generally, four main functions will be established to control the direction of the robot. These functions are as follows below:

- void forward();
- void backward();
- void left ();
- void right ();

As one would assume, void forward() will provide forward movement of the proposed system, void backward() will provide backwards movement, void left() will provide movement to the left, and void right() will provide movement to the right. Other functions to consider could be a void setup() function or a void stop() function that will provide a general means of initializing certain devices required for manual navigation and allow the proposed system a safe and effective way to stop during the competition. The diagram in Figure 6.5 explains how these functions will be implemented in the proposed system.

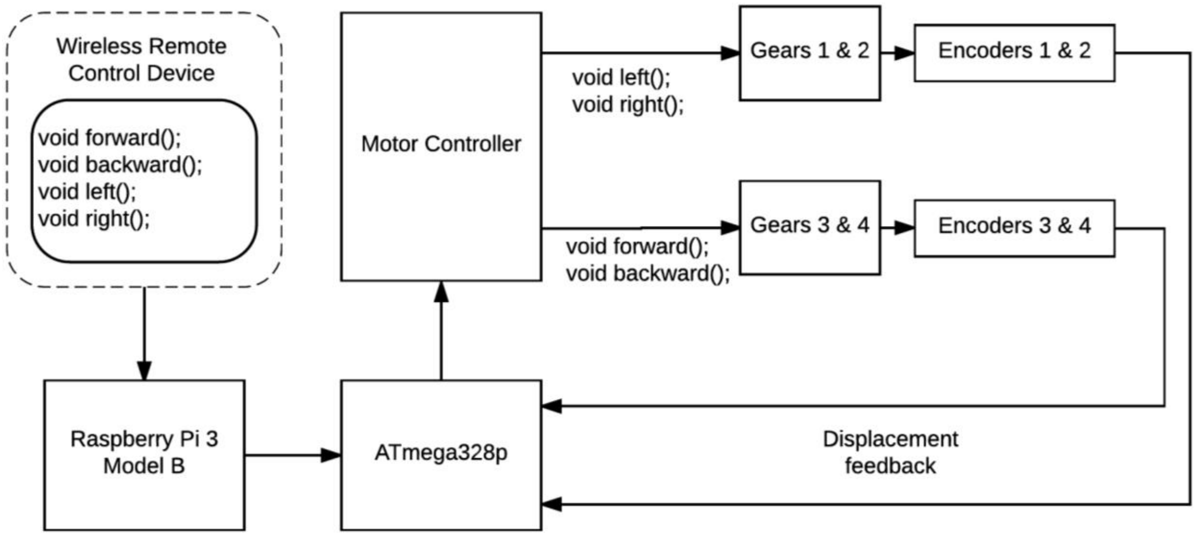


Figure 6.5: Manual Navigation breakdown
(Team Designed)

Defining output pins for the motors in the code will be extremely important to allow for successful manual navigation. These defined pins will give the motors the commands written in the functions mentioned above, allowing for the motors to be able to respond accordingly. Not only do the pins for the motors themselves need to be defined, but the pins for the motor drivers will need to be connected to the PCB for a smooth transition to the motors.

6.6. Nerf-Blaster

The programming of the Nerf-blaster and its movements are crucial to the functionality of the proposed system. The Computer Science team will be in charge of programming the Nerf-blaster to point and shoot in a specified direction autonomously. The Electrical team will oversee programming the motors to pan and tilt the Nerf-blaster as well as programming the Nerf-blaster to fire. This section will highlight the Electrical team's software plan for controlling the Nerf-blaster's direction and the Nerf-blaster's ability to fire.

Since there will be two Nerf-blasters for the proposed system, each blaster will be programmed accordingly to fire independently of one another. Even though the two Nerf-blasters will be able to fire at different times, the same motors will control the position of each blaster; one motor to pan and one motor to tilt. The two Nerf-blasters will be programmed to pan and tilt simultaneously to aim at a specified target. It has not been decided when each Nerf-blaster will fire. One idea that may be implemented is allowing the nerf dart blaster to fire at close range objects, since its accuracy is less than that of the nerf balls. This would allow the nerf balls to be solely used for long range objects for a potentially more accurate shot. These range values will be determined by the range finder, which will be positioned between both Nerf-blasters.

As discussed in section 3.5, both Nerf-blasters' mechanical switches will need to be connected as I/O pins to the ATMega328p. Therefore, all programming will need to interface through the Arduino in order to move the motors and fire the guns.

As the Electrical team breaks down the plan for coding the Nerf-blaster to pan, tilt, and fire seamlessly, the first thing that needs to be addressed is the specified range that a Nerf-blaster is able to pan or tilt. The general idea is to program the robot's pan range to be no larger than 180 degrees. This has been decided since it is ideal for the guns to be pointed towards the enemy at all times. Since it is known that the enemy will not appear behind us, the general field of view only needs to be at 180 degrees maximum. This will be a good starting point for further testing. As testing continues, this value can be improved and reduced accordingly.

Just as the range that the Nerf-blaster will be able to pan must be decided, so must the tilt range. The value for the tilt should not exceed 90 degrees. If the Nerf-blaster is allowed to tilt too high or low, it will completely miss its target as it will be pointed towards the ceiling or ground. Again, the exact value determined to be the most suitable for the proposed system can only be decided after further testing and implementation of the design. A complete breakdown all components that will be necessary in order to program the Nerf-blaster's pan and tilt capabilities are highlighted below in Figure 6.6:

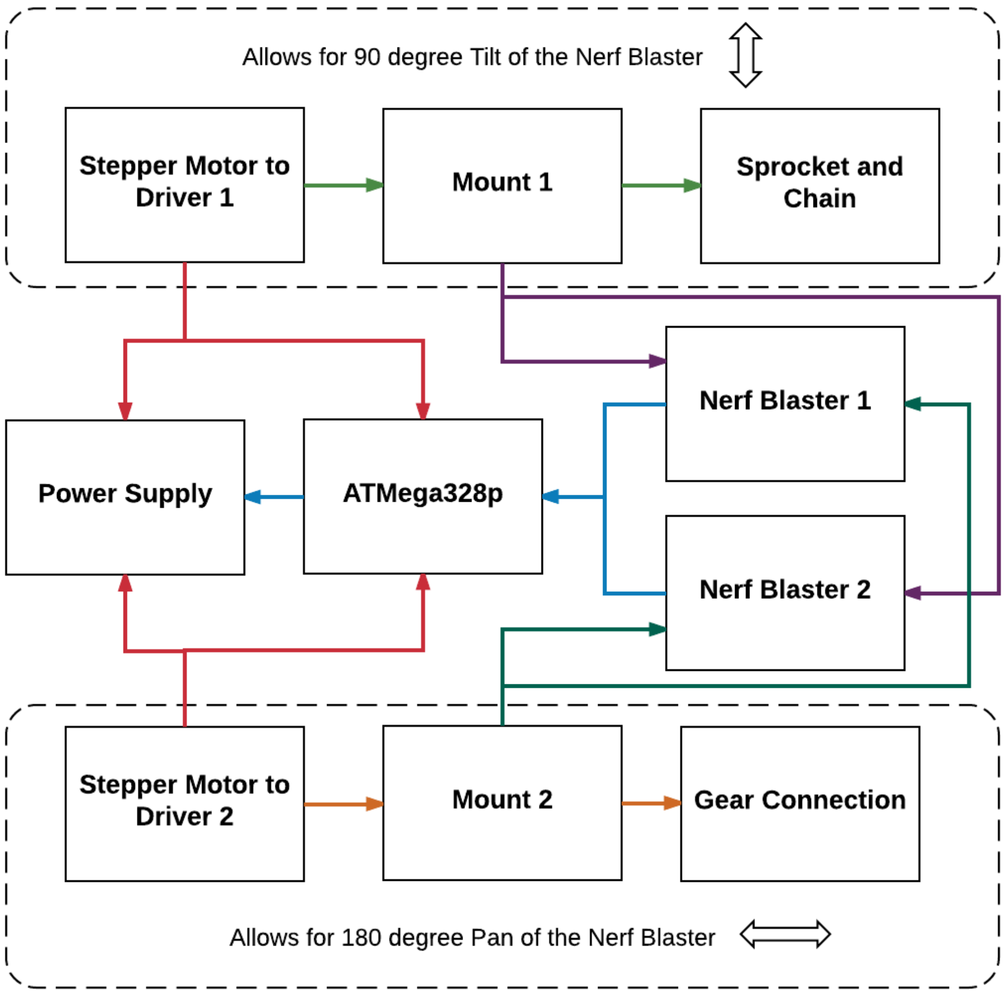

Allows for 90 degree Tilt of the Nerf Blaster
Stepper Motor to Driver 1
Mount 1
Sprocket and Chain
Nerf Blaster 1
Power Supply
ATMega328p
Nerf Blaster 2
Stepper Motor to Driver 2
Mount 2
Gear Connection
Allows for 180 degree Pan of the Nerf Blaster

Figure 6.6: Breakdown of Components for Nerf-Blaster's Pan and Tilt (Team Designed)

Another factor that must be considered regarding the movement of the Nerf-blaster deals with how fast to set the incrementation of degrees for both the pan and the tilt. This will be a constantly changing value determined by the speed of the moving target. This value can also be affected by the maximum speed that the motors will allow the Nerf-blaster to move.

It has also been determined that the camera, range finder, and Nerf-blasters will be automatically panning until a target has been found. This code that will allow for both Nerf-blasters to continuously move from left to right and must be developed in order to give the Computer Science team the opportunity to lock onto targets that are detected through the sensors. Once the robot is in “attack” mode, the panning will continue until a target is acquired.

One key functionality that may be utilized within the programming of the Nerf-blaster is serial communication between the robot and the laptop for testing purposes. Extra code may be written during this testing phase to make sure that the guns are working accordingly before the Wi-Fi signal is setup between the robot and the laptop. Adding this functionality may lead to less errors and less startup time between tests for better results.

6.7. ATmega328p

The ATmega328p chip will be relied on heavily because of the sheer amount of component connections that will be made to it. This chip will be connected to the Nerf-blaster firing mechanisms, range detection sensor, processing unit, servo motor controller, stepper motor controller, and DC motor controllers, which control the gear motors, that spin the wheels, and their respective encoders. Data will be coming in and out of quite a few components at once.

Video imaging from the Logitech HD Pro Webcam C920, the vision sensor (otherwise known as a camera), will be processing on the Jetson TK1 using object and facial detection algorithms. As these objects and/or faces are detected, the coordinates (relative to the center of the camera and where the object was detected) and a request to fire will be sent out to the ATmega.

When the request and its positioning data is picked up, the ATmega will then send out signals to the Servo Motor Controller and Stepper Motor Controller, which control the X and Y positioning of the firing system, to redirect the Nerf-blasters, camera, and the rangefinder towards the target’s position.

The LIDAR-Lite 3, or the rangefinder, will be continuously pinging distance values to the ATmega which in turn will be sending these values to the Jetson TK1 processing unit. The Jetson TK1 will continue to wait until the rangefinder signals back to it whether the target is within firing distance or not. If the target is detected and within firing distance, the Jetson TK1 will send out a message to the ATmega chip to signal the firing mechanisms on the Nerf-blasters to fire. If the target is not within distance, the Jetson TK1 will have a couple options depending on the algorithm that will be implemented by the Computer Science team. For instance, if the target is not within firing range, the Jetson TK1 may message the ATmega to reposition the stepper and servo motors to a default position, where the battlebot will continue to scan and detect objects.

While a textual explanation of this process might imply that the process could take several minutes, the entire target detection and firing process should occur in about three seconds at its slowest point (where the panning and tilting motors would go from leftmost to rightmost). Most of this time will be consumed while waiting for the stepper and servo motors to turn towards the target. This target detection and firing process by the ATmega, in junction with other system data, is mapped in Figure 6.7.

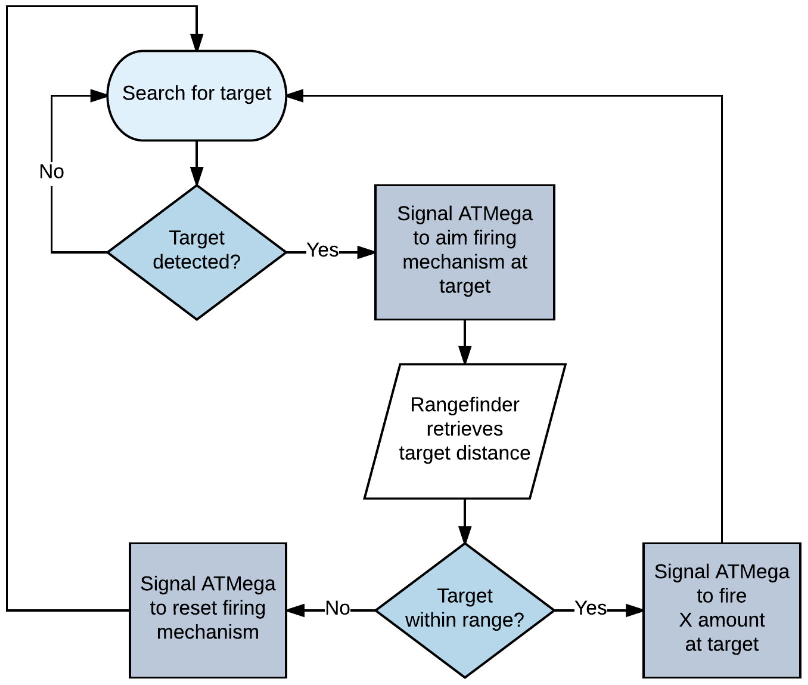

Search for target
No
Target detected?
Yes
Signal ATMega to aim firing mechanism at target
Rangefinder retrieves target distance
Signal ATMega to reset firing mechanism
No
Target within range?
Yes
Signal ATMega to fire X amount at target

Figure 6.7: ATmega328p response to firing subsystem seen as a flowchart (Team Designed)

Navigating the robot, by means of controlling the gear motors, will also be the responsibility of the ATmega chip. When a user sends navigational input via a wireless remote control, be it an actual remote control or the keyboard of a laptop, it is received by the Jetson TK1, however not physically processed by it. The Jetson TK1 delivers the input to the ATmega chip. It is then that the corresponding Motor Controllers (turning left or right and going forward or backward by any combination) receive the input and turn the gear motors as requested.

When the gear motors turn, the encoders that are built into the gear motors send feedback to the ATmega. The ATmega then delivers this feedback to the Jetson TK1 for data analysis. This feedback is information on how far the wheels of the robot have turned, which may prove useful for tracking distance traveled by the robot on a size restricted course. Movement restrictions could be set into place, such as prohibiting the robot from navigating further forwards if it means it would be crossing the course keep out area (crossing would lose points).

The ATmega chip, in short, will be acting more as a messenger than a data processor, even though it is fully capable of doing so (albeit limited in comparison to the Raspberry Pi). The ATmega will be taking input and output and delivering that data to the appropriate receiver.

Programmatically, the input and output data signals of the attached components will be configured and allocated by Electrical and Computer Engineering team in order for the Computer Science team to use for their algorithm design and processing.

6.8. Communication

The purpose of communication is for the microcontroller to be able receive motor commands sent by the Jetson TK1, and for the Jetson to receive Lidar readings sent by the microcontroller. The Serial protocol is used to implement the two way communication needed between the devices.

The hardware chosen to perform the job is the HC-06 Bluetooth module which is attached to the microcontroller, and an Intel Dual-Band Wireless AC-7260 hooked to the Jetson TK1, which provides both Bluetooth and WiFi connection.

To establish a connection between the two systems it was necessary to utilize the Bluetooth protocol RFCOMM on the Linux powered Jetson TK1. RFCOMM provided an API used to pair and connect the HC-06. A configuration file /etc/bluetooth/rfcomm.conf was made which allows us the option to create a serial device under rfcomm and to automatically bind or connect to the device. Once a connection between the Jetson TK1 and the HC-06 is made a serial port we can use is made.

AT commands were used to configure the port settings on the The HC-06. The Baud rate was set, the pair code was set, and the device name was changed to prevent any connection issues with other Bluetooth devices. The Jetson pairs the device via the Bluetooth GUI, and once the module has been paired the Jetson will not forget the HC-06. Connection between is finally established when the device file is opened by our software system.

Using the serial port on the Jetson is done by accessing the HC-06's serial device file. The device filename in Linux is /dev/rfcomm0, and the software operations on the device file are done using the POSIX terminal interface. This interface is needed because the software implementation is to be done in the C++ language and is standard to Linux systems. The advantage to using this interface is it allows many serial communication configurations to be made such as setting the baud rate, character size, parity checking, flow control, and timeouts.

The most important setting being the baud rate set 115200, which needs to be agreed upon by both ends. The character size was set to 8 bits to receive bytes on both ends. Also an important detail of the software implementation is the use of a timer before reading or writing to the device file, as opening it will cause the microcontroller to reboot. To receive data the Jetson will read data from the Serial port and store the data in a buffer. Buffer size must be specified therefore we chose a size that is twice the size of our formatted message length to avoid overflow. Similarly the transmitting of data is done using a write function.

The Arduino Serial library is used to establish communication on the microcontrollers end. Once the correct baud rate set, the microcontroller can both receive and transmit messages. The Arduino uses the Serial Available function to get the number of bytes it is being set. It then transmits to the serial port with the print function. Special care must be taken when reading this data as newline and carriage return bytes will be sent.

Messages are sent in a formatted method both ways. Therefore the Jetson has a format to follow and the microcontroller knows what to expect. Each message coming from the Jetson specifies the instantaneous control over the motors, thus the microcontroller needs to continuously poll the Serial line for the robot to be responsive. The microcontroller also sends data in a formatted method to the Jetson which only contains the lidar readings.

The benefits of choosing Arduino based microcontrollers are seen here as the Arduino Serial Library provides the programming functions to do this. Tables 6.1 - 6.6 are the functions available to initiate communication, read, and write data serially.

Function	begin()
Description	Sets the data rate in bits per second (baud) for serial data transmission.
Syntax	Serial.begin(speed, config)
Parameters	speed: in bits per second (baud) - long config: sets data, parity, and stop bits.

Table 6.1: Arduino Serial Library begin

Function	end()
Description	Disables serial communication.
Syntax	Serial.end()

Table 6.2: Arduino Serial Library end

Function	find()
Description	Reads data from the serial buffer until the target string of given length is found. The function returns true if target string is found, false if it times out.
Syntax	Serial.find(target)
Parameters	target : the string to search for (char)
Returns	Boolean

Table 6.3: Arduino Serial Library find

Function	findUntil()
Description	Reads data from the serial buffer until a target string of given length or terminator string is found. The function returns true if the target string is found, false if it times out.
Syntax	Serial.findUntil(target, terminal)
Parameters	target : the string to search for (char) terminal : the terminal string in the search (char)
Returns	Boolean

Table 6.4: Arduino Serial Library findUntil

Function	read()
Description	Reads incoming serial data.
Syntax	Serial.read()
Returns	The first byte of incoming serial data available (or -1 if no data is available) - int

Table 6.5: Arduino Serial Library read

Function	write()
Description	Writes binary data to the serial port. This data is sent as a byte or series of bytes; to send the characters representing the digits of a number use the print() function instead.

Syntax	Serial.write(val)
	Serial.write(str)
	Serial.write(buf, len)

Parameters	val: a value to send as a single byte str: a string to send as a series of bytes buf: an array to send as a series of bytes len: the length of the buffer
Return	byte write() will return the number of bytes written, though reading that number is optional

Table 6.6: Arduino Serial Library write

6.9. PID Controller- Arduino Library

In order to have a system of motors consistently functioning precisely as we desire, we will need to implement a PID Controller. The PID controller will help determine the parameters we need to continually adjust as we analyze our inputs or digital commands, and the corresponding motor output or physical movement. Tables 6.7 to 6.10 show the functions made available by the Arduino PID Library we will use.

Function	PID()
Description	Creates a PID controller linked to the specified Input, Output, and Setpoint. The PID algorithm is in parallel form.
Syntax	PID(&Input, &Output, &Setpoint, Kp, Ki, Kd, Direction)
Parameters	Input: The variable we're trying to control (double) Output: The variable that will be adjusted by the pid (double) Setpoint: The value we want to Input to maintain (double) Kp, Ki, Kd: Tuning Parameters. these affect how the pid will change the output. (double>=0) Direction: Either DIRECT or REVERSE. determines which direction the output will move when faced with a given error.

Table 6.7: Arduino PID Library - PID

Function	Compute()
Description	Contains the pid algorithm. It should be called once every loop(). Most of the time it will just return without doing anything. At a frequency specified by SetSampleTime it will calculate a new Output.
Syntax	Compute()
Parameters	True: when the output is computed False: when nothing has been done

Table 6.8: Arduino PID Library - Computer

Function	SetOutputLimits()
Description	The PID controller is designed to vary its output within a given range. By default this range is 0-255: the arduino PWM range.
Syntax	SetOutputLimits(min, max)
Parameters	min: Low end of the range. must be < max (double) max: High end of the range. must be > min (double)

Table 6.9: Arduino PID Library - SetOutputLimits

Function	SetTunings()
Description	Tuning parameters (or "Tunings") dictate the dynamic behavior of the PID. Will it oscillate or not? Will it be fast or slow? An initial set of Tunings is specified when the PID is created. For most users this will be enough. There are times however, tunings need to be changed during run-time. At those times this function can be called.
Syntax	SetTunings(Kp, Ki, Kd)
Parameters	Kp: Determines how aggressively the PID reacts to the current amount of error (Proportional) (double >=0) Ki: Determines how aggressively the PID reacts to error over time (Integral) (double>=0) Kd: Determines how aggressively the PID reacts to the change in error (Derivative) (double>=0)

Table 6.10: Arduino PID Library - SetTunings

7. Prototype Construction

This section is to show the process of design and assembling for the PCB of this project. There also explains some manufacturing company comparisons with displayed schematics of the PCB designs.

7.1. PCB Vendor and Assembly

This section focuses on the different types of PCB vendor to consider for this project.

7.1.1. Plan

As a requirement for this project, a printed circuit board must be designed, assembled and manufactured. This portion of the project will be completed through companies that can specialize in this area. With several companies in the market, this section briefly discusses the three companies of choice and their services offered.

One of the company that offers this service is Elecrow. They offer customized PCB manufacturing at a reasonable price. To acquire a PCB from Elecrow, Gerber files will be emailed to a specific project manager along with a bill of material and any PCB specifications and quantity size. Elecrow will bill the service based on the fabrication, the level of difficulty for soldering, the list of components needed and how many PCB's that will need to be made. The time claimed to finish PCB assembly and manufacturing will only take from 3 to 5 days.

Another company that can offer PCB assembly is Expresspcb. They offer services to a wide range of the different types of PCBs. The least expensive option is the Miniboard standard. These boards are a 2 layer pcb with no solder mask or silkscreen layers. They have a fixed size of 3.8 x 2.5 inches. They offer free software for schematic and PCB mapping. The files will be emailed to the company and they will send a quote of manufacturing and shipping costs and will send the boards.

Advanced circuits provides discounts and sponsorships for students seeking to use printed circuit boards for their projects. With $33 for a PCB with a 2 layer full specified, small quantity, special. This company offers a turn time of same day to 5 days and can print up to 10,000 pcbs per order. The material they used is FR-4 which is grade designation to glass-reinforced laminate sheets.

The plan for assembly is to order components for the schematics drawn, test and verify it is a working circuit on the breadboard. Then once its verified and working on the breadboard, there will be modifications towards the schematic before emailing the manufacturers to have the boards printed.

7.1.2. Final Decision

We chose the company 4PCB as our vendor of choice. 4PCB has great pricing for standard printed circuit boards at the affordable price of $33 per board. 4PCB also had the quickest processing time, which was necessary since many changes were made to the board. As a result, a second PCB design had to be ordered in the final weeks of the project implementation.

All assembly was carried out by team members. As a result, each component in the schematic had to fetch from certain libraries to ensure that the part was available for purchase so it could be properly soldered onto the board. All through hole components were chosen in the schematic to ease the process of assembly. Every component was ordered ahead of time from various vendors.

7.2. Schematics and Design

Section 7.2 contains schematics for the microcontrollers that will be used in this project. The program used to design these schematics is the Eagle PCB Design version 7.7.0. The Figure 7.1 contains all the components and chips needed to power one microcontroller. This circuit is straightforward and contains power regulation circuit so the MCU will not overload with header pins that will be used to connect the different types of components needed for the robot.

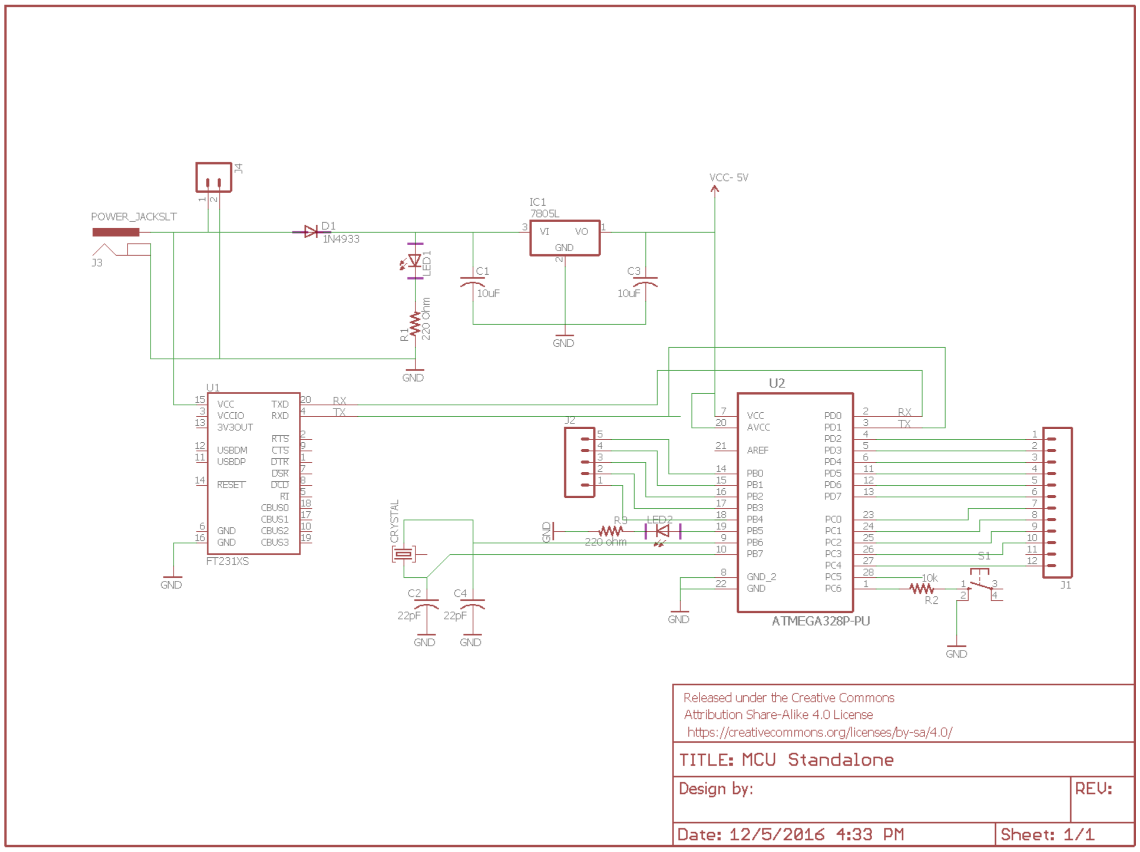


Figure 7.1: Microcontroller Schematic
(Team Designed)

Figure 7.2 is the proposed PCB layout design. All the headers will be lined up for easy access of jumper wires. Due to space constraint to the robot, this PCB design is intended to be no bigger than 68.6x53.44 mm.

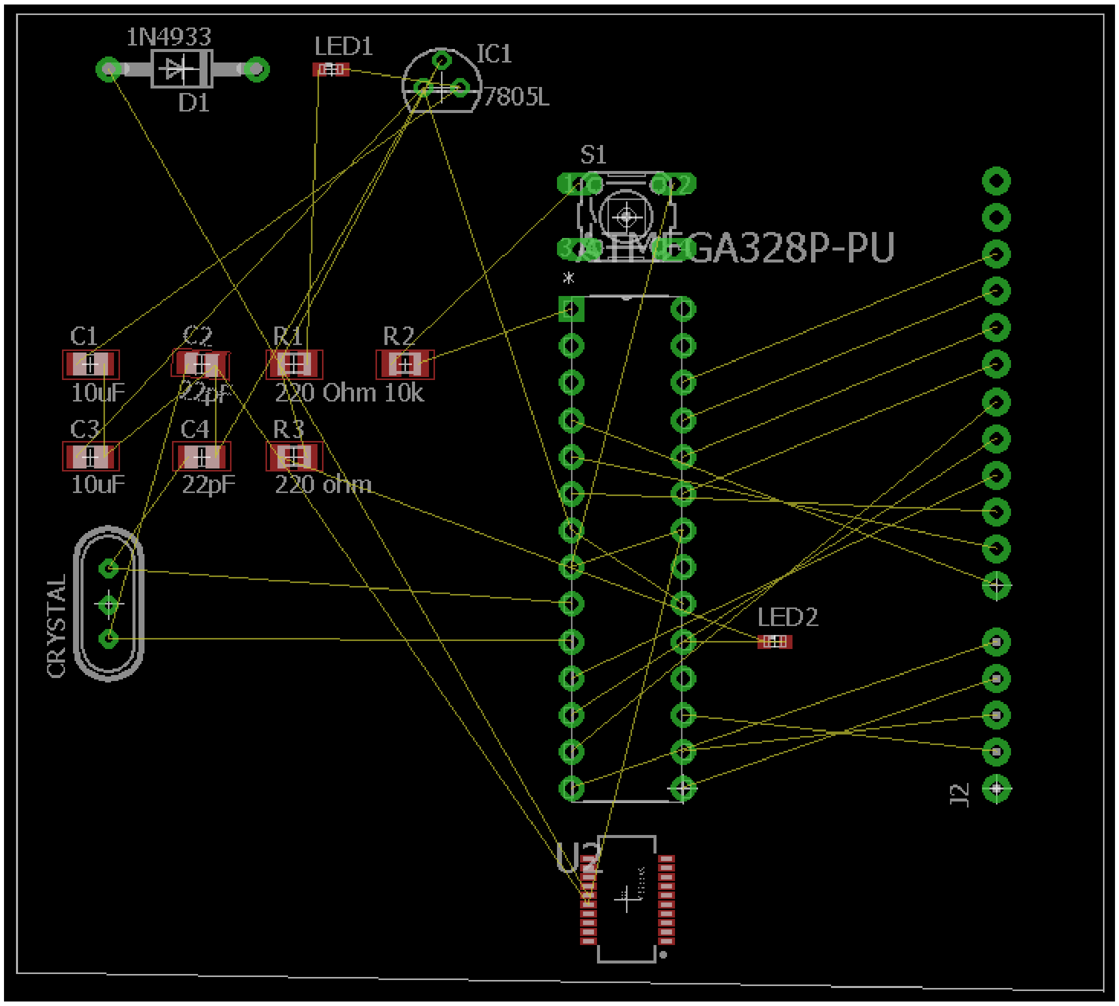


Figure 7.2: PCB Layout
(Team Designed)

This project will be required to use two MCU's therefore two PCBs will be constructed. In Figure 7.3, contains a full design of how the two MCUs will interact with the different components of the project. Figure 7.3 will not be implemented into a PCB, but rather will serve the purpose of a guideline when it comes to testing and integrating all of the components.

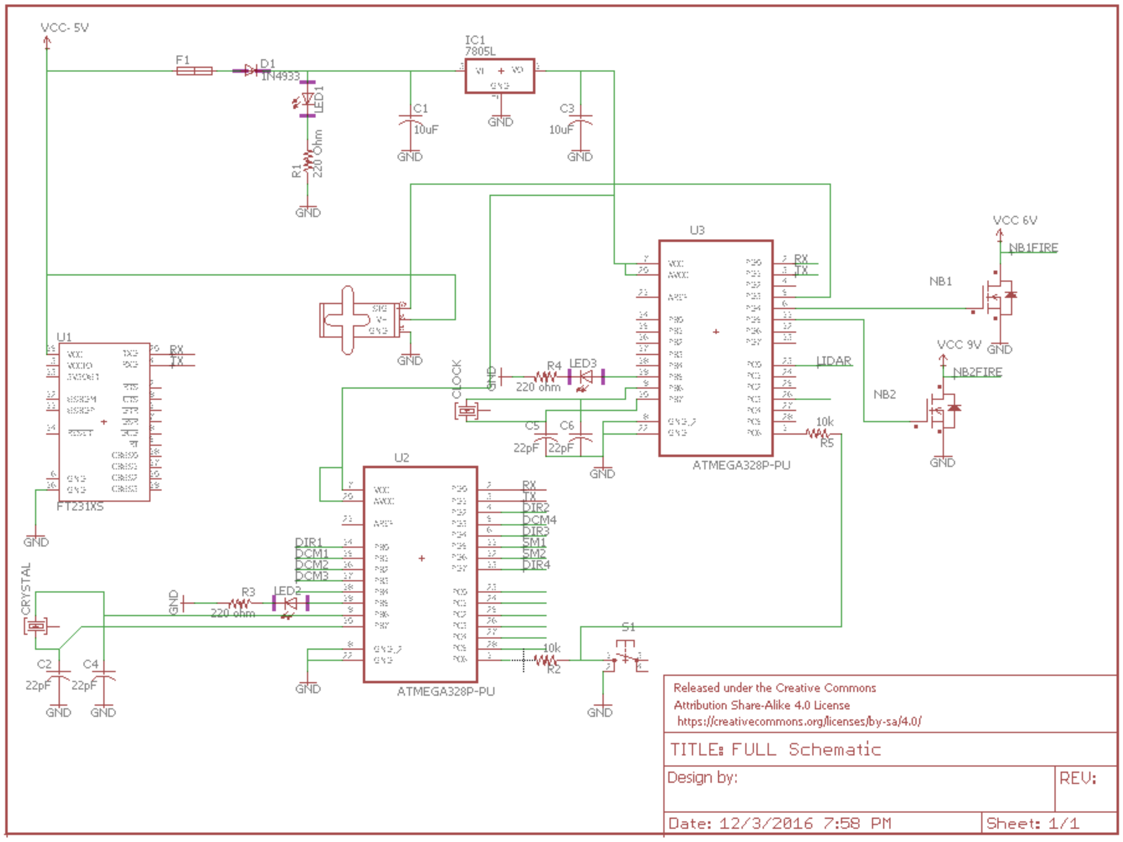


Figure 7.3: Ideal full schematic design of robot
(Team Designed)

As the project moved on, more features were needed to implement the final design. In the figure below contains the final schematic design for the PCB requirement. Initially, the idea of two MCUs were going to be used to drive 4 motors but has changed to only two motors allowing more pins to be free.

There were two PCB design changes. The first one was implemented to create space for stepper motor drivers and for initial NERF ball blasters. There were flaws in the amount of current allowed to each device therefore modifcations were to be made.

In the final PCB design, the following features were added to meet the demands of the robot. The final PCB design contained several power ports to power the NERF blasters and the Jetson TK1. Connections for the Bluetooth, encoders and LIDAR were also used in the PCB. Finally MOSFET switches were used to ensure that the NERF Blasters could be fired digitally.

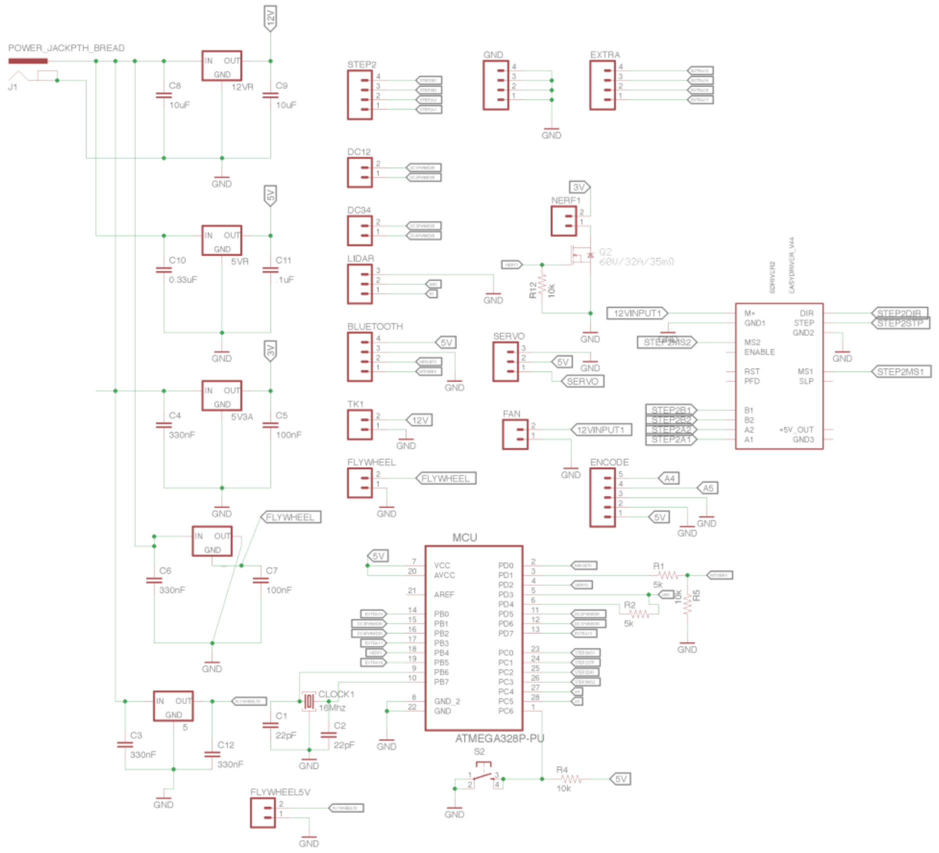


Figure 7.4 Full Schematic of final design

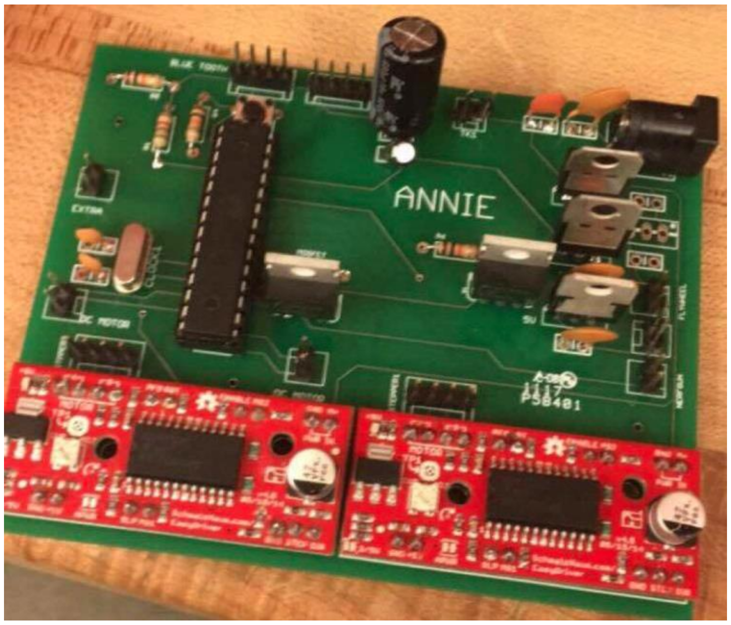


Figure 7.5 PCB assembly

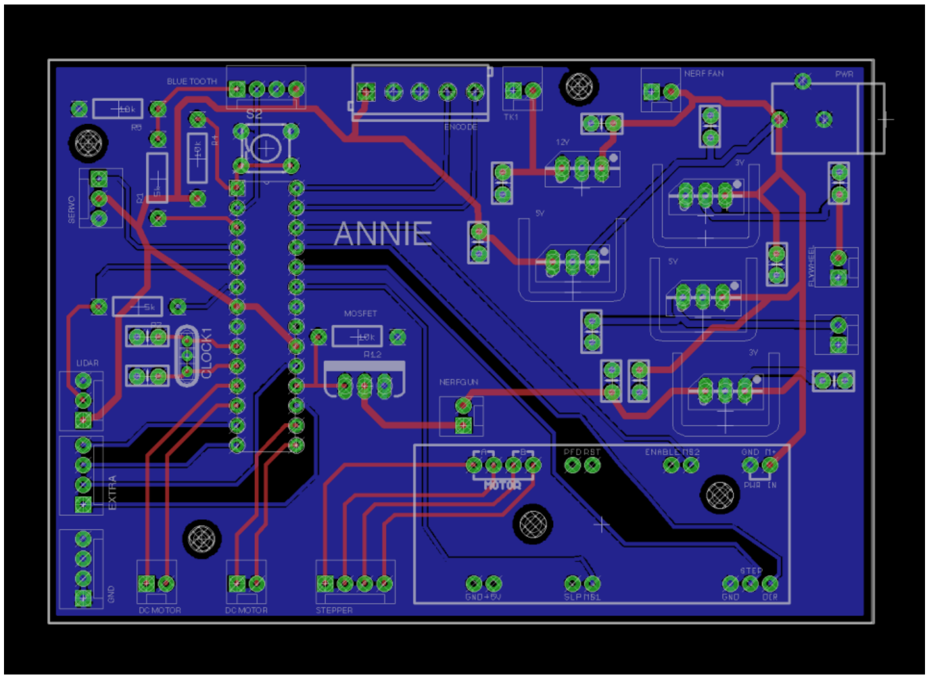


Figure 7.6 PCB Board layout

7.3. Coding Plan

In order to successfully implement the proposed system, a coding plan must be established in order to prioritize the work that must be accomplished. The following outline below highlights every task that must be accomplished as well as the order in which these tasks should be executed.

I. ATMega328p communication with Raspberry Pi:
   a. Establishing communication between the Raspberry Pi and the ATMega328p is essential for the entire implementation of the proposed system.
   b. Every component in the proposed system will either be connected to the ATMega328p or the Raspberry Pi.
   c. In order to transmit data wirelessly, the Raspberry Pi will be used; therefore, data from the ATMega328p will be transferred to the Pi so it can reach the laptop. (ex. Manual Navigation data)

II. Raspberry Pi receiving output from from the Logitech HD Pro Webcam
   a. This is essential to allow for the Computer Science team to start on their algorithm for autonomous detection.

III. ATMega328p receiving output from the Lidar Lite 3 Rangefinder
   a. Again, this is essential so that the Computer Science team is able to start with their contribution to the project

IV. Programming Stepper Motors to rotate Nerf-Blasters, Camera, and Rangefinder
   a. This is an important part of the overall development of the proposed system's design.

    b. This step is key to complete as soon as the connection has been established with the camera and rangefinder, since once an object has been detected, the system will aim and fire at that object.

V. Programming the Nerf-Blasters to fire with a button from the laptop
    a. This will immensely assist with the Computer Science team's plan to eventually autonomously aim and fire with the system.
    b. This feature is essential for further testing with the proposed system.

VI. Programming the DC Motors to rotate appropriately for manual navigation of the robot
    a. Manual navigation is a key feature that will be very important to the overall design of the proposed system.
    b. However, manual navigation will not be possible until we can obtain the necessary components from the Mechanical Engineering team.
    c. Although testing and preliminary models can be designed to practice establishing manual navigation, it is not as pressing as other features that must be implemented first.

8. Prototype Testing

This chapter covers all the prototype testing for both hardware and software designs that will be taken place with strategic procedures to ensure all components are connected properly and integrated correctly. Testing will be an essential part of the overall development life cycle of the system.

If either a software or a hardware design component does not pass a certain test, it will be tested again in order to figure out the possible problem. If a certain hardware component seems to be malfunctioning, a new component will be ordered in its place. If the component is not working due to faulty wiring, the team will have to redevelop their current design. If the component is not responding due to a software error, the team will have to rethink their current software design for a better solution.

8.1. Hardware Testing

This section focuses on hardware testing environments and planned procedures to ensure that each component is connected and working properly for the project. Hardware testing will be carried out accordingly once all parts arrive from their respective manufacturers. It is important that the hardware testing is carried out before software testing, since all software designs will be based on the assumption that the hardware is operating properly. Each section will contain a table labeling the type of testing with an objective and will then explain the practical procedure and the expected result from the testing.

8.1.1. Testing Environment

The competition is set to be indoors away from any outside elements, therefore all testing will be indoors in a controlled environment. The component and PCB testing will be conducted in labs with desktop computers. This will ensure that all necessary materials for PCB testing will be provided and easily accessible by team members. Once the robot is assembled, the field testing will be conducted in a field similar to the dimensions used in Figure 2.1.

The testing environment will be constructed duct-tape or rope to simulate the boundaries of the field. The testing environment will also have boxed to simulate obstacles that will be available during the competition. These obstacles must be present in the test environment, since the Computer Science team will need to avoid these obstacles within their algorithm. The testing environment will also have simulated targets, such as a box with a face on it for one of the stationary targets. The team will have a robot within the testing environment that will simulate an enemy for target practice as well. Someone will also be labeled as the medic to ensure that all methods of scoring are properly tested.

8.1.2. Camera

This section focuses on testing the camera through its hardware interactions to make sure that they are working seamlessly. The camera testing is an essential step towards a complete hardware design. Camera integration will allow us to successfully modify the system by being able to test whether the camera is properly connected. The camera will have to be connected to the Raspberry Pi in order to allow the Computer Science team to utilize their computer vision algorithms. Ultimately, the Raspberry Pi will be responsible for the image processing from data received by the camera. See Table 8.1 for details.

Test Objective	To test the camera module to turn on and show video stream of surroundings.
Equipment	1. Raspberry Pi 3 2. Logitech Webcam 3. Power supply 4. Desktop Computer
Preparation	1. Connect camera to Raspberry Pi. 2. Load program onto Raspberry Pi. 3. Open video viewer for camera.
Procedure	1. Check for all component connections. 2. Open video feed from computer
Expected Result	The video feed should show the camera's view.

Table 8.1: Camera Testing module

8.1.3. Motors

For testing the motors, it will be important that each motor is receiving power and running properly. DC motors will conduct a test through the designed PCB to make sure it is getting enough power. The stepper motors will be tested to ensure that they are rotating properly and connected securely to the PCB. Both the DC motors and the stepper motors will be connected to the power supply. It is essential that the DC motors tested thoroughly, since they will be in charge of manual navigation. However, it is also important that the stepper Motors are tested, since they will be in charge of the nerf-blaster's ability to pan and tilt.

Additionally, the connections between the DC motors and the DC motor controller, as well as the Motor Controller for the stepper and servo motors will need to be tested.

8.1.3.1. DC Motors

This is the testing module (seen in Table 8.2) for the DC motors intended to drive the robot around. This test will ensure that the DC motors are able to rotate as they are designed to do. DC motors will be the main source that will allow for programming of the wireless manual navigation.

Test Objective	To test whether the DC Motors are able to rotate properly once connected.
Equipment	1. 4 DC Motors 2. ATMega328p 3. Power Supply 4. 2 Dual Channel Motor driver 5. Desktop for programming
Preparation	1. Connect the two of the four DC motors to one motor driver. 2. Connect the driver to the pins of the ATMega328. 3. Repeat Steps 1 and 2 for the other two motors. 4. Connect the driver and PCB to the power supply. 5. Connect USB from computer program to the PCB. 6. Turn on the power supply.
Procedure	1. Load the test program to the microcontroller. 2. Execute the test program.
Expected Result	The testing program should allow the DC motors to rotate to the user's desired position.

Table 8.2: DC motor testing module

8.1.3.2. Stepper Motors

This section contains the testing module for the two stepper motors that will be utilized for aiming the Nerf-blasters. The module (Table 8.3) goes into detail of the objective, equipment needed how to prepare for the testing and the procedure. The final result is the expect result and if not performed correctly will go through trouble-shooting methods to resolve the project.

Test Objective	Ensure that the two Stepper motors are connected properly and can rotate from a code command.
Equipment	1. 2 Stepper Motors 2. 2 Stepper Motor Drivers 3. PCB design 4. Computer 5. Power supply
Preparation	1. Connect each stepper motor to respective motor driver. 2. Connect the stepper motor drivers to the power supply. 3. connect the motor driver to the PCB design. 4. Connect the PCB design to computer with program. 5. Turn on the power supply.
Procedure	1. Load the program onto the microcontroller. 2. Execute the program.
Expected Result	The program should command the stepper motors to turn a few phases either left or right. The stepper motors should be able to follow the program and end.

Table 8.3: Stepper motor testing module

8.1.4. Microcontrollers

This section contains a testing module (Table 8.4) for the microcontroller units that will be used to control the motors and the sensors for the project. The microcontroller, ATMega328p, will connect almost every component within the proposed system. Some of the component that will be connected to the ATMega328p include the power supply, the DC and stepper motors, the Raspberry Pi, the Lidar Lite 3 Rangefinder, and the nerf-blasters. To ensure that the system design is functioning properly, there will be multiple, repetitive tests on the microcontroller.

Test Objective	To ensure that the PCB designs are powered and each pin is functional.
Equipment	1. ATMega328p 2. Fuses 3. Diode 4. Capacitors 5. Resistors 6. LEDs 7. Clock Crystal 8. Push Button 9. 5V power supply
Preparation	1. Assemble the microcontrollers using all elements to complete the powering circuit. 2. Boot load the ATMega328 with program. 3. Connect power supply to the ATMega328p. 4. Turn on power supply.
Procedure	1. Run a program to enable a few pins at a time with an LED connected to each pin to be tested.
Expected Result	The LEDs should turn on for any pins enabled that would be used as an output.

Table 8.4: Microcontroller testing module

8.1.5. Nerf-Blaster Firing

This testing module (seen in Table 8.5) goes through the preparation and procedure in order to fire the Nerf-blasters that will be controlled digitally. The nerf-blaster will be the main component that allow us to score in the competition. Since we will be programming the guns to fire using a MOSFET switch circuit, multiple tests will need to be performed to get the guns to fire without pulling the trigger. This will enable the Computer Science team to use this feature during their programming of the autonomous system.

Test Objective	To ensure that the Nerf-blasters can be fired using a digitally controlled switch.
Equipment	1. 2 Nerf-blasters 2. ATMega328p 3. MOSFET switch circuit 4. Power Supply
Preparation	1. Set up MOSFET Switch circuit to the ATMega328p. 2. Connect the power supply to the two Nerf-blasters. 3. Connect the output of the MOSFET switch circuit to the Nerf-blasters. 4. Turn on the power supply. 5. Run program to enable the ATMega328p pins.
Procedure	1. Run the program that will allow voltage to turn on the MOSFET 2. Have program enable either Nerf-blaster 1 and 2 to fire
Expected Result	The Nerf-blaster should respond to the program execution to fire the darts and balls at a target.

Table 8.5: Nerf-blaster firing testing module

8.1.6. Rangefinder

This is the testing module (see Table 8.6) for the LIDAR range finder with a full list of what needs to be accomplished for the LIDAR to do its function. The LIDAR range finder will need to be tested in order to prevent the system from firing outside of the course range. This will also need to be tested in order to prevent the range finder from firing within its own zone. The distance value given by the range finder will be able to label an individual as a “friend” or an “enemy”. Therefore, all of the functionality of the range finder listed above will be tested in order to make sure none of these errors occur. To begin these tests, it is important to test if the rangefinder is connected properly to the microcontroller.

Test Objective	To test if the rangefinder is connected properly to the microcontrollers and can give a proper reading.
Equipment	1. LIDAR Lite Range finder 2. Microcontroller 3. Wires 4. Desktop Computer 5. Power supply 6. Measuring tape 7. Object
Preparation	1. Connect LIDAR to microcontroller 2. Connect power supply to microcontroller and LIDAR 3. Load program onto micro controller 4. Turn on power supply 5. Set and measure the distance of an object up to 40 ft.
Procedure	1. Turn on all power components 2. Load program onto microcontroller 3. Aim LIDAR at pre-measured object 4. Run the program
Expected Result	The LIDAR-Lite should turn on and record the distance of the object. If the object was 40ft away, the LIDAR should be able to detect and list the distance.

Table 8.6: LIDAR integration testing module

8.1.7. Power Distribution

This is the testing module (see Table 8.7) for power distribution across the entire robot testing on a singular battery. Power should be distributed correctly throughout all components that connected to the proposed system. Power distribution is very important to test. During the competition, we do not want a particular component draining the bulk of the power supply. This would indeed cause our robot to lose power during the competition. Therefore, it is imperative to continue testing power distribution throughout the course of this project.

Test Objective	To test all power distribution to all components using the battery so that every component is consuming the right amount of power.
Equipment	1. Raspberry Pi 2. Microcontrollers with voltage regulators 3. 2 Sensors 4. 4 DC motors 5. 2 Stepper Motors 6. 2 Nerf-blasters 7. Battery 8. Multi-meter
Preparation	1. Connect each component to the battery 2. Check all connections 3. Run through each component for power calculation 4. Fully charge battery
Procedure	1. Turn on battery power
Expected Result	All components should be powered through one battery and work up to 40 minutes.

Table 8.7: Power distribution testing module

8.2. Software Testing

This section highlights the necessary procedure in order to test the software developed for the proposed system. Each table clearly defines how the program will run, what hardware is required to execute the code, and the necessary steps required to allow for proper testing.

8.2.1. Environment

The environment in which all testing will be performed will consist of the following: This environment will be an indoor environment to simulate the same setting that will be present during the competition. This environment will also be an open space to prevent any outside factors from interfering with test results. The test environment will allow for multiple tests to be simulated so that every test can be executed properly. Finally, the test environment will have all outside materials necessary in order to complete each test.

8.2.2. Autonomous Detection

These tests (seen in Table 8.8 and 8.9) will highlight the Electrical team's plan to test the autonomous detection software that will be developed by the Computer Science team. This will help aid in the overall progression of the project.

Test Objective	To autonomously detect stationary and moving targets.
Equipment	1. Logitech HD Pro Webcam 2. Raspberry Pi 3. ATMega328p 4. Power Supply 5. Desktop for programming
Preparation	1. Connect the Logitech HD Pro Webcam to the Raspberry Pi 2. Connect the Raspberry Pi the ATMega328 3. Connect the power supply to the ATMega328 4. Connect USB from computer program to the Raspberry Pi 5. Turn on the power supply
Procedure	1. Load the program to the Raspberry Pi that will be used to interface with the camera to detect objects 2. Present an object that the robot has been trained to detect. 3. Execute the test program
Expected Result	As the program is executed, the detected objects should be highlighted through the video feed streaming from the Logitech HD Pro Webcam. The detected objects should be indicated in some way as a friend or an enemy.

Table 8.8: Autonomous Object Detection Test

Test Objective	To autonomously detect human faces.
Equipment	1. Logitech HD Pro Webcam 2. Raspberry Pi 3. ATMega328p 4. Power Supply 5. Desktop for programming
Preparation	1. Connect the Logitech HD Pro Webcam to the Raspberry Pi 2. Connect the Raspberry Pi the ATMega328 3. Connect the power supply to the ATMega328 4. Connect USB from computer program to the Raspberry Pi 5. Turn on the power supply
Procedure	1. Load the program to the Raspberry Pi that will be used to interface with the camera to detect human faces 2. Present human face to camera 3. Execute the test program
Expected Result	As the program is executed, the detected objects should be highlighted through the video feed streaming from the Logitech HD Pro Webcam. The detected objects should be indicated in some way as a friend or an enemy.

Table 8.9: Autonomous Facial Detection Test

8.2.3. Camera Sensor

This test will be done to see if information can be gathered from the camera. Details can be seen in Table 8.10 below.

Test Objective	To receive and store input from the Logitech HD Pro webcam
Equipment	1. Logitech HD Pro webcam 2. Raspberry Pi
Preparation	1. Connect the Logitech HD Pro Webcam to the Raspberry Pi 2. Turn on Raspberry Pi
Procedure	1. Open a terminal 2. Execute "fswebcam image.jpg" command to take an image and save a jpeg
Expected Result	A photo will be taken and stored on file.

Table 8.10: Camera Output Storage Test

8.2.4. Rangefinder Sensor

This section illustrates the software development test plan that will be executed in order to check that information is being received from the Lidar Lite 3 Rangefinder accordingly (see Table 8.11).

Test Objective	To pull input from the LIDAR-Lite 3 Rangefinder into the software algorithm.
Equipment	1. ATMega328p connected to PCB 2. Power Supply 3. Desktop 4. Lidar Lite 3 Rangefinder
Preparation	1. Connect the LIDAR-Lite 3 Rangefinder to the PCB. 2. Connect the PCB to the Power Supply. 3. Connect USB from the desktop to the PCB. 4. Turn on the power supply.
Procedure	1. Open a terminal. 2. Execute command that will receive information from the LIDAR-Lite 3 rangefinder. 3. Store this data into a usable variable. 4. Read from variable.
Expected Result	As the command is executed, the rangefinder shall give information regarding the distance of a given object. This data should be stored into an easy to use format for use in algorithms developed by the Computer Science team.

Table 8.11: Rangefinder Sensor Test

8.2.5. Manual Navigation

This test (see Table 8.12) highlights the steps required to test whether the programming done for the manual navigation has been implemented correctly. If there are any faults in this testing, a troubleshooting procedure will be taken to ensure all components are connected.

Test Objective	All four motors are connected and can implement a tank drive.
Equipment	1. 4 DC Motors 2. ATMega328p 3. Power Supply 4. 2 Dual Channel Motor driver 5. Desktop for programming
Preparation	1. Connect the two of the four DC motors to one motor driver 2. Connect the driver to the pins of the ATMega328 3. Repeat Steps 1 and 2 for other two motors 4. Connect the driver and PCB to the power supply 5. Connect USB from computer program to the PCB 6. Turn on the power supply
Procedure	1. Using the WASD keyboard configuration from computer: a. Press the W key for forward rotation b. Press the S key for reverse rotation c. Press the A key for left rotation d. Press the D key for right rotation
Expected Result	As the W key is pressed, all the motors should be rotating in the forward direction. As the S key is pressed, the DC motors should turn in reverse. As the A key is pressed, the right pair of DC motors should rotate slower than the left pair of DC motors and vice versa for when the D key is pressed.

Table 8.12: Manual Navigation Testing Module

8.2.6. Nerf-Blaster Pan/Tilt Position

This test illustrates what is required to test the user's ability to control the Nerf-blaster's position. This test will be crucial in allowing the Computer Science team to develop their algorithm. See Table 8.13 for more details.

Test Objective	To be able to move the Nerf-blaster to the user’s desired position by programming the stepper motors to pan and tilt the turn axis.
Equipment	1. 2 Stepper Motors 2. 2 Stepper Motor Drivers 3. ATMega328p connected to PCB 4. Power Supply 5. 2 Turn axis 6. 2 Nerf-blasters 7. Desktop for programming
Preparation	1. Connect each stepper motor to a stepper motor driver 2. Connect the driver to the pins of the ATMega328 3. Connect the driver and PCB to the power supply 4. Connect USB from computer program to the PCB 5. Turn on the power supply 6. Setup connection between each stepper motor and each turn axis 7. Place Nerf-blasters inside each turn axis
Procedure	1. Using the arrow keys on the user’s laptop, push the up arrow key to test if the guns will tilt in an upward position 2. Push the down arrow key to tilt the guns in a downward position 3. Push the right arrow key to pan guns to the right 4. Push the left arrow key to pan guns to the left
Expected Result	As the up arrow key is pressed, the guns should tilt upward. As the down arrow key is pressed, the guns should tilt downward. As the left arrow key is pressed, the guns should pan to the left. As the right arrow key is pressed, the guns should pan to the right.

Table 8.13: Nerf-blaster Pan and Tilt test

8.2.7. ATmega328p Configuration

The test case purposes for the ATmega328p are to ensure that there is direct communication with the Raspberry Pi 3 Model B and the Encoders that are attached to the Gear Motors. These test cases can be seen in detail in Table 8.14 and in Table 8.15

below. While there are several other components connected to the ATmega328p, test cases for those components and their subsystems have been or will be discussed in other sections of the testing chapter.

Test Objective	To check if the ATmega328p chip is receiving navigation input from the Raspberry Pi 3 Model B.
Equipment	1. ATMega328p connected to PCB 2. Raspberry Pi 3 Model B 3. Power Supply 4. Desktop for programming
Preparation	1. Make sure ATMega328p is connected to Raspberry Pi 3 Model B and that both devices are correctly connected to the power supply 2. Turn on the power supply 3. Establish a computer network connection to the Raspberry Pi either by LAN or Wi-Fi
Procedure	1. Send mock navigational control signals to Raspberry Pi using the arrow keys on the computer 2. Read data transmission logs from ATmega328p
Expected Result	The data logs from the ATmega328p should read that there were navigational signals (left, right, up, and down) transmitted to it.

Table 8.14: Input from Raspberry Pi 3 Model B Test

Test Objective	To check if the ATmega328p chip is receiving displacement input from the Encoders attached to the Gear Motors.
Equipment	1. ATMega328p connected to PCB 2. DC Motor Controllers 3. Gear Motors with attached Encoders 4. Raspberry Pi 3 Model B 5. Power Supply 6. Desktop for programming
Preparation	1. Make sure ATMega328p is connected to Raspberry Pi 3 Model B 2. Connect Motor Controllers to ATMega328p 3. Connect Gear Motors to Motor Controllers 4. Connect Encoders to ATMega328p 5. Establish that all components are connected to a power 6. Turn on the power supply 7. Establish a computer network connection to the Raspberry Pi either by LAN or Wi-Fi
Procedure	1. Send navigational control signals to Raspberry Pi using the arrow keys on the computer 2. Check to see that the wheels are turning 3. If the wheels are turning, read data transmission logs from ATmega328p to see if displacement data is coming in 4. If the wheels are not turning, recheck the component connections and repeat Step 3
Expected Result	The data logs from the ATmega328p should read that there were displacement values from the Encoders being sent to it.

Table 8.15: Input from Encoders Test

8.2.8. Communication

This test (see Table 8.16) is about making sure the Jetson TK1 can read a message sent by the ATMega328p. Setting up communication between the ATMega328p and the Jetson TK1 will ensure that all necessary data can be transferred to the Pi in order to wirelessly transfer the data to the laptop.

Test Objective	To check if the ATMega328p is able to transmit data to the Jetson TK1 3 Model B
Equipment	ATMega328p connected to PCB Jetson TK1 3 Model B Power Supply
Preparation	1. Connect the Jetson TK1 to the ATMega328p on the PCB via USB 2. Make sure the Jetson TK1 and PCB are powered and on. 3. Install Arduino IDE on the Jetson TK1 4. Install serialPy package on the Jetson TK1.
Procedure	1. Run Arduino IDE on the Jetson TK1 2. Use a test Processing language script with functions from the Arduino Serial Library such as “Serial, begin, println” begin serial transmission for the ATMega328p 3. Use a test script Python script with functions imported from serialPy such as “readLine” to receive data from ATMega328p.
Expected Results	The string sent by the ATMega should display in the Python program running.

Table 8.16: Jetson TK1 Receive Input from Microcontroller

9. Administrative

Project management, in general, will come down to time management and budgeting. The following sections provides details on milestones accomplished and waiting to be accomplished throughout the project, from concept to completion. This section also contains details on how the battlebot stayed within the bounds of the project budget.

9.1. Project Milestones

A summary of milestones completed in Fall of 2016 is listed below in Table 9.1.

Fall 2016 - Senior Design I		
Milestone	**Start**	**End**
Researched Project Ideas	8/22/16	9/9/16
Initial Project Idea Documentation	9/1/16	9/9/16
First Half Hour Meeting	9/19/16	9/19/16
First Sponsor Meeting	9/21/16	9/21/16
Met Mechanical and Computer Science Teams	9/21/16	9/21/16
First Sponsor Conference Call	9/30/16	9/30/16
First All Teams Meeting	9/30/16	9/30/16
Table of Contents	9/9/16	11/4/16
Draft Document	9/9/16	11/11/16
Last Half Hour Meeting	11/14/16	11/16/16
Last Sponsor Conference Call	11/21/16	11/21/16
Finalized Components List For Ordering	9/30/16	11/28/16
Ordered Components	10/28/16	11/28/16
Last All Teams Meeting	12/2/16	12/2/16
Final Document	9/9/16	12/6/16
Lockheed Martin PDR Presentation	11/2/16	12/13/16

Table 9.1: Fall 2016 milestones

A summary of milestones completed in Spring of 2017 is listed below in Table 9.2.

Spring 2017 - Senior Design II		
Milestone	**Start**	**End**
Test Components	12/15/16	1/15/17
Build Prototype	1/16/17	2/26/17
Test Prototype	2/26/17	3/02/17
Finalize Project	-	415/17
Final Presentation	4/20/17	4/20/17
Final Report	-	4/27/17
Battlebots Competition	4/14/17	4/14/17

Table 9.2: Spring 2017 milestones

9.2. Project Budget

This project was sponsored by the company Lockheed Martin. They allowed a maximum budget of $2,000, with the limitation that the final product be at a maximum as-demonstrated cost of $1,000. The budgeting for this project can be seen in two separate tables. Table 9.3 consists of primary main components of the system and Table 9.4 contains items that are additions to the main components (e.g. a microSD card for the Jetson TK1 to run its operating system and respective processes on).

Component	**Name**	**Quantity**	**Cost Per Unit**	**Total**
Image Sensor	Logitech HD Pro Webcam C920	1	$52.49	$52.49
Rangefinder	LIDAR-Lite 3 Laser Range Finder	1	$112.49	$112.49
Processing Unit	Raspberry Pi 3 Model B	1	$35.99	$35.99
Microcontroller	ATmega328p	2	$13.48	$26.96
Motorshield	10A Dual Channel Bi-directional DC Motor Driver	2	$23.49	$46.98
Encoder	*Attached to NeveRest 40 Gearmotor	4	$28.00	$112.00
Servo Motor	Futaba S3004 Standard Servo Motor	2	$12.49	$24.98
Stepper Motor	3V 1.7A 68oz-in Stepper Motor	1	$16.95	$16.95
Voltage Regulator	5V 1.5A Switching Voltage Regulator	1	$5.45	$5.45
Nerf-blaster (Dart)	CS-18 N-Strike Rapidstrike	1	$39.99	$39.99
Nerf-blaster (Ball)	Nerf Rival Zeus MXV-1200	1	$40.00	$40.00
			Total	$514.28

Table 9.3: Project budget for main components

Component	Name	Quantity	Cost Per Unit	Total
Miscellaneous	16 MHz Crystal	3	$0.86	$2.58
Miscellaneous	EasyDriver - Stepper Motor Driver	2	$13.46	$26.92
Miscellaneous	SparkFun USB to Serial Breakout - FT232RL	1	$13.46	$13.46
Miscellaneous	USB 2.0 Cable A-Male to B-Male	1	$4.99	$4.99
Miscellaneous	9V 1A Power Adapter for Arduino	1	$5.59	$5.59
Miscellaneous	32GB microSDHC USH-I card	1	$9.95	$9.95
Miscellaneous	Elegoo Upgraded Electronics Kit	1	$16.86	$16.86
			Total	$80.35

Table 9.3: Project budget for complements of main components

10. Conclusion

The design of the proposed system will highlight and demonstrate the technical skills that each team has acquired throughout the course of the engineering program. Every aspect of the current system’s design has been analyzed thoroughly with the hopes of developing a successful project. With thorough planning regarding research, hardware design, software design, and testing, the team has ultimately prepared for the build phase of the project’s implementation.

As the semester has progressed, the team has developed a detailed list of components that will be used in this senior design project build. The team has also created a timeline of when tasks must be completed to keep all team members on schedule. The team has kept up with finances, constantly updating spreadsheets with new values of parts that will be ordered. Although all of this tedious planning has been executed to the utmost degree, the team must expect changes and must be willing to adapt as the project moves into the development phase.

Working cross-discipline with other majors has allowed the Electrical and Computer Engineering team to gain experience that is closer to the real world. Dealing with various team members from different disciplines can be fairly common in the engineering work environment. This opportunity of learning and engaging with other individuals has been truly enlightening for the Electrical and Computer Engineering team, as many new ideas have continued to develop throughout the project’s planning phase.

The key purpose of the Nerf-Enabled Battlebot is to be able to autonomously detect, aim, and fire at an enemy target. This opportunity to build the proposed system has allowed the Red team to be exposed to new concepts regarding computer vision and autonomous robot systems. Given the opportunity to learn and develop regarding electrical components and how they affect the autonomy of the system, the Computer and Electrical team have truly grown, and will continue to grow in the following semester.

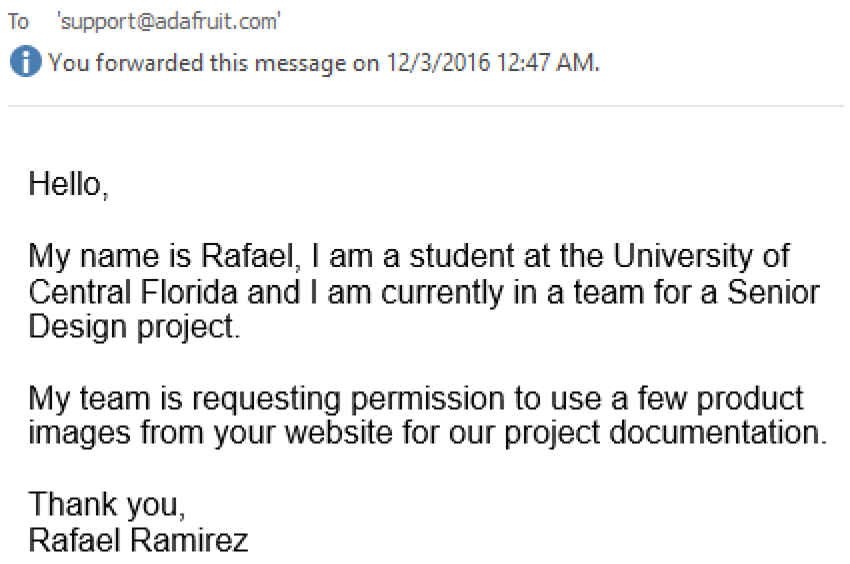
The Electrical and Computer Science team has set certain goals for the next semester. As a whole with the other majors, the team hopes to win the battlebot competition against the other teams that will be competing. The team also hopes to have a fair and friendly competition with competitors. The team will be trying to get the implementation of the proposed system completed as early as possible in order to allow for multiple changes. The final goal of this team is to work together and pull through the final course of the engineering degree.

Appendices

Appendix A - Copyright Permissions

Adafruit
support@adafruit.com

Figure(s): 3.20
Status: Requested



To 'support@adafruit.com'

You forwarded this message on 12/3/2016 12:47 AM.

Hello,

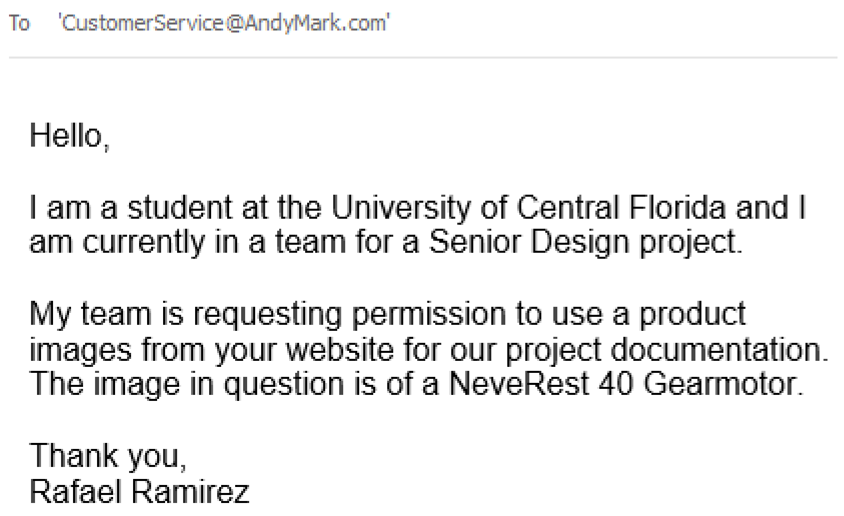
My name is Rafael, I am a student at the University of Central Florida and I am currently in a team for a Senior Design project.

My team is requesting permission to use a few product images from your website for our project documentation.

Thank you,
Rafael Ramirez

AndyMark
customerservice@andymark.com

Figure(s): 3.29
Status: Requested


To 'CustomerService@AndyMark.com'

Hello,

I am a student at the University of Central Florida and I am currently in a team for a Senior Design project.

My team is requesting permission to use a product images from your website for our project documentation. The image in question is of a NeveRest 40 Gearmotor.

Thank you,
Rafael Ramirez

Arduino
Editorial contents of the arduino.cc website, such as texts and photos, are released as Creative Commons Attribution ShareAlike 3.0. This means you can use them on your own derived works, in part or completely, as long as you also adopt the same license.

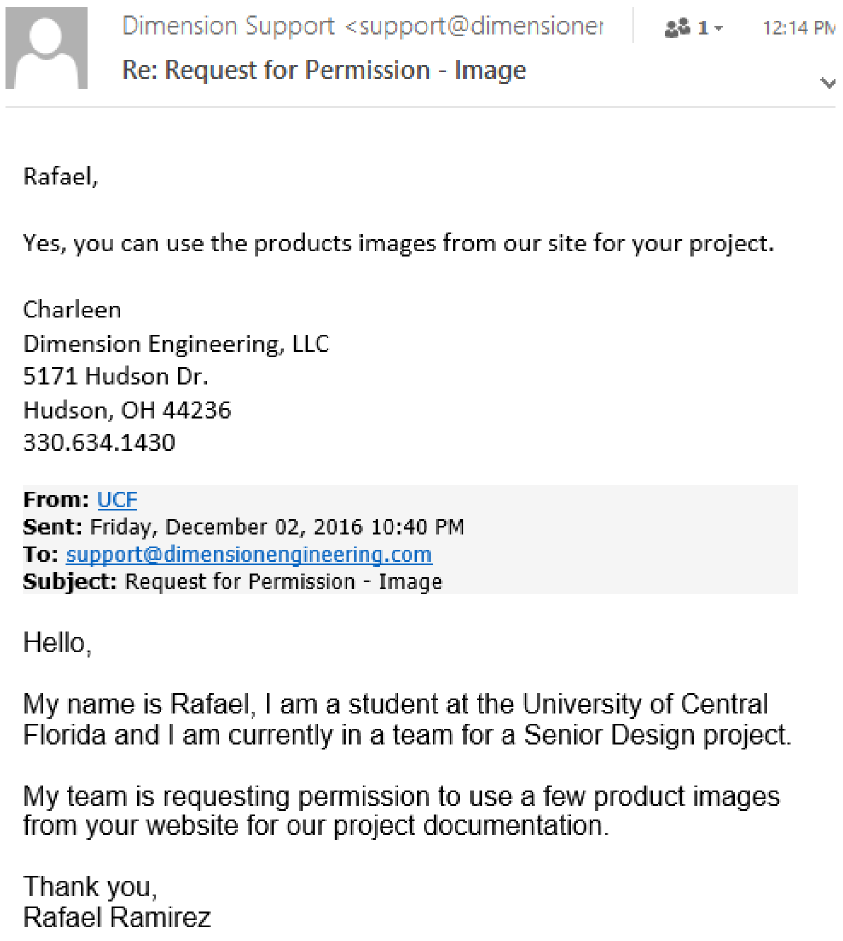
Figure(s): 5.2
Status: Granted

Chris Clark, instructor at Princeton
cmclark@cs.princeton.edu

Figure(s): 3.22 and 3.23
Status: Requested

Dimension Engineering
support@dimensionengineering.com

Figure(s): 3.11 and 3.32
Status: Granted


Dimension Support <support@dimensioner

1 12:14 PM

Re: Request for Permission - Image

Rafael,

Yes, you can use the products images from our site for your project.

Charleen
Dimension Engineering, LLC
5171 Hudson Dr.
Hudson, OH 44236
330.634.1430

From: UCF
Sent: Friday, December 02, 2016 10:40 PM
To: support@dimensionengineering.com
Subject: Request for Permission - Image

Hello,

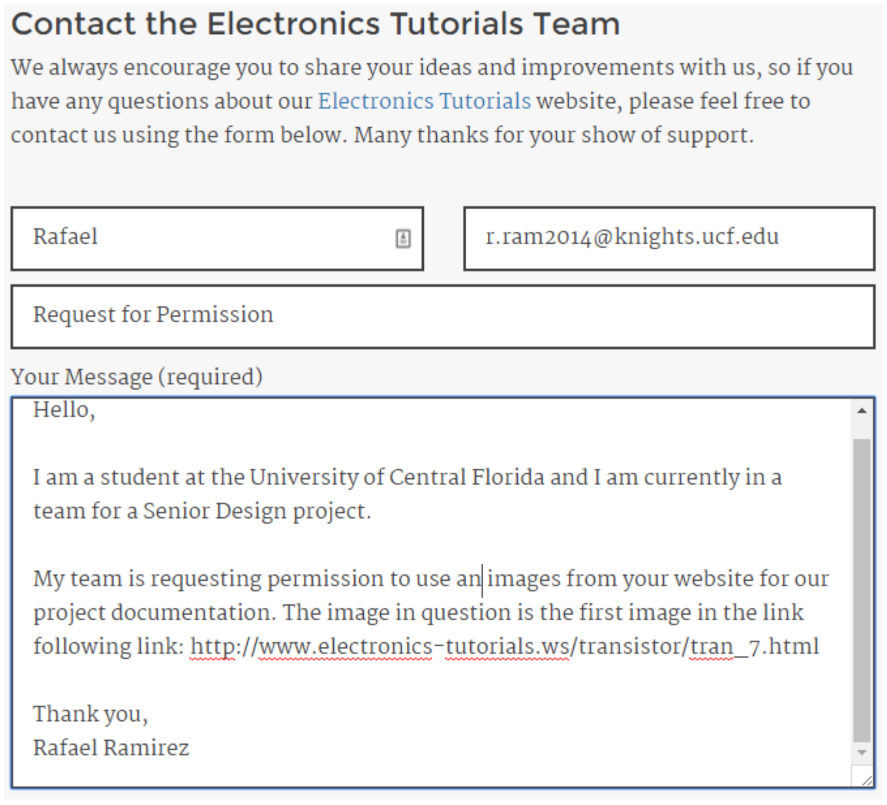
My name is Rafael, I am a student at the University of Central Florida and I am currently in a team for a Senior Design project.

My team is requesting permission to use a few product images from your website for our project documentation.

Thank you,
Rafael Ramirez

Electronics Tutorials
http://www.electronics-tutorials.ws/contact

Figure(s): 5.8
Status: Requested



Contact the Electronics Tutorials Team

We always encourage you to share your ideas and improvements with us, so if you have any questions about our Electronics Tutorials website, please feel free to contact us using the form below. Many thanks for your show of support.

Rafael	r.ram2014@knights.ucf.edu
Request for Permission	

Your Message (required)

Hello,

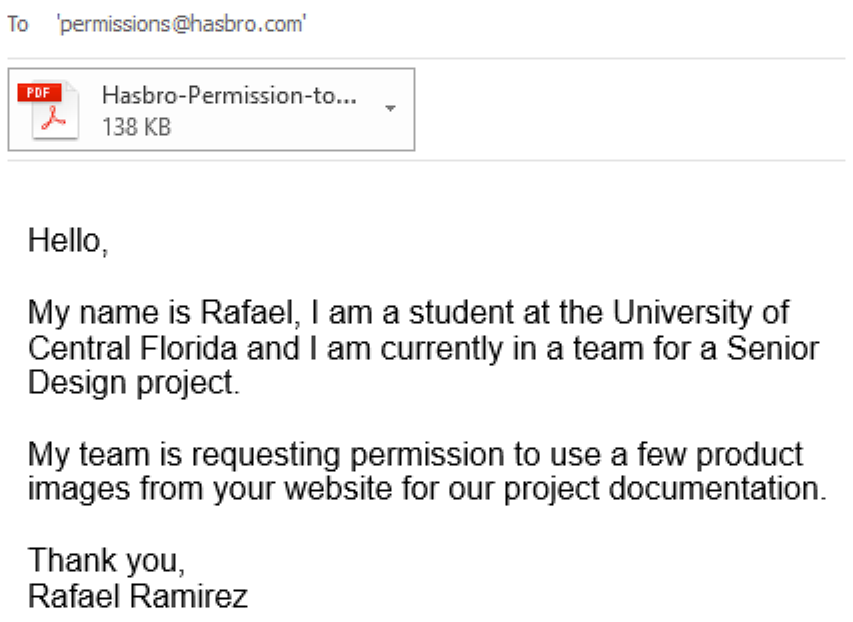
I am a student at the University of Central Florida and I am currently in a team for a Senior Design project.

My team is requesting permission to use an images from your website for our project documentation. The image in question is the first image in the link following link: http://www.electronics-tutorials.ws/transistor/tran_7.html

Thank you,
Rafael Ramirez

Hasbro
permissions@hasbro.com

Figure(s): 3.18 and 3.19
Status: Requested


To 'permissions@hasbro.com'

Hasbro-Permission-to...
138 KB

Hello,

My name is Rafael, I am a student at the University of Central Florida and I am currently in a team for a Senior Design project.

My team is requesting permission to use a few product images from your website for our project documentation.

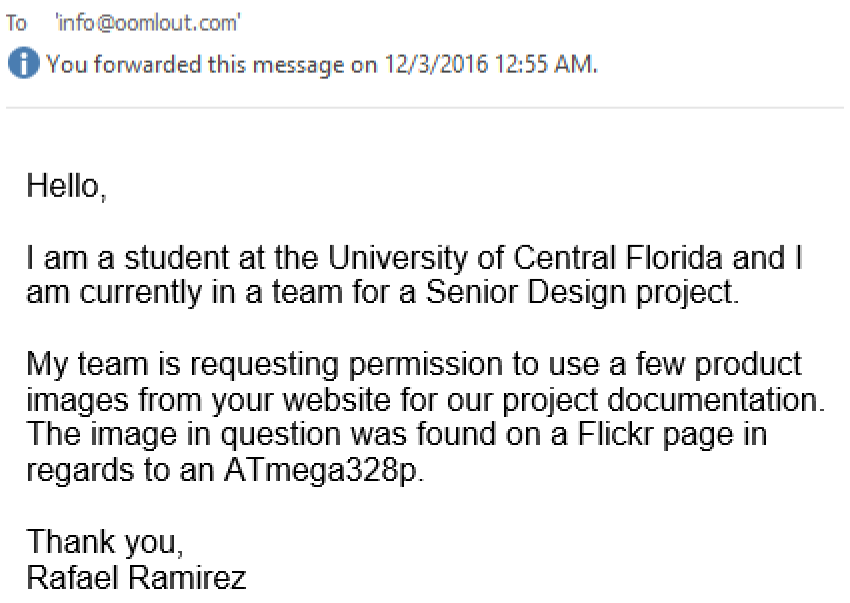
Thank you,
Rafael Ramirez

Logitech
All text, graphics, user or visual interfaces, trademarks, logos, music, sounds, artwork, photographs, and computer code ("Content"), including but not limited to the design, structure, selection, coordination, expression, "look and feel," and the arrangement of such Content, contained on this Website is owned, controlled, or licensed by Logitech. All such Content is protected by trade dress, copyright, patent and trademark laws, and various other intellectual property rights and unfair competition laws. Unless another agreement applies to particular Content (e.g., Software License Agreement, Terms of Service, etc.), Logitech hereby grants you limited permission to use the Content subject to these Terms, as long as the use of such Content is solely for your personal, non-commercial informational use.

Figure(s): 3.24
Status: Granted

oomlout
info@oomlout.com
These are the product photos we use on our web store www.oomlout.co.uk , please feel free to use them for whatever purpose you see fit, but please send us a message.

Figure(s): 3.27
Status: Requested


To 'info@oomlout.com'

You forwarded this message on 12/3/2016 12:55 AM.

Hello,

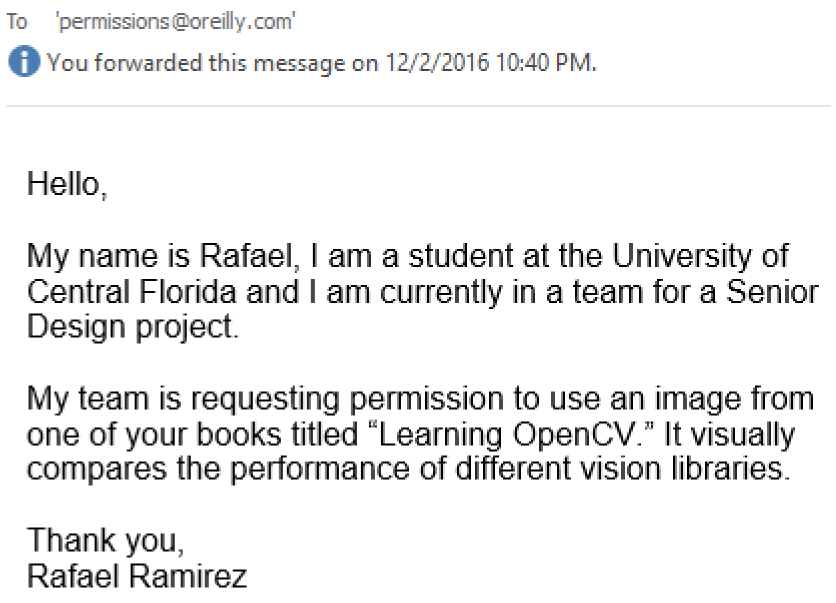
I am a student at the University of Central Florida and I am currently in a team for a Senior Design project.

My team is requesting permission to use a few product images from your website for our project documentation. The image in question was found on a Flickr page in regards to an ATmega328p.

Thank you,
Rafael Ramirez

O'Reily
permissions@oreilly.com
For permission to use text, code, or images from our books, please send us a brief email describing the specific information you'd like to reproduce and the final destination for or purpose of the reproduction.

Figure(s): 3.8
Status: Requested


To 'permissions@oreilly.com'

You forwarded this message on 12/2/2016 10:40 PM.

Hello,

My name is Rafael, I am a student at the University of Central Florida and I am currently in a team for a Senior Design project.

My team is requesting permission to use an image from one of your books titled "Learning OpenCV." It visually compares the performance of different vision libraries.

Thank you,
Rafael Ramirez

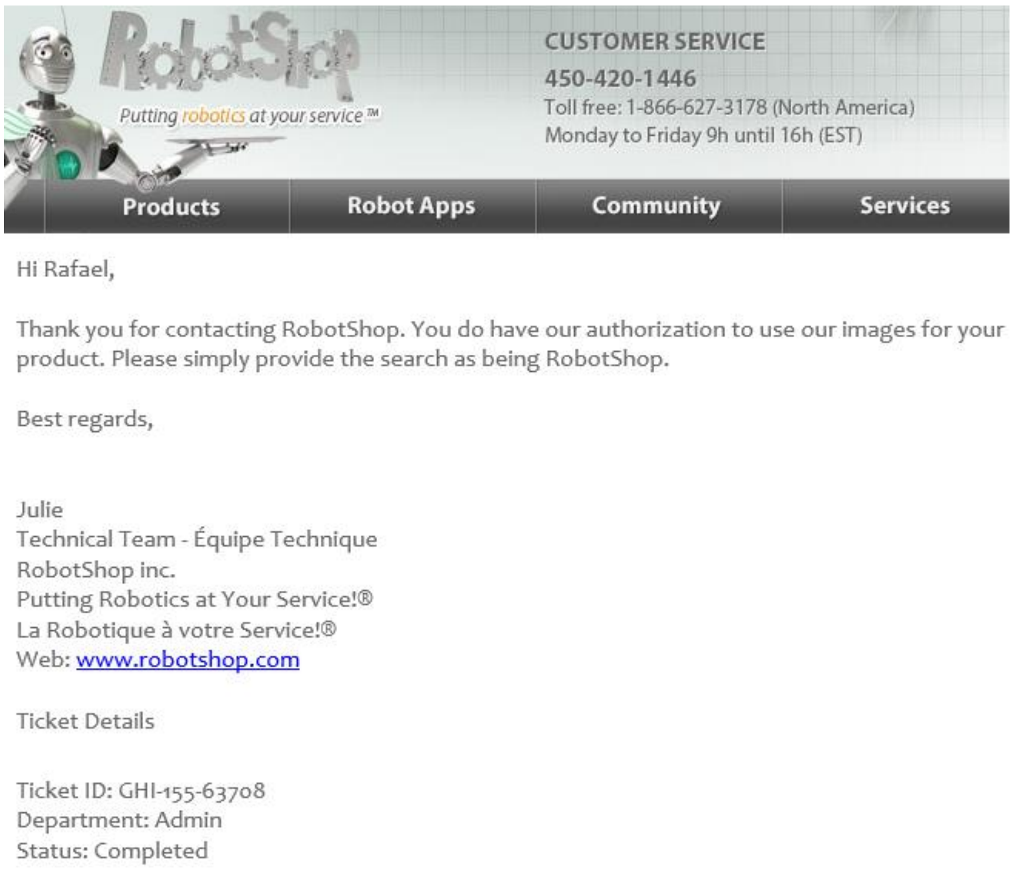
Photonix
Contact Michael @ michaellarabel.com

All information provided by Phoronix Media on any of its web properties is provided for use "as is" and is without warranty of any kind. In no event shall Phoronix Media be liable to any party for any direct or indirect damages for any use of their web-sites -- including, without limitation, any lost profits, business interruption, loss of programs, loss of programmed data, or otherwise. All information displayed on Phoronix Media web-sites are the property of Phoronix Media and is protected by copyright. Articles may not be copied or distributed without the prior written consent of Phoronix Media. Some web properties offer areas that contain readers' comments and opinions, which Phoronix does not have any responsibility or any liability over any readers' comments and opinions. By accessing any web property owned by Phoronix Media, you agree to these terms.

Figure(s): 3.3, 3.4, 3.5, and 3.6
Status: Granted

RobotShop
supportcenter@robotshop.com

Figure(s): 3.25, 3.26, 3.28, 3.30, and 3.31
Status: Granted


Putting robotics at your service™
CUSTOMER SERVICE
450-420-1446
Toll free: 1-866-627-3178 (North America)
Monday to Friday 9h until 16h (EST)
Products
Robot Apps
Community
Services

Hi Rafael,

Thank you for contacting RobotShop. You do have our authorization to use our images for your product. Please simply provide the search as being RobotShop.

Best regards,

Julie
Technical Team - Équipe Technique
RobotShop inc.
Putting Robotics at Your Service!®
La Robotique à votre Service!®
Web: www.robotshop.com

Ticket Details

Ticket ID: GHI-155-63708
Department: Admin
Status: Completed

Appendix B - Datasheets

http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_datasheet.pdf

https://www.element14.com/community/servlet/JiveServlet/previewBody/65470-102-1-287848/Raspberry-Pi%20Technical%20Data%20Sheet.pdf

Appendix C - Works Cited

[1] Unknown, (2016), *Brush DC Motor Guide,* retrieved from http://www.anaheimautomation.com/manuals/forms/brush-dc-motor-guide.php

[2] Unknown, (2015, April 15), *How does the Lead Acid Batteries Work?*, retrieved from http://batteryuniversity.com/learn/article/lead_based_batteries

[3] Woodbank Communications Ltd, (2005), *Lead Acid Batteries*, retrieved from http://www.mpoweruk.com/leadacid.htm

[4] Unknown, (2013, February 16), *How to Pick The Right Battery For Your Project*, retrieved from https://learn.adafruit.com/all-about-batteries/how-to-pick-the-right-battery-for-your-project

[5] PowerTech Systems, (2015), *Lithium-Ion vs Lead-Acid*, retrieved from http://www.powertechsystems.eu/home/tech-corner/lithium-ion-vs-lead-acid-battery/

[6] Bonheur, Kristoffer, (2016, March 13), *Lithium ion battery: Advantages and disadvantages*, retrieved from http://www.versiondaily.com/lithium-ion-battery-advantages-disadvantages/

[7] Unknown, (2016, May 30), *Nickel-Based Batteries*, retrieved from http://batteryuniversity.com/learn/article/nickel_based_batteries

[8] Texas Instruments, (2011), *Linear and Switching Voltage Regulator Fundamentals*, retrieved from http://www.ti.com/lit/an/snva558/snva558.pdf

[9] Holloway, James, (2016), *How Does a NERF Gun Work?*, retrieved from http://www.ehow.com/how-does_4699643_nerf-gun-work.html

[10] Roger's Hobby Center, (2016), *A Guide to Understanding LiPO Batteries*, retrieved from
http://rogershobbycenter.com/lipoguide/

[11] Future Electronics, (2016), *What Is a Switching Regulator,* retrieved from
http://www.futureelectronics.com/en/regulators-references/switching-regulators.aspx

[12] Morani, Giovanna. "Understanding Resolution In Optical And Magnetic Encoders." *Components Content from Electronic Design*., 23 June 2013. Web. 2016.
http://electronicdesign.com/components/understanding-resolution-optical-and-magnetic-encoders

[13] Unknown, "Accuracy and Ultrasonic Distance Meters." *Laser Distance Measurer*., 29 June 2015. Web. 2016. retrieved from
http://www.laser-distance-measurer.com/accuracy-and-ultrasonic-distance-meter/

[14] Krig, Scott. "Computer Vision Metrics." *Google Books*. Web. 2016. retrieved from
https://goo.gl/100P9C

[15] Unknown, *Lidar-uk.com*. Bluesky International Limited, Web. 2016. retrieved from
http://www.lidar-uk.com/index.php

[16] Calponi, Elso "Self-Targeting Autonomous Turret System" *University of Central Florida*, Web. 2016. retrieved from
http://www.eecs.ucf.edu/seniordesign/sp2014su2014/g08/documents/SD2%20Document%20Group%208.pdf

[17] Dodge, Brian "The Autonomous Sentry Robot" *University of Central Florida*, Web. 2016. retrieved from
http://www.eecs.ucf.edu/seniordesign/sp2015su2015/g09/Archive/ASRSD1FinalReport.pdf

[18] Diaz, Bryan "Autonomous Chasing Robot" *University of Central Florida*, Web. 2016. retrieved from
http://www.eecs.ucf.edu/seniordesign/fa2014sp2015/g19/files/SD1_paper.pdf

[19] Durrant-Whyte, Hugh. *Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms*. Web. 2016. retrieved from
https://people.eecs.berkeley.edu/~pabbeel/cs287-fa09/readings/Durrant-Whyte_Bailey_SLAM-tutorial-I.pdf

[20] Blas, Morten Rufus. "SLAM for Dummies." Web. 2016. retrieved from
https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf

[21] Eitel, Elizabeth. "Basics of Rotary Encoders: Overview and New Technologies." *Sensors Content from Machine Design*., 7 May 2014. Web. 2016. retrieved from
http://machinedesign.com/sensors/basics-rotary-encoders-overview-and-new-technologies-0

[22] Unknown, "New to Encoders." *New to Encoders Encoder Products*. Web. 2016, retrieved from
http://encoder.com/new-to-encoders/

[23] Davide Migliore, (2009) *Use a Single Camera for Simultaneous Localization And Mapping with Mobile Object Tracking in dynamic environments*
http://www.rawseeds.org/home/wp-content/uploads/2009/10/Migliore_et_al_Workshop_ICRA_20091.pdf

[24] Chieh-Chih Wang, (2007), *Simultaneous Localization, Mapping and Moving Object Tracking*
https://www.ri.cmu.edu/pub_files/pub4/wang_chieh_chih_2007_1/wang_chieh_chih_2007_1.pdf

[25] Intro to Robotics, (2013), *Fundamental Guide for Stereo Vision in Robotics*
https://www.intorobotics.com/fundamental-guide-for-stereo-vision-cameras-in-robotics-tutorials-and-resources/

[26] Stefano Mattoccia, (2015), *Stereo Vision: Algorithms and Applications*
http://vision.deis.unibo.it/~smatt/Seminars/StereoVision.pdf

[27] RobotShop, *Blackbird FPV Camera*
http://www.robotshop.com/en/blackbird-2-3d-fpv-camera.html

[28] RobotShop, Stereoscopic Nerdcam 3D FPV
http://www.robotshop.com/en/nerdcam3d-mk2-stereoscopic-fpv-3d-flight-camera.html
http://www.eecs.ucf.edu/seniordesign/sp2015su2015/g09/archive.html

[29] Guarav Sukhatme, (2010), Real-time Motion Tracking from a Mobile Robot
http://robotics.usc.edu/publications/media/uploads/pubs/650.pdf

[30] ElectronicsTutorials, MOSFET as a switch
http://www.electronics-tutorials.ws/transistor/tran_7.html

[31] RobotShop, (2011), Controlling Your Robot
http://www.robotshop.com/blog/en/how-to-make-a-robot-lesson-6-controlling-your-robot-2-3688

[32] CircuitDigest, WiFi Controlled Robot
http://circuitdigest.com/microcontroller-projects/arduino-wifi-controlled-robot

[33] Sending data from Arduino to Raspberry Pi
https://www.raspberrypi.org/forums/viewtopic.php?t=111882&p=767549

[34] All about agile, (2011)
http://www.allaboutagile.com/agile-development-cycle/

[35] digi, Zigbee Wireless Standard
https://www.digi.com/resources/standards-and-technologies/rfmodems/zigbee-wireless-standard

[36] Our every day life, Blue-Tooth vs Wifi Power Consumption
http://techin.oureverydaylife.com/bluetooth-vs-wifi-power-consumption-17630.html

Appendix D - Figures and Tables

Appendix D contains a list of all the figures and tables used in this document.

Table of Figures

All figures referenced in the document and their descriptions and request status for permission of use are seen listed below in Tables A-E.

2. Project Description			
No.	**Description**	**Source**	**Permission Status**
2.1	Example of the playing field	Project Supplied	-

Table A

3. Research			
No.	**Description**	**Source**	**Permission Status**
3.1	Outline of the SLAM Process	SLAM for Dummies	Request Pending
3.2	Intricate SLAM Map showing spikes and edges	SLAM for Dummies	Request Pending
3.3	C.RAY Benchmark	Photonix	Granted
3.4	John The Ripper Benchmark	Photonix	Granted
3.5	Smallpt Benchmark	Photonix	Granted
3.6	Performance Per Dollar Benchmark	Photonix	Granted
3.7	An example of a 128x128 pixel image	Team Designed	-
3.8	Benchmark of different vision libraries	O’Reily	Requested
3.9	Original low contrast image	OpenCV	Requested

3.10	Shows Histogram Equalized Output	OpenCV	Requested
3.11	Front View of Chassis	Team Designed	Granted
3.12	Bottom View of Chassis	Team Designed	-
3.13	Front view of chassis	Team Designed	-
3.14	Side view of chassis	Team Designed	-
3.15	Front View of Robot	Team Designed	-
3.16	Bottom View of Robot	Team Designed	-
3.17	Side View of Robot	Team Designed	-
3.18	Completely Assembled Robot	Team Designed	-
3.19	Nerf Rival Zeus MXV-1200	Hasbro	Requested
3.20	CS-18 N-Strike Elite	Hasbro	Requested
3.21	Insides of Rapidstrike CS-18	Team Designed	-
3.22	Insides of Rival Zues MXV-1200	Team Designed	-
3.23	Rival Zues MXV-1200 Modification	Team Designed	-

Table B

3. Research			
No.	**Description**	**Source**	**Permission Status**
3.24	Rival Khaos MXVI-400	Hasbro	Requested
3.25	Insides of Rival Khaos MXVI-400	-	-
3.26	Odometry p1- Example showing the relations of variables	Princeton	Requested
3.27	Odometry p2- Example relating circular α to the orientation	Princeton	Requested
3.28	Odometry p3- Example denoting difference of Δd and Δs	Princeton	Requested
3.29	Logitech HD Pro Webcam C920	Logitech	Granted
3.30	LIDAR-Lite 3 Laser Range Finder	RobotShop	Granted
3.31	Raspberry Pi 3 Model B	RobotShop	Granted
3.32	ATmega328p	oomlout	Requested
3.33	10 A 5-25V Dual Channel DC Motor Driver	RobotShop	Granted
3.34	Futaba S3004 Standard Servo Motor	RobotShop	Granted

Table B-2: Continuation of Table B

5. Hardware Design			
No.	**Description**	**Source**	**Permission Status**
5.1	Power Flow diagram	Team Designed	-
5.2	ATmega168/328 pin mapping	Arduino	Granted
5.3	GPIO pinout for Raspberry Pi 3	Element14	Request Pending
5.4	Forward and Reverse Drive	Team Designed	-
5.5	Left and Right Drive	Team Designed	-
5.6	Inside of a Nerf-blaster	Team Supplied	
5.7	Mechanical switch of a Nerf-blaster	Team Supplied	-
5.8	MOSFET as a switch	Electronics Tutorials	Requested

Table C

6. Software Design			
No.	**Description**	**Source**	**Permission Status**
6.1	High Level Software Diagram	Team Designed	-
6.2	High Level Software Diagram in Neutral Mode	Team Designed	-
6.3	Agile Development Life Cycle	Team Designed	-
6.4	Data Flow Sensors	Team Designed	-
6.5	Manual Navigation Breakdown	Team Designed	-
6.6	Breakdown of Components for Nerf-Blaster’s Pan and Tilt	Team Designed	-
6.7	ATmega328p response to firing subsystem seen as a flowchart	Team Designed	-

Table D

Table E

Table of Tables

All tables referenced in the document are seen listed below in Tables A-G.

Table A

6. Software Design	
No.	**Description**
6.1	Arduino Serial Library begin
6.2	Arduino Serial Library end
6.3	Arduino Serial Library find
6.4	Arduino Serial Library findUtil
6.5	Arduino Serial Library read
6.6	Arduino Serial Library write
6.7	Arduino PID Library - PID
6.8	Arduino PID Library - Computer
6.9	Arudino PID Library - SetOutputLimits
6.10	Arduino PID Library - SetTunings

Table E

8. Prototype Testing	
No.	**Description**
8.1	Camera Testing module
8.2	DC motor testing module
8.3	Stepper motor testing module
8.4	Microcontroller testing module
8.5	Nerf-blaster firing testing module
8.6	LIDAR integration testing module
8.7	Power distribution testing module
8.8	Autonomous Object Detection Test
8.9	Autonomous Facial Detection Test
8.10	Camera Output Storage Test
8.11	Rangefinder Sensor Test
8.12	Manual Navigation Testing module
8.13	Nerf-blaster Pan and Tilt Test
8.14	Input from Raspberry Pi 3 Model B Test
8.15	Input from Encoders Test
8.16	Raspberry Pi Receive Input from Microcontroller

Table F

9. Administrative	
No.	**Description**
9.1	Fall milestones
9.2	Spring milestones
9.3	Project budget for main components
9.4	Project budget for complements of main components

Table G