BRO-NS

Bluetooth Remote Operated Nerf Sentry

EEL 4915: Senior Design II
University of Central Florida
Department of Electrical Engineering and Computer Science

Group 24

Ross Ellis - Computer Engineering
Daanish Habibi - Computer Engineering
Juan Landron - Computer Engineering
Kyle Quarti - Electrical Engineering

Table of Contents:

1. Executive Summary

In the early 2000's we grew up playing with different toys that set memories ingrained into our brains. One such pastime that we developed a strong nostalgia for was NERF, owned by Hasbro. These toys range from different kinds of foam balls for sports, different darts and guns, and their supers soaker water guns. We enjoyed these toys very much as children as did millions of others. This is why our team was eager and excited to work on this project.

The main goal of this project was developing a prototype of a bluetooth remote operated NERF sentry that can also utilize facial recognition software to support autonomous firing upon locking onto a given target. This type of turret can be purchased and used by average consumers in homes by children or adults. The types of technology it uses can be taken in and made use by law enforcement or military units. It is capable of rotating itself on axes in different directions while drawing little power from the battery pack installed. This makes the turret more on-the-go friendly as it is light and portable while retaining its wireless connectivity. These features enable the turret's scalability in future use and application.

This project gave our team enough complexity to challenge ourselves through a combination of hardware and software, retaining a consumer level budget to create a popular concept that is user friendly. The user has the capability of connecting their own smartphone wirelessly to the turret and fire upon a target that has been selected by the facial recognition software or manually acquired after the target has entered the turret's field of view. It will continue to scan objects and the area within its view until another target has been acquired.

Deployment can be completed by a small crew of team members and maintain accurate firing upon targets at center mass, within range based on distance parameters as accurately as possible while conserving its ammunition supply. The user is able to leave the sentry on its own to be a proximity security device as an unmanned guard. The computer imaging technology in facial recognition is able to recognize an enemy based on its face to lock onto. BRO-NS is an enjoyable project with interesting hardware and software component elements that can be applicable in real companies or organizations. It has interchangeable parts and components for others' ease of use. We aspire this project and its application be used by consumers in business or personal settings that would provide enjoyment and or security.

2. Project Description

2.1 Project Motivation and Goals

The primary motivation for this project was to create a remote controlled nerf turret and have fun shooting targets. We all grew up playing with nerf guns so we figured we could design and build one in the form of a sentry, whilst being controlled by Bluetooth software, which we make use of in our everyday life. This project blends our skills with hardware and software within our majors, seamlessly. Turrets such as this sentry can be utilized by law enforcement and military units for professional, non-lethal simulation. This project is complex enough to challenge our skills in design and application whilst being cost effective in its requirements and in time. Our decision matrix shows the types of project designs we considered and how we came to this final decision.

The primary goal of this project was to design and build a functioning prototype of a Bluetooth controlled Nerf sentry turret that can fire at selected targets accurately and efficiently. It can be to connect to our smartphones, rotate in multiple directions, use a small amount of power while untethered, and reload magazines. We designed this sentry to be lightweight and portable, allowing you to take it on the go and place it where you like to then control it with the wireless connectivity. Its practicality and ease of use is that of high power in launching projectiles and only drawing a small amount of power.

The Nerf sentry is eligible to be connected wirelessly to a smartphone in order to make use of software and hardware, enabling it to rotate in different directions on a dual axis chassis, and fire at the target selected by the user after rotating to point at it. Once the user has fired the desired amount of ammunition at the target, the user can rotate to select another target and continue firing. In addition to being able to be manually controlled, our turret is also equipped with a means of automatically detecting potential targets in its line of fire. By implementing various computer vision technologies, we found a suitable solution that allows the turret to detect threats by itself, while still allowing the entire system to remain inexpensive, and mobile.

This project is complex but practical while being budget friendly. Our team was informed of this project's idea's popularity after we thought of it but we still wanted to take on the challenge in our own way. We were deciding between using airsoft, paintball or Nerf as the firing mechanism and ammunition type source. Airsoft and paintball would be more accurate than nerf but more costly, while the latter also being messy. Previous years' versions of this project included face recognition software and color detection software, connected via Bluetooth or wifi.

2.2 Objectives

The main objective of our project is to design and build a prototype sentry turret that will be both cheap to manufacture and in a small enough package that it is still portable. We want our turret to be able to detect any potential threats that come into range, and then be able to engage that target if need be. We would also like to have manual targeting functionality by enabling the turret operator to control the turret's axial movements and firing mechanism with a wireless device. By building our turret out of affordable and easy to access materials, we both minimize the total cost of production as well as make finding replacement parts easy. As far as portability, we want the turret to be easily moved and operated by a single person. By designing the turret to be set up by a single person, the manpower typically needed for a larger weapons system can be utilized elsewhere. Because our turret is designed to be carried and operated by a single person, making the turret system as lightweight as possible will minimize the risk of injury.

One of the other core objectives of our project is the automatic target acquisition. By allowing the turret to scan, detect, and engage potential targets, this ensures that the turret operator is kept safe if they are unaware of a potential threat. This would also allow the turret to be placed and then left to operate in full sentry mode. This application of the sentry would prove useful as a form of proximity perimeter security, allowing posts traditionally taken by a manned guard to be performed by the turret. In order to demonstrate the target detection of our sentry we designed a system that will allow the sentry turret to distinguish ally from foe. Because of the computer imaging technology we used, the turret has the ability to recognize specific shapes or colors as an enemy and have the turret lock-on and fire at targets with that specific shape or color on it.

We also wanted our turret to be completely battery powered. Having the turret powered by battery will maximize its portability by allowing it to operate in places where a conventional 120V outlet can’t be accessed. While battery power brings many positive aspects to the project, there are also many negatives. With a traditional DC power brick, all of the voltage regulation to decrease the high voltage coming from an outlet to the low voltage required by the MCU and the other sensitive circuit components is done inside the power brick. Because our turret is powered by a battery pack, the voltage regulation has to be done by a circuit we designed to ensure that every component is getting the correct amount of power it needs to operate. Because of the limitations potentially imposed by powering the entire turret by batteries, we wanted to ensure we are using technical components that will consume a minimum amount of power. This will maximize the battery life of the turret and allow us to divert more power to things like the firing mechanism to enable our turret to have a longer range.

2.3 Requirement Specifications

- Wirelessly controllable
- Wireless range of ~ 10m
- 80% accuracy within 5m
- 50% accuracy within 10m
- Use less than 30W power
- Is deployable in under 3 minutes
- Weighs less than 10lbs
- Can operate for at least 10 minutes on a single battery pack
- Can run completely untethered with no cables connected to outside devices for power or data transmission
- Can identify and fire on a target within 10 seconds of gaining vision on suspect
- The turret has a horizontal range of motion of at least 135°
- The turret has a vertical range of motion of at least 60°
- Turret has the ability to complete a sweep on its horizontal aiming axis in 3 seconds
- Turret has the ability to complete a sweep on its vertical aiming axis in 2 seconds

With these design specifications, our goal was to make the wireless sentry portable and safe for a crew of people to maintain and operate. The turret should be controlled completely wirelessly so that it can be deployed in a dangerous area, and allow the operation team to retreat to a safe area. By setting accuracy specifications, we can set a target range at which the turret will wait to fire. If the target is outside the range that would allow the turret to take an accurate shot, the turret can track the target and wait for it to come within range to fire. When the target does come into firing range of the turret, having the turret quickly lock onto the target and fire will ensure the most accurate shot. The maximum projectile distance parameter is set so that we can ensure the turret will be able to fire a projectile with enough force to maintain an accurate shot. The projectile speed parameter also plays a role in accuracy by allowing the turret to adjust its aim, fire, and have the projectile reach its intended target before the target has moved a substantial distance and potentially requiring the turret to readjust its aim and fire again. Because a firing mechanism was not able to be sufficiently developed, an alternative method of demonstrating target acquisition and engagement was used for the turret demonstration. We decided to have the turret aim at a potential target with a red laser when the target enters into the proximity of the turret's camera system. If the turret deems the target a potential threat, a light could be flashed indicating a shot has been fired at the target. If the target is deemed an ally, then a green light can be illuminated indicating the target can pass.

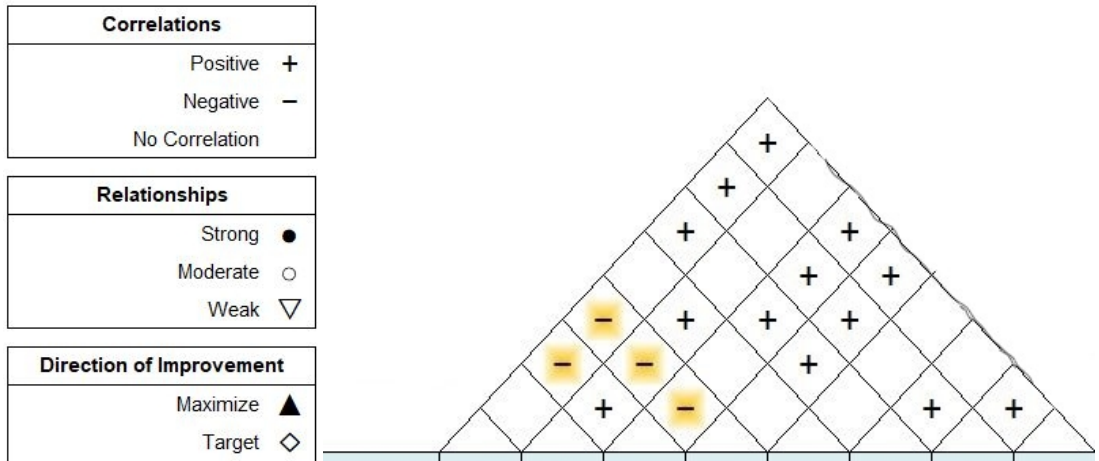
The specifications for the range of motion allow the turret to engage targets in a multitude of positions. The horizontal range of motion allows the turret to identify and engage targets with a range of motion approximately equivalent to a stationary human. The vertical range of motion allows the turret to engage targets positioned above and below where the turret is deployed. We have also set pan and tilt speed specifications to ensure that the turret will be able to locate and track targets traveling at sufficient speeds.

Having a magazine for the turret allows the turret to engage multiple contacts, or fire multiple times at a single contact before having to reload its ammunition supply. We also want to ensure that the turret's magazine can be replaced quickly, allowing an operator to swap it quickly and then retreat back to safety. Because of the compact design we implemented, it is important that we make our turret functions as accurate as possible so that ammunition is not wasted by missing shots on a target.

Our power usage target of 30W was made so that the turret could be designed to run at a low enough power to allow it to be run on batteries instead of relying on an external AC/DC power source. We also want the portable battery source to allow the turret to operate for a substantial amount of time without needing to recharge or have a new battery swapped in.

Having a quickly deployability time allows a crew to set up the turret quickly and operate it from a safe area. By designing our turret to be as lightweight as possible we significantly increase its portability by allowing it to be carried by a single person. We want the turret to run without the need of any external wires needed for power or data transmission. This will ensure that the turret can be deployed in remote locations and operated from a safe distance.

2.4 House of Quality Analysis



Column #	1	2	3	4	5	6	7	8
Direction of Improvement	▲	◇	▼	◇	◇	▼	▼	▼
Customer Requirements (Explicit and Implicit) \ Engineering Requirements	Control Distance	Fire Range	Power Draw	Operation Time	Accuracy	Weight	Dimensions	Cost
Wirelessly Controllable	●	▽	○	▽	▽	○	○	●
Accurate	▽	●	○	▽	●	▽	▽	○
Multiple Round Magazine	▽	▽	▽	▽	▽	○	●	○
Battery Powered	○	○	●	●	▽	●	●	○
Deploy Quickly	▽	▽	▽	▽	▽	○	○	▽
Target Identification	▽	●	●	●	●	○	○	●
Ease of Use	▽	○	▽	▽	○	●	●	○
Portable	▽	▽	○	▽	▽	●	●	▽

Table 1: House of Qualities

The chart shown in the previous page is our House of Qualities (HOQ) table. The HOQ takes customer design requirements and compares them to the engineering design constraints. The level of correlation between the customer's desired specifications and the engineering constraints is shown in the lower portion of the table. Several of the customer's design specifications have little to no impact on the engineering design requirements of the turret. For example, wanting the turret to have a multiple round magazine has no impact on the first five engineering requirements, and will only slightly affect the overall weight of the turret. However, having to account for the storage of multiple projectiles will significantly impact the overall size of the chassis for the turret. Some of the more prominent customer requirements that will strongly impact the engineering design requirements are the need for the turret to be battery powered and for the turret to have target identification. Relying on battery power for our sentry turret will mildly impact the control distance and firing range of our turret as more power may need to be supplied to the wireless communication device for range, and a longer firing range will require more energy to be put into the projectile. Implementing target identification into our sentry turret will most significantly limit the firing range of the turret. Because the turret will be identifying targets with an external camera and imaging software, the range of the turret is now limited to how far the camera can see instead of how far the projectile will be able to travel.

Immediately above the engineering requirements section is the desired direction of improvement. The direction of improvement shows what we, as engineers, want to maximize or minimize about our design. Because of the customer design specifications we want to minimize the power draw so it can power effectively, the weight and dimensions should be minimal to increase portability, and the cost should be minimized to make it financially possible. There are targets set in place regarding the range of fire, operation time, and accuracy, so as long as those standards are met they do not need to be maximized. Lastly, the wireless control range of the turret should be as high as possible so that it can be controlled from a safe distance.

At the very top of the table is the pyramid shaped correlation chart. This section shows how much the engineering design constraints correlate with each other. The first two negative correlations observed when looking at the chart from left to right are between control distance and power draw, and control distance and operation time. As the wireless control distance is increased the total operation time will likely decrease due to increased power demands required by the wireless transmitter. While some positive correlations may benefit our project, there are some positive correlations that have an overall negative consequence. When observing the last positive correlation on the left between the turret dimensions and the overall project cost, as the turret requires more materials to be built, the total cost of the materials is increased.

3. Project Research

3.1 Existing Similar Project and Products

Designing and implementing an automated Nerf sentry is no new concept and this proposition has been made numerous times through website and video tutorials across the internet. They all involve a similar standing turret that can either be pointed at a target or recognize it, lock on to the target and fire upon it. There have been many senior design projects predicated on this exact concept that involve software implementing color recognition, facial detection, facial recognition, shape recognition and others. Now all of these automated sentry turrets are beyond the budgeting that our BRO-NS sentry would be given. The research time and technology are ultimately too deep and sophisticated to understand with the scope of this project along with the fact that almost all the information regarding these technologies' designs and implementations are most likely classified.

Figure 1: Phalanx CWIS

3.1.1 Phalanx CIWS

As of currently, automated sentry turrets are used around the world, primarily by military defense in some countries. They are not readily available to be sold in the market to the average consumer. These sentries don't use recognition to determine if friendly or hostile such as it would with facial recognition. Their target identification functions off of motion such as in the U.S. Navy's Phalanx CIWS which contains a search radar that detects if the target's range is increasing or decreasing in relation to the ship. It then checks if the target is able to strike the ship based on its velocity and trajectory.

Figure 2: SGR-A1

3.1.2 SGR-A1

In South Korea, the SGR-A1 is deployed along the Demilitarized Zone to assist troops using a system that integrates surveillance, tracking, firing and voice recognition. South Korean Firm DoDAAM showed the Super aEgis II which can lock onto vehicles or humans with thermal imaging, unaffected by night or weather conditions, up to 3km away. Israel has implemented something similar, deployed on the Gaza border but requires a person to give input to fire.

3.1.3 PROWLER

In the 1980's the US began an experimental project of a robot mobile sentry gun device known as Programmable Observer With Logical Enemy Reponse, or PROWLER. It was the first robotic sentry surveillance system and was for outdoor use, equipped with twin M60 machine guns and even a grenade launcher. It had a sensor array to detect targets and the visual representation of this turret resembled that of a small tank. It's manufacturer, Robot Defense Systems, in Thornton, Colorado has unfortunately gone out of business.

Figure 3: XTR-101/102

3.1.4 XTR-101/102

The National Chung-Shan Institute of Science and Technology of Taiwan developed the XTR-101 and XTR-102 which are a pair of 20mm remote weapon system sentry guns. This Short-Range Automated Defense Weapon System, or SADWS, can rotate 360 degrees at an astonishing 60 degrees per second with elevation ranging from 15 degrees to 80 degrees and features fire control, imagery identification, and target tracking. The Republic of China Navy has deployed many of them and plans to order more.

Figure 4: Autonomous Turret Mountted Sentry

3.1.5 Autonomous Turret Mounted Sentry Fall 2018

The Autonomous Turret Mounted Sentry was the senior design project of Stephen Greene, Anthony Speake, Emmanuel Sotomayor, and Conan Wilson during the Fall 2018 semester. This project consisted of an autonomous wheel-based robot called Tankbot, and is very similar to ours. It contains an automatic airsoft rifle, LiDAR scanner, and Pi

Camera V2. It is remote controlled and streams video to a remote device with Wi-Fi capability. It has real time position tracking

3.2 Relevant Technologies

The mechanical parts include the dual axis chassis board, the tripod, and the firing mechanism in which this group may purchase and use a nerf/airsoft gun to be attached or design and build the firing mechanism from scratch. There are several mechanical and electrical components that must be researched to find the best application for use in our project. Among these are a large variety of motors, and several options for both battery power and standard 120V residential power. A brief breakdown of these potential components will be discussed below with a more technical dissection in *section 3.3 Strategic Components and Parts Selections.*

3.2.1 Motor Varieties

Stepper - Stepper motors are among the most common types of DC motors. A stepper motor gets its name from how it operates. The gears inside of a stepper motor align with a series of jagged electromagnets that are typically set up in 90° phases. When a current is applied to one of the specific electromagnets, the teeth of the central gear become aligned to that specific magnet. By having the electromagnets turn on and off in a clockwise or counterclockwise order, the teeth become aligned to whichever electromagnet is currently being powered. Because the central gear inside of the motor is typically only a couple of centimeters in circumference and has dozens of teeth, each individual pulse of the electromagnets results in a very small movement of the motor output shaft. While each step of the motor may result in a very small output movement, the electromagnets can be switched rapidly on and off many times a second. Because of the nature of how the gear train inside the motor works, stepper motors move very linearly and have a constant acceleration. Some advantages of the stepper motors are that they are one of the most simple and inexpensive options for small load application motors. Some disadvantages are that they do not utilize any feedback mechanisms, as they rotate faster they lose output torque, and they do not react to changes in load.

Brushed - Brushed DC motors use a system of wound coils and two pole electromagnets two translate electrical pulses into rotational energy. By causing the electromagnets to cycle, the central barrel of the motor called the armature, is rotated to align with the pole orientation of the magnets. Brushed DC motors are extremely cheap and can have their movements tuned more easily than stepper motors. Brushed motors are typically not very reliable and do not have a very long life span relative to other varieties of DC motors.

Brushless - Brushless motors operate similarly to brushed motors with the exception that brushless motors utilize a permanent magnet on the external rotor, and operate with three phases of coils. Inside of the motor there is a sensor that tracks the position of the rotor that way electricity is sent to the coils at the appropriate time. Brushless motors typically have higher operating loads than other types of simple DC motors, and they can operate well at a variety of speeds. Brushless motors are more complex than both brushed and

stepper motors, and they rely on some type of feedback to control the output torque of the motor. Brushless motors also cost more than their representative stepper and brushed counterparts.

Servo - Servo motors operate similarly to brushless motors with the exception that the permanent magnet is on the internal rotor and the three phase windings are on the external rotor. As current is run through the windings an electromagnetic field is generated pulling the permanent magnet in the central rotor to align with the magnetic field. Servo motors require tuning to get the movements dialed in for specific applications.

3.2.2 Power Delivery

One of the other primary areas of research required for our project is figuring out what kind of battery technology will best fit our project. Some of the key points that determine how useful a battery will be for a given application are the batteries geometry, its power density, its energy density, and its recharge efficiency.

Battery geometry is the easiest to determine as different battery formats and constructions require different sized cells to reach their designed voltage output characteristics. Something like a lead-acid battery most commonly used in automotive applications is significantly larger than a common AA size battery and most likely wouldn't be fit for anything that a AA size battery would be used to power.

The next criteria used to determine a battery's functionality is its power density. The power density of a battery is how quickly a battery is able to discharge its stored load. Batteries with a high power density are more useful in applications where a large amount of power is needed quickly, but not for any sustained duration of time. An example of a battery technology with a very high power density are super capacitors. Super capacitors are able to store large quantities of energy and release them very rapidly.

Energy density is similar to power density but is instead focused around how much energy a battery can store, not how quickly it can release its stored energy. Batteries with a high energy density are capable of storing a large amount of energy, but typically can only release the energy slowly. Rechargeable batteries that also have a high energy density tend to take significantly longer to recharge to their full capacity than their high power density rechargeable counterparts. Lead acid batteries are a good example of a battery with a high energy density. They can store a lot of energy but dissipate their energy more slowly and take many hours to recharge.

The last criteria for determining the effectiveness of a battery in a given application is its recharge time and efficiency. Recharge efficiency is how much of the battery's total capacity can be effectively recharged after a discharge cycle. A good recharge efficiency ensures that after several discharge/charge cycles the battery will still have the same effective capacity as when it was brand new. Another key factor of a battery's recharge capabilities is how many recharge cycles it can handle. Ideally, if a battery is being used

in an application where it is designed to be recharged, you want a battery that can be recharged many times. This effectively increases the lifespan of the battery and also plays into the recharge efficiency of the battery.

C-Rate - The C-rate of a battery is a rating given to batteries that is used to determine what the maximum charge and discharge rates of that battery are. The C-rate is used in conjunction with the battery's capacity to calculate the total peak number of amps in which the battery can handle being output or put into it. A battery with a high C-rate will be rated to withstand a higher level of charge or discharge current than a battery with the same capacity but a lower C-rate. The C-rate of a battery is used to calculate the peak current of a battery using the following set of equations.

$$Battery\,Capacity \in Amps \times C\ rate = Peak\ input/output\,Current$$

If two batteries have the same capacity but different C-rates, the battery with the higher C-rating will have a higher rated peak current. The typical range of C-rating for smaller form factor batteries is usually 1C to 40C. Batteries with a very high C-rating will be able to charge more quickly, as well as deliver a lot of power (depending on the voltage rating) in a short period of time. As the C-rating of a battery is increased, there tends to be an increase in waste heat energy as well. Standard AA lithium polymer batteries are typically rated sub 1-C as they are not designed for high current applications. Lead-acid batteries have extremely low C-ratings usually around 0.05C, thus are required to be trickle-charged where a constant low current charge current is applied over the course of several hours.

Lithium Ion - Lithium ion batteries are commonly used in small form factor electronics where rechargeability and energy density take priority over power density. Lithium ion batteries most commonly come in the standard AA, AAA, D-cell, and C-cell form factors. Lithium ion batteries also come in pouch form factors like the kind that would be commonly found in a cell phone or a laptop. Lithium ion batteries are not able to store or output large quantities of energy, but due to their small form factor, they have very high energy densities typically around 100-150 Watts-per-kilogram (Wh/kg) and around 400-500 Watts-per-liter (Wh/L). Although Li-ion batteries do not have a high power density and can typically only output 1.5V to 3V, due to their small size, several batteries can easily be connected in series to boost their output voltage while still maintaining a small design footprint. Another benefit of Li-ion batteries is their rechargeability. Li-ion batteries can be recharged hundreds of cycles during their lifespan and can typically charge within two hours. The main drawback to Li-ion batteries is that they are not ideal for use in environments with extreme temperatures. Both extremely cold and extremely hot climates will significantly affect battery discharge rate and can impact the lifespan of the battery. Another negative aspect of the Li-ion battery is that it is very sensitive to overcharging. Inside the charger or the device utilizing a Li-ion battery there must be a controller that cuts off the charge current to prevent the battery from overcharging. Overcharged Li-ion batteries become bloated and deformed and can become at risk of rupturing.

Nickel Metal Hydride - NiMH batteries are widely available in the same form factors as Li-ion batteries. NiMH batteries have lower power and energy densities than Li-ion batteries, around 50-100 Watts/kg and about 250-350 Watts/liter, but are much more capable in extreme environments. In terms of rechargeability, NiMH batteries are only able to endure approximately half of the recharge cycles of Li-ion batteries.

Lithium Iron Phosphate (A123) - LiFePo batteries are technically a specific variety of Li-ion battery. LiFePo batteries have a higher energy density than NiMH batteries but have a lower energy density than Lithium-Cobalt Li-ion batteries around 150 Watts/kg and 300-400 Watts/liter. LiFePo batteries have among the highest rechargeability lifespan as they can typically be recharged anywhere from 1000-2000 cycles, about four times that of NiMH batteries and twice that of Lithium-Cobalt Li-ion batteries.

Lead-Acid - Lead acid batteries are commonly found in automotive applications like car batteries. Lead acid batteries have by far the lowest energy density of any of the battery types we have considered for this project, typically only about 50 Watts/liter of 50-100 Watts/kg. Lead-acid batteries also have the worst rechargeability of the battery types mentioned as they only have a recharge life of 50-100 cycles. One of the main drawbacks of the lead-acid battery is its size.

3.2.3 Wireless Communication

Due to the myriad forms of communication it's tough to choose one over another due to each one's benefits and disadvantages. Some of these are satellite communication, Radar, Mobile Telephone Systems (Cellular Communication), Infrared Communication, Bluetooth, WLAN (Wi-Fi), Radio Frequency Identification (RFID), and NFC.

Satellite Communication - This form of communication is based on devices orbiting space in order to have international communication. The cost will vary depending on what resources are used to make the connection utilizing a satellite, such as an open satellite connection or creating one yourself. The upside is that the connection is international and it has high bandwidth. On the other hand the downside is that the distance may cause distortions in signal, high maintenance cost, and it needs to be monitored to check the situation of the satellite.

Radar - This form of communication is mainly used for detecting objects in the distance. It cannot however allow us to communicate with the robot in order to give it instructions. Although we can't directly communicate with the robot we are able to use radar in order for the turret to be able to sense objects and potentially determine to shoot or remain idle.

Cellular Communication - This is one of the most commonly used forms of communication to most people nowadays. The newest development being the 5g which allows fast wireless speeds and connects users worldwide. This would allow us to remotely operate the turret at a further distance given that there are antennas in the area.
Infrared Communication - Infrared (IR) utilizes infrared LEDs that produce light that pulses in a pattern in order to send out a signal. These signals are sent out as pulses from

one device to another, where it is received and processed. Usually there is no encryption and the speed of this form of communication is fairly slow. This is an inexpensive form of communication, however it does require a direct line to the target and the range is short.

Bluetooth - This is a wireless communication technology that utilizes radio frequency in order to send and receive data. Bluetooth is able to operate with lower power than wifi but has a smaller range of operation given the devices that are being used to transmit the signal.

Wifi - The most common form of wireless communication to most people is Wifi. Although it consumes more power than bluetooth it has a much farther range. There is also the support of multiple users whereas bluetooth does not have as many.

NFC - Stands for Near Field Communication. NFC is usually used in order to transfer information from close quarters. This allows a more secure process due to the range of transmission. Unlike RFID, NFC is able to allow two way communication.

RFID - Is an inexpensive form of communication similar to NFC. The range of an RFID is farther than that of an NFC even though they are both low power. Although they are similar in nature, RFID only allows communication in one way rather than a bidirectional form of talking.

In the end each form of wireless communication is dependent on the use. If a person requires multiple users on a device to connect them to the internet then wifi might be a thing to consider. When referring to setting up a wireless turret, there are many things to consider. Some of these are whether the turret will be in sight and even the distance which will most likely be the most important aspect. The reason distance will play the biggest role in choosing a form of communication is because a turret may be operating in another area that is hundreds of miles away, in that case NFC or RFID won't be a sufficient form of communication. Therefore the option that we have decided to use is bluetooth as it is low power and allows for a fast enough connection to view the camera and control it remotely while at the same time giving us the option to control its facial identification as well.

3.2.4 Memory

Memory is essential for the performance of the microcontroller for it holds the code, driver libraries, and registers that hold the data from either peripherals or other data. Without memory, the microcontroller cannot hold any type of information which means that there is no way it will know how to function. There are three types of memory that are used in microcontrollers, these are RAM, ROM, and Flash memory.

RAM - RAM is an acronym for Random-Access Memory, which is a form of memory that is volatile, thus does not store data permanently. The main use of this type of memory is to hold onto information that is not important to keep and just necessary for a

short period of time, an example being the microcontroller does not need to keep any previous location of the firing mechanism because there is no value in holding it since it will focus solely on the current target needing to fire. The moment the RAM forgets the value depends on what type of RAM it is. The RAM can either be static or dynamic. In a static RAM (SRAM), it will forget the data it holds the moment there is no more power. The other is the dynamic RAM (DRAM), which holds the data for a period of time even without power but only for a certain amount of time. In this project, the microcontroller will have an SRAM as the DRAM gives no added benefit in terms of the turret design presented.

ROM - The Read-Only Memory is a memory module that stores data permanently without the ability to change the value. This type of memory is good for keeping the driver libraries since those hold meaningful functions that provide the programmer with the ability to utilize the peripherals of the microcontroller and design it in the way it is intended to function.

Flash Memory - The Flash memory is a non-volatile memory, the opposite of the RAM and as such can contain data. It is also reprogrammable so that it can change its data when new information is sent to be stored in the same memory space or be erased completely as well. It is because of this nature that it is the memory that holds the code and the storage of any data needing to be kept whether the microcontroller is on or off.

3.2.5 Communication Protocols

UART - An acronym for universal asynchronous receiver transmitter, is one of the main communication protocols for devices to send and receive data from. It's popularity comes from its simplicity and only needing two wires, a receiving and transmitting wire. The simplicity is because the transmission of the data does not need to have a clock that both devices need to follow, but rather a set, and configurable speed, known as the baud rate. Note that both devices must have the same baud rate as the timing of data traversal and retrieval are determined by the baud rate. The baud rate is the number of bits per second transferred, through serial transmission, and is the reason for why there is no need for a synchronized clock for both devices, as both have the same rate of transfer.

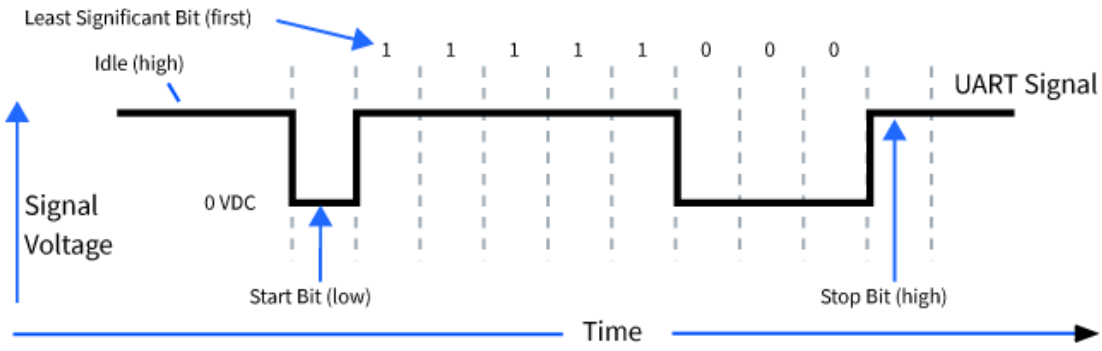


Figure 5: UART Control Signal

The transfer of data goes in the form of a data packet, in which this packet holds the start bit, data frame, parity bits, and stop bits. The start bit is the signal that changes the data transmission line from high to low voltage for one cycle, which signals the receiving end to start acquiring the data. The transmission from high to low signals the receiving end because the transmission wire is set to high when it is not transferring data. The data frame holds the information in bits of 5 to 8 bits long with the parity bit, or 5 to 9 bits long without the parity bit. The transfer of data between the two devices starts from the data frame's least significant bit to the most significant bit. The parity bit is an optional bit that allows the user to determine any changes in the data due to interference either through environmental factors or communication errors such as different baud rates. The parity bit works by being the value of even or odd number of 1's in the data frame. If the parity bit is a value of 0, then the data must have an even number of 1's, else if 1 then it should have odd. UART then compares the value of the parity bit to the data and will determine that if the parity bit describes the data correctly, of either being odd or even, then the data is correct, otherwise it was affected by some changes. Finally, the stop bit is the end of the data packet that signals the UART that all the data has been sent. The receiving end will then be set from a low to a high voltage value for the one or two bit duration, determined by whether the stop bit is one or two bits.

I^2C - This communication protocol transmits data similarly to the UART with two wires. However, the difference lies in the ability to perform data transmission on multiple devices. The devices are classified into a master and a slave, where the master clocks the bus, addresses the slaves in order to differentiate these devices, and writes and reads to the slaves' registers. This protocol can support multiple master's and slave's depending on the unique addresses and capacitance of the bus.

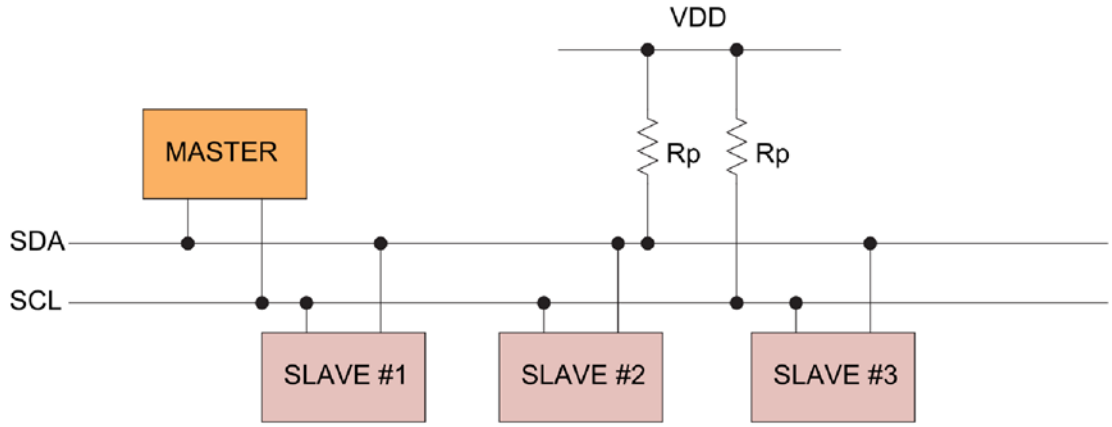


Figure 6: I^2C Model

When sending data, it first sends the address of the slave with a 7-bit address and another bit that determines whether the slave will be written to or read from. If the master sends the slave a read command, the slave will take control of the data line and send a byte of data at a time. After a byte, the receiving device sends an acknowledge or not acknowledge to determine whether the data was sent properly or not. The master also

sends a not acknowledge for slaves it no longer wants to read data from. The data is sent in the 8-bit most significant bit format.

SPI - This communication protocol is similar to I^2C with the exception that there can only be one master and the number of slaves are limited to the number of pins in the SPI module that each of them connect to. There is the exception which is the daisy chain configuration, however, in which the master is connected to a first slave and the consequent slaves are connected to each other in series. The master in SPI is the same as I^2C in which it supplies the clock signal through a clock line. The difference is, as stated earlier, there is a chip select which is connected to each separate device to determine which slave will communicate with the master. There is also the MOSI and MISO, where MOSI is master out slave in and corresponds to the master sending data to the slave, while MISO or master in slave out which corresponds to the slave sending data to the master.

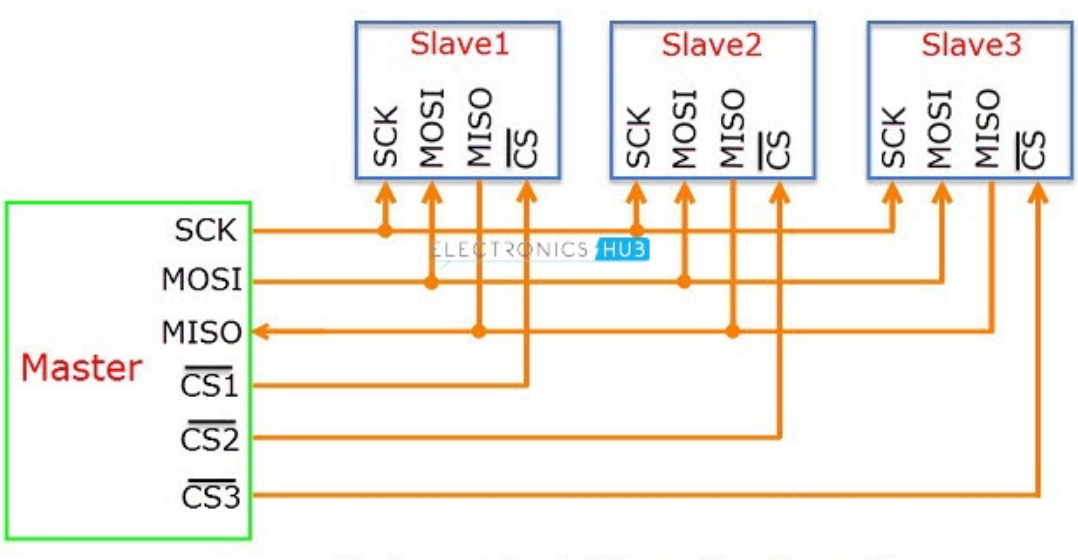


Figure 7: SPI topology

Similar to the other protocols, 8-bits are transmitted to and from the master and the slave. When only a byte is sent to one receiver, the receiver needs to send a bit of data even if it is garbage.

Comparison: For the communication protocol, the UART is definitely the easiest choice while the I^2C is largely expandable and SPI has the fastest speed. The communication protocol that would be mostly used would be the SPI, for the fact that it has the fastest speed which would improve the efficiency of the turret, as well as the turret does not need many devices for communication as the only required devices are the camera, for the computer vision, and the bluetooth controller for the individual control. I^2C is not needed as there is no need for extensive expansion and the two devices for the SPI may not prove to be complex enough that it would warrant the degradation of speed.

3.3 Strategic Components and Part Selections

This section is devoted to the research of components and the justification of why we would use each of these components. The components are divided based on the necessary component necessary for our project, the autonomous turret and thus we will describe details of the specification and the idea of what each component will bring that will be incorporated into the overall design of the turret that will function as it is intended.

3.3.1 Microcontroller

The microcontroller is one of the most important devices in a project. It is the brains of the operation and it provides a sophisticated program to control peripherals to do useful applications. Most applications around the world use microcontrollers due to the need for a dedicated device that does specific tasks and focuses on those tasks. Microcontrollers consist of the processor(s), memory, crystal oscillators, and I/O peripherals all under a single IC chip and work in tandem to give the device the means to produce the job it is intended for. They are used in many applications such as microwaves, refrigerators, cars, etc. For a project, the microcontroller is one of the key technologies to look into for what is needed for the project. In the case of an automated turret, many of the things we are looking for is suitable memory space for the programming of the device so that it can deliver commands to the peripheral I/O devices as well as determine clock speeds, communicate with other modules or microcontroller(s), and optimize the longevity of the microcontroller battery power through the use of low-power modes.

3.3.1.1 MSP432P401R

The MSP432P401R processor is part of TI's SimpleLink microcontroller platform, which provides many resources and development environments for software. There is also the devices' compatibility for the usage of Wi-Fi and Bluetooth, which would be valuable for the wireless controller for turret operations when set to function by a human being. The MSP432 has a ARM Cortex-M4 processor which has the ability to operate up to 48MHz. The architecture will be discussed in section 3.4 Possible Architecture and Related Diagrams, in the next sections we will describe what this microcontroller has to offer.

3.3.1.1.1 Memory

The MSP432P401R has 4GB of address spaces which are divided up into eight 512MB spaces. The Debug/Trace Peripherals memory space which holds all the registers associated with the operation of debugging and other functionalities such as the reset controller, system controller, NVIC, debug control registers, and other peripherals of the Cortex-M4 microprocessor and its functionality. The Peripherals memory space holds all the registers that are used for the programming and operation of the tasks the user wants to be done, such as the UART, SPI, I^2C, Timers, Interrupts, and other registers. The Code memory space holds flash, ROM, and internal SRAM. The SRAM memory space is self explanatory, it is where the SRAM is held. The internal SRAM of the Code memory

space is aliased to have accessibility to the SRAM memory space for efficient access of data. The rest of the memory spaces are unused.

3.3.1.1.2 Specification

The absolute maximum ratings for the MSP432P401R microcontroller is a voltage value of 4.17V for the DVCC and AVCC pins with the difference between the two pins being a max of 0.3V. The datasheet states that if there is a difference above the 0.3V or the maximum rated voltages, the device will likely malfunction. The voltage to any pin should have a max of 4.17V and the max storage temperature of the device is 125℃. The recommended value for the supply voltage VCC is 3.7V and the recommended temperature in free-air is 85℃. The recommended external component values for the microcontroller are the capacitor on DVCC, capacitor on the VCORE, and the capacitor of the AVCC pin should be 4.7μF.

3.3.1.1.3 Low-Power Modes

The MSP432 power saving mode is called the low-power mode. These modes are used to reduce power consumption while maintaining the operation of the MCU. These are useful for projects that don't have a consistent input source of power and rely solely on the power supplied by a portable means such as a battery, or for general power saving, especially for projects that have massive necessity of power, it is always best to optimize the performance of the device to reduce consumption.

The active modes, or the normal operating modes of the MCU, are divided between DC-DC or LDO based, while the low-power modes are based on LDO. This is due to the DC-DC modes offering high current high performance operations from the MCU while the LDO is for efficient low power advantages. The project will heavily rely on the LDO based modes, especially the low power modes due to the necessity to reduce current consumption to maximize the longevity of power given to the device while it is operating. The low-power modes are numbered and the higher the number, the less current consumption and operational components. LPM0 is useful for when the MCU has sent all the instructions for operation to all peripherals and there is no need for the code to be processed. This will leave the peripherals and clock frequencies on a sort of autopilot with their operations already set before the low-power mode was enabled. LPM3 and LPM4 are a more restrictive mode of operation, with the difference being shown to allow the Real Time Clock (RTC) and the WatchDog Timer (WDT) to be on in LPM3, while they are off under LPM4. WDT is a module that performs a system restart when an error in software occurs. The RTC is as the name implies a real clock, which is a module that gives the programmer the functionalities of clock counters such as alarms and calendar times.

3.3.1.1.4 Interrupt

In order to use the Low-Power Modes, there requires a system that allows the MCU to turn back on the processor and have the code run again. This is what is called an

interrupt, a type of exception that occurs to enable the CPU again to allow the system to react in the way that the code asked to do it once an interrupt is called. As said earlier, the usefulness lies in the ability to have an MCU functioning continuously while still operational but wasting only the most essential power necessary. In the MSP432P401R, the register NVIC, or the Nested Vectored Interrupt Controller, oversees these interrupts and handles them. This NVIC is stated to support 64 unique interrupts and 8 priority levels which are numbered from 0-7, where the lower the number the higher the priority. How it achieves this is the code sets exceptions to certain peripherals. When an exception is called the processor state is saved in a stack. Once there is an exception called, the state is accessed and the processor executes the code that pertains to the exception. The ISR, or Interrupt Service Routine, is called to resave the processor state to the stack once the operation necessary has finished. The ISR automatically saves the processor state to the stack as soon as an exception is called.

In terms of the priority, the method of processing priority levels within exception handling is the use of a vector table that holds vector addresses which then holds the exceptions. These exceptions, when not running, are in a pending state. Under the pending state, the exception waits on the processor to be freed in order to operate. This pending state is achieved when a peripheral sends an interrupt request. Once the peripheral's request is now sent to the processor, it changes to active state. The processor then runs the code for the particular exception and once finishes, exits the code and saves the processor state. The interrupt of that particular peripheral then becomes inactive. The last state is the active and pending state.

These operations pair well with low-power modes as they allow the MCU to power down the processor and even memory when not needed, and still have a means to achieve the results through an interrupt to power back the processor for specific tasks and revert back to power down when unused.

3.3.1.2 Atmel Atmega 2560

The Atmel Atmega 2560 is an 8-bit RISC microcontroller and is the other contender for the MCU of this project. The main appeal is it's high performance, low power and the powerful instruction set which was described to give 1 MIPS (Million Instructions Per Second) per MHz.

3.3.1.2.1 Memory

The Atmega offers 256K bytes of both Flash memory and SRAM. The Flash memory is divided into a boot program and an application program section. The microcontroller also holds an EEPROM data memory that can hold up to 4Kbytes of data. EEPROM, which stands for Electrically Erasable Programmable Read-Only Memory, is programmable for a read-only and thus must hold data that should only change when necessary. Due to this, there is a writing procedure to EEPROM writes so that there is no mistaken deletion of data from a ROM that can be programmable. A noteworthy warning is that when Vcc is

low, the EEPROM can be corrupted, therefore writing to it must require the sufficient voltage requirements in the specifications.

3.3.1.2.2 Specification:

The Atmel Atmega 2560 has a maximum operating temperature of -55 to 125℃℉, max operating voltage of 6V, and maximum current for Vcc and gnd pins being 200mA, and per I/O pin the current is 40mA. The maximum clock frequency is 16MHz.

3.3.1.2.3 Low-Power Modes

The low-power modes for the Atmega2560 is different to the MSP432P401R, however the same concept of powering off the oscillator, processor, and other components based on the low-power mode stands. In this microcontroller, the low-power modes, or sleep modes, are idle, ADCNRM, power-down, power-save, standby and extended standby.

Idle mode: is the sleep mode that turns off the CPU and Flash memory, but retains the other clocks and peripherals such as SPI, USART, ADC, WDT, Timers, etc.

ADCNRM: ADC Noise Reduction Mode, is a mode that is used to reduce the noise of the ADC by turning off the CPU, Flash, and I/O and only allows interrupts, ADC, WDT, timers, and 2-wire serial interface to stay on. This mode only allows external reset, WDT interrupt, Brown-out interrupt, serial interface, timer, and an EEPROM can wake up the MCU.

Power-down mode: This turns off all internal peripherals except the WDT reset, serial interface, external reset/interrupt, Brown-out reset can cause the MCU to wake up.

Power-save mode: Power-down mode except allows the timer 2 to run if enabled

Standby mode: Power-down mode except external oscillator runs, and the MCU wakes up in six clock cycles.

Extended Standby mode: Same as standby mode, except MCU wakes up in twelve clock cycles.

3.3.1.5 Atmel Atmega 328

The Atmega 328 is another 8-bit RISC architecture that is being looked at as the custom PCB to be utilized to control the motors for the turret movement and firing, as well as controlling the range finder and low power modes for operating optimally. As for the low power modes, the atmega328 utilizes the same sleep modes as the atmega2560 and therefore the contention between these two will be determined by peripherals offered, the amount of those peripherals, and the voltage supplied to the microcontroller as well as

what voltage it will supply to the peripherals as well as the price and the need for the amount of peripherals are needed.

3.3.1.5.1 Memory

The Atmega 328 has a 32K byte of flash memory with a 2K byte SRAM, divided up into 32 general purpose registers and 64 I/O registers. The 32K byte flash memory is enough to program our intended turret project since the program only utilizes the low power modes, control of the motors and range finder, and the coordination of the computer vision, which is programmed and utilized by the other microcontroller. The SRAM is also more than enough to operate an autonomous and/or wireless controlled turret.

3.3.1.5.2 Specifications

The Atmega 328 has an operating temperature of maximum 125 degree Celsius and a minimum of -55 degree Celsius. The maximum operating voltage is 6V, but most likely will be supplied with a 5V. The current of I/O pins has a typical value of 40mA for the I/O pins, while having a 200mA for the VCC and GND pins.

3.3.1.6 Comparison

The three microcontrollers being compared for the custom main pcb design are the Atmega 2560, 328, and the MSP432P401R. A comparison table is shown below that describes the many features each of them have and what peripherals and how many of those peripherals the microcontrollers have.

Microcontrollers	MSP432P401R	Atmega2560	Atmega328
Operating Voltage Range	1.62-3.7V	1.8-5.5V	2.7-5.5V
PWM Channels	20	12	6
UART/USART	4	4	1
SPI	8	1	1
Flash Memory	256KB	256KB	32KB
SRAM	64KB	8KB	2KB
ROM/EEPROM	32KB(ROM)	4KB(EEPROM)	1KB(EEPROM)
Maximum Clock Speed	48MHz	16MHz	16MHz

Table 2: Microcontroller Comparison Table

The meaningful comparisons of the microcontrollers are on display in figure 8, the comparison table. We look into the most important features required to have the operation of the turret be a working project. The first metric is the voltage operating range. The voltages for the atmega had slight minimum range difference however they function well at 5.5V maximum, while the MSP432 has maximum of 3.7V. The advantage of the atmega will be that usually motors are rated for 5V, especially the ones we are currently looking at which will be discussed more in the motors section. This means that the atmega is the better choice if we wanted to directly connect the microcontroller to the motors, however, the MSP432 will still be able to utilize the motor if we step up the voltage. The other metric is the PWM channels and the quantity each microcontroller has. The MSP432 having 20 channels and the atmega 2560 holding 12 is impressive, however, the winner in this case would be the atmega328 because of the demand of PWM channels being met with the 6 and any more than that is an extra that will not be used in this project. The SPI metric shows that the MSP432 holds 8 unique SPI modules while the Atmega's only hold one. The advantage of having 8 instead of 1 is that the more the microcontroller needs to communicate, the better the larger amount of SPI's due to having parallelism and more quantities in the amount of devices to communicate. However, having one SPI is not necessarily a detriment, especially if there is no need for more than one of these communication modules. There is also the fact that there are other serial communication modules that can be used to either have the parallelism or just to have another module of communication for another device. The flash memory has only two differences with the MSP432 and Atmega2560 holding the 256K bytes flash memory, which is able to hold the code of a project that is large, in fact far larger than this project may even turn out to be. The Atmega328 holds only 32K Byte of flash memory, but this is only lower in comparison. In actuality, 32K bytes is more than enough to hold the programming of our device and the peripheral code plus more. The same principle holds true for the SRAM, where the memory size of the Atmega 328 holds more than enough data for the temporary storage of data and values for as long as the turret is operating.

One large difference between the Atmel and the TI microcontrollers are the ROM's are different types, with the MSP432 holding a traditional ROM while the Atmel microcontrollers hold EEPROM. The EEPROM used in the Atmel microcontrollers act as another memory for the data, which is useful because the SRAM will not hold the values when it is not being powered, but the EEPROM will. The ROM, used in the MSP432 holds the libraries of drivers, which is useful as there is an already pre established functionality within the microcontroller but the ROM is not editable and as such will only remain with those drivers. The size is also important but only for the EEPROM because the ROM is uneditable and as such not going to be used. The EEPROM will come with either 4K bytes for the 2560 or 1K bytes for the 328. Generally, more is better in terms of safety as with more memory space, there is room for any additions but the 1K byte will be enough if we require very little data to be stored for later use with the need to store that data even when the turret shuts down.

The final metric is the clock speed. The maximum clock speed for the MSP432 is 48MHz, which is incredibly fast in comparison to the two Atmel microcontrollers, which hold 16MHz. The faster the clock speed, the faster the reaction and for the turret, a 48MHz clock speed will be a huge boon when it comes to responsiveness to facial recognition and the motor movement to the target. 16MHz can definitely do the job but with less speed. In this case, the MSP432 is the clear winner in this regard, especially when the voltage required to power the 48MHz is a 3.3V, lower than the Atmel microcontrollers which both need a 5V power supply to effectively power on the 16MHz clock.

3.3.2 Computer Vision

Over the years technology has advanced quite a bit. One of these advancements has been on facial recognition to the point where there are two possible ways of doing it. One of these ways is software-based where the computer with a camera plugged in uses software to identify the different faces. The other possibility is a hardware-based approach where the device is already capable of facial recognition without much tweaking to the code.

3.3.2.1 Software

To achieve the best results for image processing then software would be the best approach, whereas the hardware is not able to recognize the difference between an object unless the software that is loaded into the hardware is able to do so. When considering the software approach there is no special hardware required just a regular camera and a computer. Depending on the hardware used the possible computations can increase and the possible software programs that can be used are improved if the hardware is strong enough. Now when considering the software there are a few popular algorithms that are used to detect faces; some of these include the Haar cascades, HOG + linear SVM algorithm, as well as the CNN face detector.

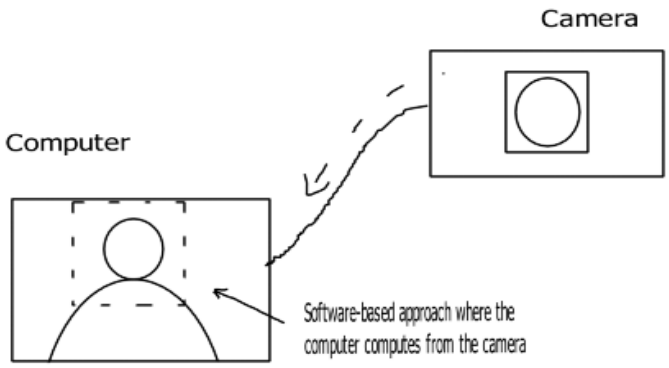


Figure 8: Software-based approach

Beginning with the Haar cascade which is usually included with OpenCV and is known to most. It is a lightweight code that allows for fast computation in systems and does not require an abundant number of resources, which allows us to use smaller computational devices such as a Raspberry pi. Once loaded the code needs to be set up to cycle through the frames obtained from the camera to distinguish whether there is a face present or not. This is done through a pre-trained model that is included in the library allowing for a fast and easy setup. The algorithm uses edges and lines rather than the pixels because it allows for faster computation. The reason for this is because there are many pixels to cycle through and determine its correlation in the frame whereas focusing on the edges and lines allows for a wider section to be used rather than going one by one. Not only is it faster but it also allows multiple faces to be detected in a frame. Although this is great in terms of detection speed it does succumb to false positives. This can either be faces not being detected that are there or a fake face that is not an actual face.

The second option that allows facial recognition would be the Histogram of Oriented Gradients (HOG) + linear support vector machine (SVM. The idea behind this is that the algorithm will divide the frame from the camera in this case to smaller connected sections. Each section will be read and a histogram is created for each which is then combined into a single main histogram which will have the unique data for each face. This technique is more accurate than the Haar cascade and has better documentation giving us an easier time to tune but at the same time, having few parameters which mean less problems in the code. This is favorable when looking at the ease of use factor for the automation side of the turret.

The last option when looking at the software that we have decided to use is the Convolutional Neural Network (CNN) face detector. The CNN is available through the dlib library and has a good ratio between the speed along with the accuracy on facial recognition. Unlike the Haar cascade and the HOG + Linear SVM the CNN to an extent is better able to recognize faces at peculiar angles. The best result can be achieved with the addition of a GPU which can help increase the speed of the face detection Since we are not using a GPU due to economic and availability constraints, we have to rely on the cpu which will not offer the best performance but should be efficient.

By keeping the design factor in mind, the software-based approach may be a little limited in what can be accomplished. As a laptop or GPU may not be viable in the design as they add additional weight or are not available to purchase freely. Due to these limitations, we have decided not to use the CNN face detector as the best result would be achieved with the use of a GPU. Now considering the Haar cascade and the HOG + linear SVM algorithm they both are a good option as they do not require special hardware or tons of space to get optimal results. As we have decided to include a raspberry pi the two options are perfect but when considering that we are also using a distance measuring technology as well the Haar cascade with its faster computation will allow us to track people better than a bit slower HOG + linear SVM algorithm.

3.3.2.1 Hardware

As the software-based approach utilizes two components which are the camera and the computer for computation. Without one or the other this approach is seemingly useless as the camera is not able to interpret what it is able to see and the computer by itself is not able to have any sort of vision without having a camera mounted on it. The camera is connected to the computer and the software running on there will decipher video feed. This can be accomplished in a few different ways, one being a live decryption where it must go frame by frame as it comes in to interpret faces. The second approach is a bit slower, where the camera takes the instance from the camera and outputs the results once it has been deciphered. Although the second option would be ideal to get the best results it is not something that we are going to be using as it slows down the facial recognition process and gives problems in terms of tracking people as the person will have moved by the time that the face was recognized. The software however does allow the use of a computer and its myriad of software languages available to use facial recognition such as C, Open CV, and Python.

Although computers have come a long way in battery life, weight, and size it may still be an issue to include one in our design. The reason for this is the additional weight that will need to be incorporated into the design, potentially limiting the portability, as well as the battery life being impacted from the computational power of the computer. This would not only mean that we would need to manage battery expenditure but also would require us to invest more in having a device powerful enough to have a fully operational device with a large enough battery to keep the same length of operation. To limit the weight, the battery of the computer would be used, making the computer that would have lasted a few hours to last much less than that. Along with the additional weight and the operational time there is the cost value due to the best performance being achieved by having a higher end multicore device or a device with a comparable GPU.

While looking at the software-based approach there is another option. This option would be the raspberry pi, which is a small, sized computer that comes without a battery allowing us to use one of our own. Taking into consideration the power of this consumption being approximately 64 watts with all four cores being stressed (not including the additional peripherals on the pi) this would be a great option as it is miniature in size and offers a substantial amount of computational power, however it still lags behind a traditional computer. Unfortunately, like the computer it is not capable of facial recognition out of the box but can be programmed to use an external camera to identify and track faces. Not only would this allow us to make a more portable version from the precious proposed computer, but it would also allow the use of a battery due to its lower power consumption. This would potentially be capable of using more powerful software to create an automated device that can target people. Along with the option to increase the battery size this could, with proper power management power the device and run for as long as the computer would have if not more.

On the other hand, the hardware-based approach gives us the ability to create a more portable device compared to the software-based solution. As the computational capability is built into the camera itself therefore you would only need a central processing unit to interpret the signals from the camera, not only saving space but also power as well. The options for this method include the OpenMV M7/H7, Pixy 2 , and the Jevois. They are a

portable version of the computer and camera combination which allows use of a larger battery from the saved space as well as keeping a smaller form factor in mind. At the same time this would give us a better operational length of time as the devices are lower power as well. The cost is relatively similar to one another, and they provide a couple of different features between each other.

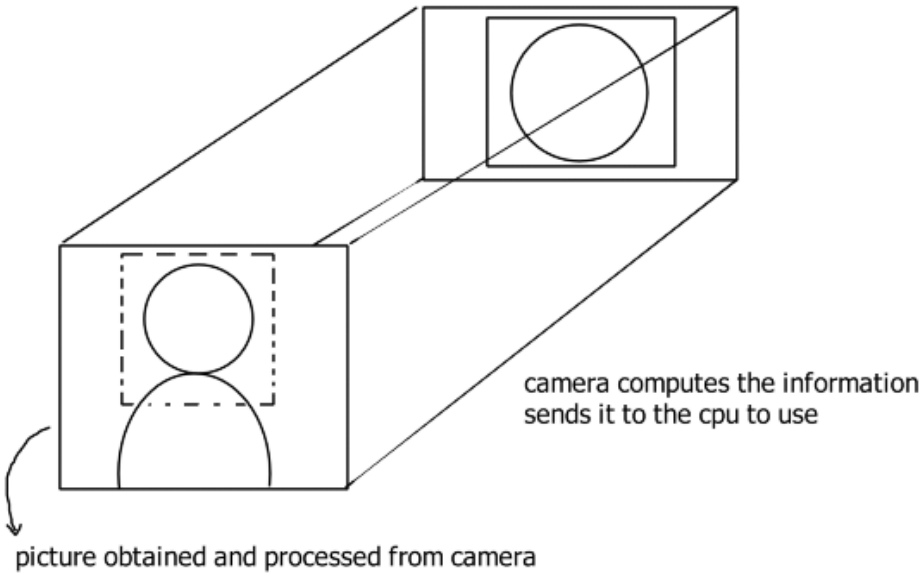


Figure 9: Hardware-based approach

Starting with the OpenMV M7/H7 which has cross platform support. The Open MV utilizes a version of python called micropython. Like the raspberry pi this option draws a small amount of power as well and only uses about 200 mA allowing us to use a battery pack. This gives us the possibility to emphasize portability which still gives us the resources to build an automated device. Since we are using micropython the library is almost preset for the device, this means that although the device does have capabilities to have the code modified and improved the actual tweaking of the code is limited to an extent. The device lacking in software modifications is not a huge disadvantage as it does come with many I/O ports. The Open MV cam has a SPI bus, I2C bus, Async serial bus (RX /TX), ADC, DAC. Playing with the idea of ease of use this option would be great in setting up the automated turret due to its programming limitations, and with the extra I/O ports it makes it easier to connect the extra devices to expand and develop the device as well.

The second option is the Pixy 2 camera which has access to a viewing app only that allows you to teach the pixy cam up to seven different colors. Although this may work in terms of recognizing and tracking colors it will be hard to use it to track objects especially people in real time due to color differences. For that to be possible we would have to create a new color recognition every time a new person comes into view for the best results. An option assuming there is a set dress code where people wear a specific color to determine an enemy from a friendly. However, in terms of facial recognition the

pixy 2 is not able to accomplish this unless running a haar cascade classifier which can be used for an object detection. The haar cascade can identify objects from a small distance with the pixy cam, but as the range increases the camera's accuracy will decrease. Taking into consideration the power consumption of this device being 140 mA at 5V this is also a favorable option even with its limited ability to detect faces and primarily detecting colors only.

The third option that we have decided to consider is the Jevois. This is like the OpenMV and has cross platform support. Unlike the OpenMV, Jevois does not use micropython but python itself. This allows us to fully utilize the camera and see if there are other approaches in the software to give the best results. Along with the option for facial recognition this would be a great device to consider given that it uses a total power of 800mA totaling around 4 watts. There is also the option to reduce the CPU usage by modifying the code to further reduce the power usage as well. Along with the option for facial recognition this would be a great device to consider because of the modular software design giving us full control over the camera. Although this would be great in terms of software the hardware part of the camera is somewhat lacking. Unlike the OpenMV the camera only has one usb port.

Devices	Power usage	Software / hardware	Programming language	I/O ports
OpenMV M7/H7	200 mA (max)	hardware	micropython	SPI bus, I2C bus, Async serial bus (RX /TX), ADC, DAC
Pixycam 2	140mA (5V)	hardware	Not programmable	I/O connectors and usb
Jevois	800mA (4w total)	hardware	python	usb
Raspberry pi	64 W (w/all cores)	software	Almost any	Ethernet, usb, hdmi, usb c, I/O connectors

Table 3: Device Comparison

Overall, the best approach would have been using the computer and camera combination to achieve the best results. Due to budget and design restrictions, we have decided to narrow down the usable devices to raspberry pi, Jevois,Pixycam 2, and lastly the OpenMV M7/H7. Although the Jevois is great in terms of software, due to its limitation in hardware I/o the best options are further narrowed down to the Open MV, Pixycam 2, and the raspberry pi. If we take into consideration the software limitations of the

OpenMV even with it's great hardware support and extensions, the raspberry pi would be the overall winner. However, we have decided to use a combination of the raspberry pi and pixycam 2. This would allow us to use the color feature of the pixy as well as the software capabilities of the raspberry pi for the best facial recognition results.

3.3.4 Batteries

Trying to choose a specific battery technology and form factor are almost entirely reliant on what electronic components are actually going to be used in the project. The amount of power delivered by the battery must be sufficient enough to adequately power whatever MCU we decide to use, as well as the numerous motors we needed to allow our turret to maneuver correctly. As discussed in *section 3.2.2,* there are several parameters that determine the effectiveness and the efficiency of the application of each of the battery technologies in our project. These key factors are; the available form factors of the batteries, the power densities of the different battery chemical makeups, the energy density of the batteries, and the rechargeability of each of the state battery technologies.

Symbol Key
Best in Category ↑↑
Good ↑
Poor ↓
Neutral –

Battery Technology	Energy Density	Power Density	Rechargeability	Footprint
Lithium Phosphate	↑	↑	↑↑	↑↑
Nickle Metal Hydride	–	↑	–	↑
Lithium Ion	↑↑	↑↑	↑	↑↑
Lead-Acid	↓	–	↓	↓

Table 4: Battery Technology Comparison

The easiest criteria to start with is the energy density and power density of each battery type. Of the four battery technologies we have preliminarily chosen; lead-acid, Li-ion, LiFePo, and NiMH, all would be sufficiently capable of powering our turret with the help of voltage regulators and potentially step-up or step-down converters. While the energy densities of the Li-ion and LiFePo are significantly higher than that of the lead-acid and the NiMH batteries, all four battery technologies would be able to provide the relatively low voltages and currents to adequately power our turret.

The next criteria is the rechargeability of each of the given battery technologies. All four battery types are capable of being charged and discharged for numerous cycles, but lead-acid is the worst performer in this category by some margin. Lead-acid batteries only have about a quarter of the charge/discharge lifespan of the next worst performer in the rechargeability category, NiMH. On top of having fewer charge cycles in its lifespan, lead-acid batteries also take significantly longer to charge. While lead-acid batteries do not require the charge controllers that Li-ion and LiFePo batteries need to prevent catastrophic overcharging, they must be trickle-charged extremely slowly over the course of 8-10 hours. Whereas Li-ion and LiFePo battery packs can be charged in about a quarter of the time.

The final criteria is the available form factors that each battery type is commonly available in. Because many of our design specifications revolve around our sentry turret being as small, light, and portable as possible; a battery that is available in smaller form factors would align with our design specifications to a greater extent. NiMH, Li-ion, and LiFePo batteries are all commonly available in small form factors that would allow us to conform to our engineering and design constraints.

From the criteria listed above, the lead-acid battery is the easiest to eliminate from the pool of battery options as it performs the most poorly in every category we specified. The NiMH batteries are the next worst option as they do not perform as well as the Li-ion or the LiFePo batteries and are less commonly found in the form factors we desire.

Because of our project's power requirements and desired application, we have decided that LiPo, or Li-ion batteries would be best suited for our project. There are several different form factors that both LiPo and Li-ion batteries can be bought in.

Cylindrical Cell Batteries - Cylindrical cell batteries are the most commonly available battery for civilian use. The standard size AA, AAA, etc. batteries that are used in millions of devices around the world are of the cylindrical cell variety. The cylindrical cell is generally produced with optimal automated processes and techniques, boosting consistency and decreasing the cost per unit, thanks to its excellent mechanical stability and design. A cylindrical cell's electrodes are tightly wrapped and enclosed in a metal shell. This reduces the risk of electrode material breaking due to mechanical vibrations, as well as heat cycling from charging and discharging and mechanical expansion of the current conductors within. To improve the voltage and capacity of the battery pack, several cells are connected in series and parallel. The impact of a single defective cell on the entire pack is minimal. Cylindrical cell batteries typically only have an output of around a volt, and are typically wired in series to be used in higher voltage applications. When the batteries are wired in series, their voltages become additive, so the more cells that can be wired in series, the higher the output voltage will be. The most commonly available chemical compositions for cylindrical cell batteries are Li-ion, NiCd, NiMH, and LiFePo. Cylindrical cell batteries are typically very structurally robust, and have good tolerances to heat and puncturability.

Pouch Batteries - Pouch format batteries function similarly to cylindrical cell batteries but instead of their cathode and anodes being wrapped into a cylinder, they are laid flat with non-conductive insulation material. Because of their flatter profile, pouch batteries can be more easily implemented in electronics where a small profile is desired. Pouch batteries can also typically be bought at much higher voltages ratings and capacities than cylindrical cell batteries due to the fact that more sheets of cathodes and anodes can be easily added to increase the output voltage of the battery. The most common implementation of pouch style batteries is in portable electronic devices such as cellphones and laptops. The most common downside to pouch style batteries is that they are extremely susceptible to punctures, which can lead to catastrophic failure. Due to pouch style batteries being found in portable electronics where size and weight are kept to a minimum, the outer sleeve of pouch style batteries is typically made of thin sheets of aluminium or a soft plastic composite. This makes pouch style batteries much weaker than their cylindrical cell counterparts. Due to the cell structure and the chemical composition of most pouch style batteries, a severe puncture that results in the internal components of the battery being exposed to air can cause the battery to violently combust or explode.

Although the specific voltage and current requirements for our project are entirely dependent on what type of MCU and motors we decide to use, our project's requirements can be fulfilled with minimal electronic equipment and will not have an especially high power requirement. Of the MCUs we are choosing between for our project, all of them only require 3.3v-5v inputs and have a rated current draw of fractions of an amp, typically in the hundreds of milliamps range. Some MCUs also have an unregulated voltage input that will allow a much wider range of voltages to be input to the MCU and the MCU itself will regulate the input voltage. This allows us to be more flexible with the rated voltage for whatever battery we decide to use. As for motors, some types of motors require a more significant level of voltage to be able to drive them. Because motors generate their torque based on the amount of power they use, if a motor is rated for 9v but only receives 5v as it’s input, the motor will not produce it’s rated torque and will operate much slower. Because our goal is to have our project acquire it’s target quickly and then be able to follow the target, we have to ensure that our motors are getting as close to their rated power as possible so the motors can operate at their designed speeds.

We have narrowed down our battery options to the following three specific batteries.

Anker Powercore 10000mAh - The Anker powercore is a popular portable cellphone battery bank. It is made up of three Li-ion cylindrical cells that are wired in series, and they are all encased in a sturdy plastic shroud. Along with the individual cells, there is an on-board voltage regulator that ensures that whatever device is plugged into the battery bank will not draw more current or voltage than the battery pack can handle. The Anker Powercore has a capacity of 10Ah and a rated voltage output of 5v at 2.4A. The capacity of the battery bank is more than enough to power an MCU of our choosing for several hours, however, whatever motors we decide to use will significantly increase the rate of power consumption. Because of the load that the main MCU and the motors will draw, the Anker Powercore 10000mAh will most likely not be able to handle the combined

load. It will, most likely, be able to power the secondary MCU that is just in charge of running the computer vision. The combined power draw of the MCU and the computer vision module is expected to be less than 1A, well within the 2.4A sustained output of the power bank.

Figure 10: Anker PowerCore 10000mAh Cell structure

Venom 7.4v 4000mAh - The Venom 7.4v 2000mAh battery is a high output battery designed for use in high performance remote control cars and planes. Due to the high speeds that these RC vehicles travel at, the amount of power needed to drive the motors in the vehicle is substantial. The Venom battery is a LiPo battery consisting of two pouch style batteries encased in a single shroud and wired together with a proprietary RC output connector. The Venom battery is rated at a continuous output of 100 amps with a burst output of 150A. It's rated minimum output voltage is 6v with a peak output voltage of 8.4v. While 100 amps may several times what will be required of our MCU and motors combined, it is a cheap and effective solution to the expected power draw of our project. Because the Venom battery comes with a proprietary connector, we will have to include a matching connector on our custom circuit board in order to power our project effectively.

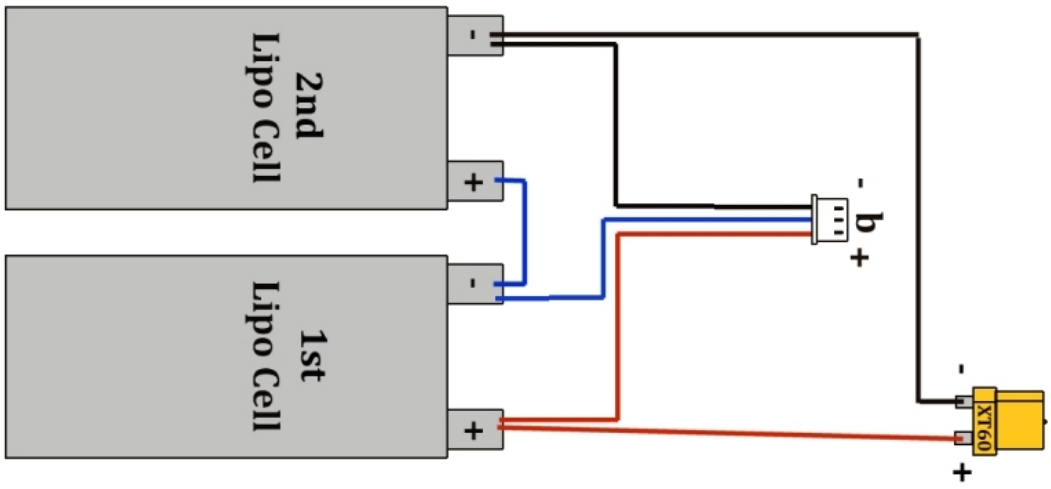


Figure 11: Two Cell LiPo wiring diagram

Adafruit 3.7v 6.27Ah - The Adafruit 3.7v battery pack is a popular solution to portably power low power MCUs such as the Raspberry pi, or the Arduino Nano. It consists of three Li-ion cells wired in series and encased in a soft plastic casing. While similar in construction to the Anker Powercore battery pack, there are no hardwired regulation circuits or other safety measures. The peak draw current of the Adafruit battery pack is 3.3A, but sustained loads that high are not recommended. The recommended operational load is under 1.3A. Due to the power requirements of the main MCU and the motors that will be used to pan and tilt our turret, this battery is not a suitable option to power our project.

3.3.5 Motors

The type of motor we are using in this project is a servo motors due to the need of positioning and high torque with higher speeds. This puts this type of motor ahead of the others, although more expensive, in regards to this project's goals. Some of the key characteristics that need to be looked for when searching for the servo motor is the stall torque and the torque. The stall torque is the torque that quantifies the force in which the motor maintains zero speed. There is also the stall current which is the current in which the motor uses when the motor is using stall torque. The motor needs to be able to support and move the weight of the firing mechanism in a dual axis motion, as well as be the firing mechanism for the nerf gun. The motor for the trigger of the firing mechanism does not need to have high torque or stall torque due to the trigger having no significant weight on these motors. However, due to the positioning, it will count as the weight the servos of the dual axis chassis board will have to support for their motion. Most motors have three pin connectors or more. The motors are split into two categories, the trigger motor and the motion motor.

3.3.5.1 Trigger Motor

The trigger motor will be attached to the nerf turret's trigger mechanism in order to turn the trigger to fire and return the trigger to a non-firing position. This motor will be attached to the trigger mechanism of the nerf gun in order to press the trigger as soon as the MCU powers the motor with a PWM to rotate the trigger to the firing state and then return back to the non firing state as soon as the nerf gun finishes firing.

Tower-Pro MG90D Servo Motor:

The MG90D micro servo motor has a 90 degree rotation, with 45 degree in each direction from the origin point. This is enough for a trigger motor since the degree rotation is only needed for the pushing of the trigger mechanism on the nerf gun, which at most may need 30-40 degrees in one direction. The size of the micro servo is 22.8x12.2x28.5 in mm, which will be important to know for the creation of the chassis board and the topology of the turret being built. That will be discussed more in the prototypes as the chassis board will have to be configured in such a way so that it fits all the components which include this motor. The speed it rotates is 0.1 seconds for a 60 degree rotation at 4.8V, and 0.08 seconds for a 60 degree rotation at 6V. The voltage will most likely be at 5V, which means we were required to determine what speed it will rotate. However, even the slowest speed is more than enough for a trigger. The stall torque is 2.1kg-cm at 4.8V and 2.4kg-cm at 6V, which is also more than enough due to the trigger being less than 2lbs. The stall current of this servo has not been specified and will have to be tested. This motor is RoHS compliant.

Tower-Pro MG92B Servo Motor:

The MG92B micro servo motor has a 180 degree rotation, divided into two 90 degrees in both directions. This micro servo has a stall torque of 3.1kg-cm for 4.8V and a stall torque of 3.5kg-cm at 6V. The servo motors dimensions are 22.8x12x31 mm in size. The speed of the motor is 0.13 second per 60 degree rotation at 4.8V, while at 6V the operating speed is 0.08 seconds per 60 degrees. The requirement shown for this motor is that the signal requires 5V, which will not be a problem due to having the means to produce 5V power either directly through the microcontroller, like the Atmega 2560 or 328. Otherwise, if using the MSP432P401R, we needed to use a step up converter to boost the signal to 5V. The weight of the motor is 14 grams, which will not add much to the overall weight of the turret and as such the rotation of the chassis will not be hindered. This motor is a lightweight servo with enough torque to move the trigger mechanism of the nerf gun quite easily. The stall current was not found, however, under a forum on adafruit there was an estimate of 850mA. Finally, the motor is RoHS compliant.

SG92R Servo Motor:

The SG92R micro servo motor has a 180 degree rotational movement, with a 90 degree rotational movement on both sides just as the previous motor. The dimensions of this

motor are 23x12.2x27mm, and has a weight of 9g. The motor has a stall torque of 2.5kg-cm at 4.8V, and does not have any specifications beyond that voltage, thus we assumed that the operating voltage at stall torque is only at 4.8V. This motor also does not state the stall current, however, this motor and previous motors that do not state the stall current should be in the range of 1-1.5 A, and thus there is no need to worry heavily on the supply of current needed as the power supply will have plenty to power these motors. Finally, this motor is also RoHS compliant.

Comparison:

Servo	Stall Torque	Price	Weight	Dimensions
MG90D	2.1kg-cm at 4.8V 2.4kg-cm at 6V	$9.95	13.4 grams	22.8x12.2x28.5 mm
MG92B	3.1kg-cm at 4.8V 3.5kg-cm at 6V	$11.95	14 grams	22.8x12x31 mm
SG92R	2.5kg-cm at 4.8V	$5.95	9 grams	23x12.2x27 mm

Table 5: Motor Comparison

The SG92R is the motor with the cheapest cost, the least heaviest, and has a similar form factor as the MG90D and slightly less than the MG92B. For cost effectiveness and a generally smaller form factor, the SG92R is the better choice. In terms of the torque, the MG92B is the better choice, but is generally heavier and has a slightly larger form factor, however the degree of difference between these sizes for all the motors or small enough to be insignificant except for the most extraneous of circumstances, though with good design this will not occur. The MG92B is the middle choice, in which it has a similar torque as the SG92R, while also being a bit more expensive, but less expensive than the MG92B, and the weight is also similar to it. In terms of the application of motion that will pull or release the motor, the necessity of torque will depend heavily on the application of how it will pull and release the motor. For greater distances, the larger the torque required in order to compensate. With the specifications of these motors, we generally do not have anything to worry about due to the trigger not needing much weight.

3.3.5.2 Motion Motors

The motion motor is the motor to be used to move the chassis board, and thus the majority of the nerf turret. The motor is required to have enough torque and speed to supply the turret with the ability to move to the target location specified by the computer

vision program or bluetooth controller in a timely manner rather than be too slow and consume the time for an effective turret. One of the motion motors will be attached below the turret firing mechanism or the chassis board to move the turret in a horizontal fashion. The other motor will be used for the vertical motion and will be determined based on the design of the chassis board and turret.

MG995R Servo Motor:

The MG995R servo motor is a high torque metal gear servo motor that has a total of 120 degree rotation, with 60 degrees from the origin to either direction of rotation under the 1.5-2.5 ms pulse, however with increased pulses the rotational degrees increase to a 170 degree range. However, the specification of this degree is stated to depend on the servo. The average speed of the motor was stated in the specification to be 60 degrees in 0.2 seconds for a voltage of 4.8V and 0.16 seconds per 60 degrees for a voltage of 6V. The stall torque was given as 9.4kg-cm at 4.8V and 11kg-cm at 6V. The motor dimensions were 40.7x19.7x42.9 in mm. The current at idle is 10mA while the stall current is 1400mA. This motor was also specified to be RoHS compliant, a standard that will be discussed in the standards and constraints section. This servo has all the required specifications to be able to create the horizontal motion of the turret when looked at being applied directly. It is cost effective and provides enough torque to move the nerf gun, which is estimated to be around 2lbs, which is 0.9kg. This value shows that it will have no problem for as long as the components are attached correctly.

MG996R Servo Motor:

The MG996R is the successor and upgrade of the MG995R that is stated to have better accuracy with an improved control system through a redesign of its PCB design. It is also stated to have better internal gearing. The specifications for the MG996R is very similar since it is a direct upgrade to the MG995R. They have the same stall torque for the same voltage values. The operating speed for the 4.8V is 0.19 second/60 degree and has a 0.15 second/60 degree for the 6V, a 0.1 second faster. Operating voltages are from 4.8V-6.6V.

DSSERVO DS3218MG Servo Motor:

The DS3218MG servo motor is a high torque servo motor that offers a stall torque of 19kg-cm at 5V and a 21.5kg-cm torque at 6.8V. This amount of torque is a lot higher than what may be required but it is looked at for the chance that the turret's build will require a more torque intensive servo motor than originally thought. The speed of the motor is stated to rotate at 0.16 seconds per 60 degrees at 5V and 0.14 seconds per 60 degrees at 6.8V. The operating voltage range is from 4.8V to 6.8V and has a weight of 60 grams. The size of the motor is 40x20x40.5 mm and is RoHS certified.

Motor Comparison:

Servo Motors	MG995R	MG996R	DS3218MG
Operating Voltage Range	4.8-6.6V	4.8-6.6V	4.8-6.8V
Speed	0.2-0.16s/60deg	0.17-0.13s/60deg	0.16-0.14s/60deg
Stall Torque	9.4-11kg-cm	9.4-11kg-cm	19-21.5kg-cm
Stall Current	1200mA	1400mA	Not found
Size	40.7x19.7x42.9mm	40.7x19.7x42.9mm	40x20x40.5mm
Price	$19.95	$20.99	$15.99

Table 6: Motion Servo Motor Comparison

The three motors being compared are shown in the figure above. As previously stated, the MG996R is an upgraded version of the MG995R and as such shares many of the metrics. The comparison is thus whether the value is greater and for the most part it is. The price for the MG996R is slightly higher due to being the upgrade, while the DS3218MG cost less. The operating voltages are all close to one another and will work with the autonomous turrets' 5V. The speed shows that the MG996R is faster than the MG995R in both voltage values, but only wins out in the higher voltage range to the DS3218MG. The stall torque for the MG996R and 995R are both equal to one another but the DS3218MG has a greater stall torque with a nearly 10 kg-cm gain. While the torque difference is quite big, the need for all that torque is not necessary, however, with the metric of price showing that the value of a DS3218MG cost less than the MG995/6R gives the edge to the DS3218MG. One metric the DS3218MG does not state is the stall current, and there was no datasheet found to look through the specifications, all data is given based on the vendor description of the specification. However, the MG995R holds 0.2A less current than the MG996R, and as such the MG995R will be the motor with the assumed least amount of current required to be powered. The sizes are all similar which means that there is no clear winner in the compactability department.

3.3.6 Distance Measuring Sensors

There are a few ways to measure the distance when it comes to the automated turret. Keeping in mind the design factor as we need to utilize something that is low power, portable, and easily accessible. Given these parameters we have a couple of options that include the ultrasonic sensor, IR distance sensor, laser distance sensor (LIDAR), and the LED Time-of-Flight sensors. With these we can calculate the distance between the target and the turret and possibly set a firing requirement if the target is within a certain

distance. Not only that but we are also able to add a couple of features like some sort of warning message, whether that is a warning shot or even a vocalized message.

3.3.6.1 Distance Measuring Techniques

To begin with, the ultrasonic sensor due to its pricing and availability is probably the best and easiest option. By using high frequency waves it can calculate the distance between the object and its location on the turret. This is great because it is a low power device as well as a portable form factor which would keep us well within the weight and size of the automated device. The power management may still be an issue as we could have to determine whether to keep the device always powered on while the turret is active or whether to have it shut down until a person is detected from the camera, and there is also a possibility of waiting till a person is detected before switchin the sensor on and instead of continuous bursts we use short bursts every so often. Although the device specifications are great for the design it has some drawbacks. These include the accuracy as the distance from the sensor increases as well as environmental changes such as if an object happens to block it for a second. The other downside is the distance factor which is around 2cm - 500cm, this distance can vary depending on the device purchased but to achieve an even longer distance it would require us to spend more.

The second option is the infrared (IR) distance sensor and there are two options for it, one being the active and the other being passive. The active sensor has two parts where there is an LED and a receiver. The LED is used to shine the light at the object which is reflected and picked up by the receiver. The other option for an IR distance sensor is the passive one where the sensor only has a receiver to detect the presence of infrared radiation. With these two options the passive one will not work as the turret needs to perceive the distance from the target so if the target does not emit infrared radiation, then it will not be able to determine the distance. On the other hand, we should be able to set up a perimeter around the turret to let it interpret whether the target has passed that preset distance. This would however cause an issue in the portable design as well as the issue of ease of use in terms of how long it takes to set up and go. Therefore, the best option is this case would be the active IR distance sensor. The advantages of this sensor would be that it can be used at any time of day, is lightweight/small, and can measure the distance of objects with complex surface structures which is a good option for tracking people. The downsides to this technique would be in the measuring of the distance itself which is limited to about 100cm – 500cm, and like the ultrasonic sensor can be increased with an increase in budget for the device.

The third option would be the LIDAR system which is basically a laser-based sensor that measures distance. There is an option for a passive LIDAR system but as discussed with the IR distance sensor it is not the best option. The active LIDAR system laser would shoot a beam of light at the object/target which would be reflected to the receiver. The distance between the object would then be calculated by using the relationship between the speed of light and the time it took to receive the return signal. Like the IR distance sensor, the LIDAR is also a lightweight and small form factor allowing us to keep the portable design as well as the addition of a battery. The advantages of the LIDAR are that it can be used on complex structures, is fast, can be used at any time of day to detect even

small objects, and has a max operating range of around 10m. However with the targets being covered in fabric that range may be affected negatively.

The final option that we are considering is the LED Time-of-Flight distance measurer. It is like the LIDAR system that uses the pulsing lasers to get the distance. However, the main difference is how the lasers are used, for one LIDAR uses lasers to build a point cloud from single pulses in different locations to create a 3D map, whereas the Time-of-Flight uses light detection which can be done through a regular RGB camera. The advantages of this method is that it offers a good accuracy, it is capable of 3D imaging, has the ability to identify large objects, and an operating range of 4m. Unlike the LIDAR which can be affected by soft fabric this will not be negatively impacted. The downside to the Time-of-Flight would be the high cost and the Z-depth being a little off at times.

Distance Sensors	Approximate Range	3D Mapping	Complex Structure Mapping
Ultrasonic Sensor	2cm - 500cm	no	no
IR sensor	100cm - 500cm	no	yes
LIDAR	Around 10m	Yes (if using multiple)	yes
LED Time-of-Flight	Around 4m	yes	yes

Table 7: Distance Sensor Comparison

While considering the different options they all have the benefits and disadvantages. The different options being the ultrasonic sensor, IR distance sensor, laser distance sensor (LIDAR), and the LED Time-of-Flight sensor we also had to look at the budget, accessibility, as well as the performance. The IR sensor as well as the ultrasonic sensor are not viable options mainly due to their minimal range, therefore the LIDAR, and the Time-of-Flight are the top choices. If we take a further look into the two the LIDAR is a bit more on the expensive side without offering much over the Time-of-Flight. Unlike the Time-of-Flight the LIDAR offers a longer range even with the possible negative effect on distance from the use of fabric. Although the Time-of-Flight is a more attractive technique due to its limitations in distance, we have decided to go with the LIDAR system because we also have the use of a camera.

3.3.6.1 Distance Measuring Devices

Now that we have decided which form of distance measuring we are going to be utilizing, we have to look into the available devices for the ultrasonic sensor. Some of the ones that we are considering are the Slamtec RPLIDAR A1 - 360, Lidar-Lite v3, MakerFocus TFmini-s Micro Lidar Module, SmartLam TP-Solar. Each of these devices are a take on

the LIDAR sensor and they are an active version as well for all. While they all deploy the same tactic for determining distance the features and price range differ quite a bit. The overall size of the devices are fairly small meaning that will not be included in the comparison for the overall decision in what device to use.

The first device to consider would be the Lidar-Lite v3. The Lidar-lite requires a power source and a microcontroller to process the information collected, this is similar to the rest of the LIDAR devices as well. Now the communication part is handled via a 12C and a PWM connection allowing full control on how to control and establish a connection between the device and the turret. Now that we have discussed how the LIDAR will connect to the turret we can compare the specifications of the device. The range for the Lidar-Lite v3 is around 5cm to 40 meters (+/- 2.5 cm). The power consumption is around 6V max and the current consumption is 105mA (idle) and 130mA (continuous) with a peak power of 1.3 watts. The overall power usage is small enough to include and operate with the use of a battery along with the other components that make up the autonomous turret. *On the other hand we would have to monitor the power usage as a constant use from.*

The second device is the Slamtec RPLIDAR A1 - 360. The main difference between this particular LIDAR system and the rest of the options that we have chosen is the additional feature of it having a 360 degree rotation capability. The advantages of this would be that we could limit the use of the camera and save energy to extend the use of turrets operational time. In order to achieve that possibility we would have to use the LIDAR to rotate continuously to map out the surroundings and detect changes. Now the downside to this approach would be determining whether the change is caused by a person or whether there was something else that disrupted the surrounding. There is also the possibility that the height of the LIDAR may affect the results as if it is too high it may miss the person that is sitting below that scan height and if it's too low it may allow a small animal to interfere with the scan such as a dog. Continuing to the product specifications we have a reduced range from the Lidar-Lite v3 which is reduced to around .15m to 12m (+/- 2.5%). The power consumption is .5 watts with the system current being 100 mA and the system voltage being 5V.

The third option is the MakerFocus TFmini-s Micro Lidar Module. This model is compatible with the UART as well as the I2C interface. The main application for this device would be pedestrian detection and vehicle detection. This makes it a perfect use for our autonomous turret as our main targets would people using face detection. The electrical parameters for this device would be a max current of 140mA, power consumption of .7 watts, and the peak current of 200mA. The operating range is similar to the Slamtec RPLIDAR A1 - 360 from .1m to 12m (at 90% reflectivity and +/- 1% accuracy at 12m). Although the range is similar it lacks the feature of a 360 degree view, and it will solely rely on the motors of the turret to view the surrounding area.

The final option that we decided to look into was the SmartLam TP-Solar. Like the others this is a small sized and a single point detection system relying on the motors of the whole system to view the area. The interface that it uses for the communication is UART and I2C making it the perfect option to control it with the system on the autonomous

turret. The electrical operation of this device is a power consumption of .4 watts. This is most similar to the MakerFocus TFmini-s Micro Lidar Module the main difference being operational frame rate which can go up to 500Hz unlike the TFmini-s which operated around 100Hz.

Devices	Power usage	range	360 degree rotation	speed
Lidar-Lite v3	6V, 130mA	5cm - 40m	no	500Hz
Slamtec RPLIDAR A1 - 360	.5W, 100mA, system: 5V	.12m - 12m	yes	8K sampling rate 10Hz
TFmini-s	.7W, 200mA	.1m - 12m	no	100Hz
TP-Solar	.4W	.2m - 12m	no	500Hz

Table 8: LIDAR Sensor Comparison

All of these would be a great addition to the turret for keeping track of the distance from the target and allowing a fully autonomous approach on shooting. Given that the range of the Lidar-Lite is up to 40m it is possibly the best option as it gives us the choice of tracking people as they move closer at a constant rate. On the other hand the TFmini-s, RPLidar, and the SmartLam all have an operating range of around 12m give or take on the accuracy which differs from each device. With the addition of the extra feature on the RPLidar being the 360 degree scanning it is not favorable as the range is still around 12m and we are able to use the motors on the turret. Along with the range of the turret being 27m and that would be assuming that the conditions are optimal with no environmental conditions affecting the range such as wind. Therefore we have narrowed down the choices to the TFmini-s and the SmartLam. Although they are similar in performance, the price difference and the main difference being the operational frame rate being 500Hz (SmartLam) versus 100Hz (TFmini-s). In the end we have decided to go with the SmartLam.

3.3.7 Firing Mechanism

The firing mechanism is an attached nerf gun that will be attached to a dual-chassis board that will support its motion through the use of the motors. The firing mechanism will have a motor on its trigger so that when the MCU either finds the target using the automated computer vision system, or if the user using the bluetooth controller decides to fire, The MCU will power the trigger motor that will trigger the firing of the nerf gun.

Figure 12: NERF Doomlands Desolator Blaster

The Doomlands Desolator Blaster Nerf gun has a magazine of 10 darts with a firing range of 90 feet, which translates to 27.432 meters. This provides more than enough firing distance and magazine size to meet the requirements. This nerf gun is also symmetrical in the design, which gives the flexibility of having the chassis board and trigger on the center which will help with the dynamics of positioning the nerf gun with respect to the chassis board and the PCB as the critical components needed to operate the turret will be close enough that it will be able to be configured and connected in close proximity to one another. The chassis board will most likely be centered around the area between the ammo cartridge and the trigger as that seems to be where the center of mass is likely to be located.

3.3.8 Bluetooth

The automation aspect will be one and main feature of the automated turret. The other aspect that will make the turret unique is the second feature which will be a hand controlled option allowing the person to control the turret from a distance. When referring back to the 3.2.3 Wireless Communications section, we decided to go with the bluetooth option as its low power usage, speed of connection, and a good enough connection range. The next step would be to pick the bluetooth module for use with the system. We narrowed down our choices to JBtek HC-05, DSD TECH HM-10, HC-05 Arduino, KOOBOOK. Rather than going with a usb that uses bluetooth we decided that using a shield/module allowed us to make a smaller and more compact design without the need to use a usb port.

Starting with the JBtek HC-05 which is an easy setup with the arduino. Since we are using the atmega 328 which is also similar to the one found in the arduino uno making it

a prime module to use. The simplicity of this is another benefit as it is designed for the arduino making the programming similar to the arduinos which saves time. In order to get this setup we would need to install it into the microcontroller and change the settings on the module to allow for a connection. From there we are able to send a list of commands directly from a device such as a phone or computer as long as the device links up with their specified bluetooth software. The Operating voltage for this device is 4V to 6V and the operational current for it is 30mA. The range being the most important part as we need to operate the turret remotely, and with this module the range is around 100m. Along with the range it is a full-duplex meaning we are able to have a communication two ways. It communicates through Universal Synchronous and Asynchronous Receiver-Transmitter, or USART, with a baud rate of 9600.

The second module that we decided to go with is the DSD TECH HM-10. Unlike the JBtek HC-05 this module will require a voltage regulator which is not the most convenient process but it does have a wider voltage range, this ranges from 3.6V to 6V. Similar to the HC-05 the bluetooth range is also around 100m in an open area. Along with the range it also has a similar baud rate which is 9600.

The third option is HC-05 Arduino which is almost the same as the HC-05 but does not support iOS which may cause problems if we end up relying on one. This would be the optimal choice if we were using two arduino boards to communicate with one another as the communication is two way but would require us to use two, one connected to each of the boards. This version has an operational voltage range of 3.3V to 5V and a baud rate of 9600. Similar to the other two options so far the range is also around 100m.

The last option is the KOOBOOK which is probably the best overall for compatibility. When the bluetooth connection is not set up we are able to manually set the baud rate, name, and the password to pair the module with the AT command. Similar to the rest of the devices the operating voltage is 3.3V to 5V and the range also being 100m making it as optimal of a choice as the rest.

Device	Operating Voltage	Operating Current	Range
JBtek HC-05	4V to 6V		100m
HM-10	3.6V to 6V		100m
HC-05 Arduino	3.3V to 5V		100m
KOOBOOK	3.3V to 5V		100m

Table 9: Bluetooth Module Comparison

All of these would be a great addition to operating the turret for wireless communication. Bluetooth being a low power usage device is beneficial for this project as it is one less thing that we would need to worry about if there was not enough power. As shown in table 9 the comparison between the devices shows the main differences between each

device. The range is similar through the devices and the operating voltage range is also similar so in the end we had to see if there were some sort of compatibility issues or if something was a bit easier to use. In the end we decided to go with the JBtek HC-05 as it had the easiest setup and very little time to even start using. Once the module has its credentials set it is as easy as connecting from a device's bluetooth software.

3.3.9 Controller

The controller is the tool used to control the turret in its manual control setting. This is a wireless control that will be used by an individual to control the turret in a close proximity. There are two choices for the wireless controllers, one is to utilize a standard joystick controller and another is to create an app that will be able to connect to the main board microcontroller to control it.

3.3.9.1 Joystick Controller

The joystick controller would be useful as it will have the ability to create a personalized experience in controlling the turret in a user friendly, comfortable, and accustomed way of handling. We could set the microcontroller to connect to the controller and use the inputs of the controller as the means to control the motors. Some options for these controllers could be the popular Xbox series X controller or the Playstation 5 controllers. However, when looking into these technologies, there were very many references to projects and as such, if we were to use these or any other joystick controller technology we would have to look into the libraries of these joystick controllers to find out the output of the controllers and what they represent and these output are translated to movement. Then utilize their bluetooth compatibility with our own bluetooth module and connect the modules to the controller to process the actions used by the user to the microcontroller and then move the motors based on the commands of the joystick controllers. However, a downside to this approach is that there will not be additional features that can be added, instead of the case of the apps.

3.3.9.2 Apps

The other controller is the app controller. This controller utilizes primarily a phone, but can also be a tablet running a specific OS, in which the two main contenders being Android or iOS. The use of the app controller is not only capable of controlling the turret using the phone with designing the app to hold action buttons such as a digital joystick to move around and a fire button for firing the turret, but also allowing more features such as adding photos that could be sent to the microcontroller to then have a set target for the autonomous turret to find when the user sets the turret to autonomous.

3.3.9.2.1 iOS

The iOS app was programmed using Swift, which was then used to create an app that will be able to control and communicate with the main board microcontroller. An iOS app is a

more closed system than Android, which means that to create an app is a bit more difficult due to the focus on security of the ecosystem. However, the iOS can have the same app be utilized in both tablet and phones due to the standardized ecosystem, with iOS having customization options that are common across all of its devices.

3.3.9.2.2 Android

The Android app was programmed using Kotlin and, like the iOS, will interface with the main board microcontroller. The advantage of Android is the ease of creating an app due to being an open system, unlike iOS and their closed system which makes it difficult to create an app and utilize it in the ecosystem. Android is also great due to having a variety of different devices with different modules and peripherals that can be used to achieve the results. There are also the different OS versions that are used as well as many different phones having updates before other phones, which may affect the app depending on the update and having to coordinate an effective time table for updates between different devices. This will not be an issue for the most part due to not having to depend much on the Android ecosystem in and of itself and the updates will not affect how peripherals like bluetooth operate and thus not an issue in our project case, as our project is self-contained.

3.3.10 Programming Languages

The languages will need to be set up for the microcontrollers for the turret, database, the app design for android, and the bluetooth module. To even think about accomplishing this we needed to identify how we were going to build the communication and the interface for the app. To build communication we need to specify the languages that we plan on using for each of the individual devices. Since we are using python it would make things easier to work with as we are already utilizing python for facial recognition. However there are other forms of languages that we can use which will probably be easier to use depending on what we decide to use. These will depend on the devices we are planning to use them with as python works well with training devices for facial recognition. The devices include the arduino, bluetooth, and the app for the phone.

3.3.10.1 Raspberry Pi Software

The raspberry pi software will include a few variations as we have decided to use python for the facial recognition process. Other than the python we have to decide how to program the raspberry pi to use the python codes output from the camera. As we are also using the atmega328 so it would be beneficial to use the languages of C/C++, the transition from programming to each of the boards using the same language makes it easier on two parts. The first part would be sending information between boards that won't have to be translated, and the second is that the use of the same language would require less of a learning curve.

From the three languages python is one of the choices that we are able to use since we are already using this for facial recognition it will probably be the most optimal choice to

use. There are also benefits of using python such as a lower learning curve, efficiency, as well as the coding syntax. Starting with the learning curve, python is a higher level language meaning it is easier to write and read because it is closer to human language. Of course it is a bit slower because high level languages need to be translated to a form that the hardware can read. This form is machine language, and the way they transform it from high level to machine is through the use of a compiler or interpreter. The next benefit of python is the efficiency and the coding syntax that go hand in hand as they work together. Python is an object oriented language which makes things easier to be modular as each section can be made from a class so if something decides to break we are able to narrow down the issue to a class rather than looking through the code troubleshooting.

On the other hand C/C++ is also a choice that we can use to program the raspberry pi. Depending on which language we use between C or C++ things will be easier and the coding variation will differ. The language C one has is not an object but is a bit of a lower end language making it faster than the previous language that we planned to use which is python. Although it is faster there will need to be more code written as it does not support object oriented programming because we are not able to declare a class or a parent and specify a child to that parent which means the child will inherit attributes from the parent. On the other hand, C++ is an object oriented language similar to the language of Python. Even though it makes things slower to compile due to the translation process it is also a bit faster from Python due to it being based on C.

Programming Languages	Object Oriented?	Compilation Speed (1-3 Best to Worst)	Use/Possible Use
Python	yes	3	Facial Recognition
C	no	1	Raspberry pi / atmega328
C++	yes	2	Raspberry pi/ atmega328

Table 10: Bluetooth Module Comparison

Overall the language options are all viable due to the languages that we are already using. Python being a language that we decided to use as they have the best compatibility with the OpenCV library. On the other hand C and C++ is possibly a language that we are going to be using for the development board. In the end we decided to use C++ because the output of the python code can be changed to another form to be interpreted by C++ and it being an object oriented program will make things more efficient to program similar to Python but it will also be faster.

3.3.10.2 Development Board Software

The development board which uses the atmega328 is similar to the arduino which is able to use the C/C++ language. As mentioned in 3.3.10.1 Raspberry Pi Software, there are a few differences between C and C++. Although C is faster due to it being a lower level language, C++ makes things easier to program due to the object oriented factor of the language. Therefore we have also decided to use the C++ language because it makes it easier to communicate between the boards and the speed as well.

3.3.10.3 Bluetooth Software

The bluetooth module that we decided to use is the HC-05 because it is the easiest to set up and the time to set up is fairly simple. The HC-05 is connected to the development board using the atmega328 chipset. To set this up we needed to attach the module to the board and make sure the power is on. Since we are using the atmega328 the bluetooth module will also be using the C++ language. The reason for this is because the module does not work independently but operates with the development board, therefore the board will send out commands to the board which can be transmitted to the app.

3.3.10.4 Phone App Software

The phone app software will differ in a major way from the rest of the components as it is an interactable design as well as working with the android software. For this setup we have a few languages that we can use such as C++, Python, Java, and Kotlin. Each of these will have their benefits and disadvantages. As the android system runs off a linux kernel that was the deciding factor on what language used.

As the language that we decided to use for the raspberry pi and the development board is C++ this is probably the easiest option from the choices. The learning curve will decrease making us work faster as we are using similar code throughout and the bluetooth module being controlled by it as well. There is also the factor of translation between system to system as they would be using the same language. As this is also faster than most other options as it is similar to C except it being an object oriented language.

The second option that we decided to go with is Python as we are already utilizing it for the facial recognition process. As we are already using it and similar to C++ the learning curve will be easier to bypass. The minimal learning curve is beneficial except we were required have to learn to use python to make an interactive app. As this is slower than C++ we did not choose to go with this.

Another option that we have is Java which is another object oriented language similar to Python and C++. Since we are mainly programming the app using an android, we have to understand that they run on a linux kernel. Java is a preferred language for most due to its compatibility with java. There is a vast amount of open-source software to work off of for a use in our app making it easy to develop our own app design.

The last option that we decided to go with is kotlin which is yet another object oriented form of language. Similar to Java it has a higher compatibility than Python or C++ for the android platform. Although they are similar in these factors the kotlin software is able to compile a bit faster and tries to prevent apps from getting too big. This may be beneficial as we increase the amount of features from the base app design; it will keep the storage of the system down.

Programming Languages	Object Oriented	Compilation Speed vs Compatibility (1-4 Best to Worst)	Already being used in previous device
Python	yes	4	yes
C++	yes	3	yes
Java	yes	2	no
Kotlin	yes	1	no

Table 11: Bluetooth Module Comparison

Overall any of these would be a great choice of software development for the android app design. If we decide to use Python or C++ it will give an easier time to program due to the lower learning curve as we are already using C++ for the raspberry pi and the development board. On the other hand Java and Kotlin have better support for the app development on android. Between those two options Kotlin is a lighter and faster version of Java. In the end we have decided to go with Kotlin due to its support of Android as well as the speed and lightweight factor of the programming language.

3.3.11 Database Setup

Facial recognition is one of the main features for this turret that we are designing. As we are using the raspberry pi the amount of RAM that we have is 2GB which is enough to help with the whole procedure for facial identification. Along with that we are using a 64GB micro sd card to install everything including the library and the operating system for the raspberry pi. When designing the facial recognition process storage is not that big of an issue as the information is temporary especially if the person we are targeting is not in the list to be eliminated. However when referring to which target to eliminate we need to make sure that the information is not going to be overwritten by something else or accidentally deleted due to a shortage of storage space. Therefore we decided to set up a database which can be done in two ways. One of these ways is hosting it locally directly on the turret itself and the other option is to establish a wireless connection through remote access to a database.

In order to decide how to deal with the storage situation we have to look at the needs and feasibility as well as the cost for each option. Starting with the local database which is the most reliable and fastest way of having a database set up. Not only is it the most reliable or fastest it is also arguably the cheapest for that we could use for the database. The reason for this is because storage in general does not cost as much to buy anymore, an example of this would be to buy a couple terabytes of storage for less than $100. This also splits in to a few other options that we can go through either maximizing the storage with the size and the price as well, and the other option is sacrificing storage and decreasing the size. Each option has its benefits, one obvious benefit to maximizing the storage would be the storage itself of course. With the storage the price will be higher and the weight will be more which means we would have to adjust the motors. This is also dependent on whether the storage device is an HDD or an SSD, although an SSD is ideal the HDD is the cheapest. HDD usually requires more power to operate and as the size taken by the overall database of images increases the operational rate of the hard drive will decrease. The SSD on the other hand will be faster than the HDD but they cost more and spending at least $100 would only get us a terabyte. As we are on a budget and are using low power devices such as the atmega328 and the raspberry pi for portability the higher storage options are not viable as it would take a long time to traverse through all the data. The best option in this case so that we do not increase the weight too much would be to use an SD card although the storage will be limited, it is enough for the atmega328 or the raspberry pi to traverse and use efficiently.

The other option for storage that we have is a remote form of storage which is the least reliable when looking at it from the point of reliability. The reason for this is because different areas will have spotty connections and depending on the location and quality of the remote database the storage may also be very slow, potentially getting to a point where there is no connection with the database anmore. This would result in the turret not being able to identify who to target in the first place thus compromising the turret's capability. There is also the factor of cost, and in order to get the best performance we would need a two factor investment. The first investment would be the connection either having it hardwired to the internet, having a wireless connection through wifi, and lastly using a cellular connection for a wider range of operation. The best one would of course be the hardwired choice but as portability would become an issue the next best alternatives would be cellular communication and the wifi. From those two wifi would arguably be the best choice as long as there is a strong connection and cellular will have varying connections in different places and potentially none in some buildings.

In the end either option is a viable choice depending on what the requirements are. For this project we felt the best option would be the local storage as getting a proper connection could be a problem and having a fast enough connection can be costly which is not a single time purchase meaning as the length of usage increases so does the price unlike the local storage. The local storage also offers a very stable and reliable connection so that the automated turret can operate even when the internet is down. When referring to updating the database we are able to manually take the storage device out or through the use of the bluetooth module that we have decided to use. Through the app we are able to update the device remotely as long as we are in range of those

bluetooth modules which is approximately 100m give or take. Therefore we have decided to go with the local storage option which would give us total control without really having to worry about whether there will be a signal.

3.4 Parts Selection Summary

3.4.1 Computer Vision Technology

Our choice of which computer vision technology we wanted to use was potentially one of the most important design decisions we have had to make as it significantly limited our choices of MCU. Ultimately we decided to use the Pixycam 2.0 camera to implement computer vision in our project. While the Pixycam 2.0 is not capable of facial recognition, it is easily programmable to recognize colors. By programming our turret to associate a specific color as a “threat” and a different contrasting color as an “ally”, we can allow our turret to have a means of rudimentary target distinction. The Pixycam 2.0 also has a large catalog of support and function options that will allow us to easily troubleshoot any potential problems we may face with the hardware/software combination. The Pixycam 2.0 also allows us to program it in either C++ or python, thus allowing us to implement whichever of the two libraries that we have more experience with. The Pixycam 2.0 also has an integrated light source that will allow us to detect potential targets in a more dimly lit environment without having to complicate our design by adding the additional power and circuitry requirements of wiring an independent light source. Another key aspect of our decision is that the Pixycam 2.0 has a very low power requirement for a hardware/software integrated camera. At only 150mA, there will be very little impact on the battery life of our turret, while adding many features that otherwise would not be nearly as efficient if designed by us from scratch and implemented separately. The Pixycam 2.0 also comes with a configurator that runs on MacOS, Windows, or Linux systems, and allows the Pixycam 2.0’s functionality to be easily programmed from the configurator. The only downside to our commitment of using the Pixycam 2.0 in our design project is that it is only designed to work with two classes of MCU, the Adruino and Raspberry pi family of chips.

3.4.2 MCU

Because we had previously committed to using the Pixycam 2.0 as our computer vision software, we were limited to only the supported Raspberry pi and Arduino chip families. We have also decided that using two separate MCUs will allow our turret to perform the most optimally. Because these MCUs are designed to be very power efficient and small form-factor, they are not designed to handle a lot of processes at a time. If we decided to only implement one MCU to control both the computer vision and the motor controls, we would risk bogging the processor of the MCU down as it struggled to keep up with all of the processes being scheduled. This would make our turret sluggish and unable to effectively track targets. By implementing a two MCU design, we can have a single MCU control the computer vision and interface with the Pixycam 2.0, while the other MCU acts as the controller for the motors and firing mechanism. With this dual MCU

format, the MCU controlling the motors and firing mechanism would act as the slave device, and the MCU controlling the Pixycam 2.0 would be the master device. The slave device would feed all of the relevant data to the master device and they could communicate via a wired connection between the two control boards.

Because our project must also have an element of circuit design, we also needed a family of MCU that would allow us to buy just the microprocessor instead of the entire integrated circuit. This would allow us to design a custom PCB around the microprocessor of our choice to satisfy the circuit design element of our project. Because we are implementing a dual MCU design in our project, we can choose to use the integrated circuit as the master device and our custom PCB as the slave device.

We decided to use the Raspberry pi family of devices in our design as they have a large database of various resources that we can use to help program our project. Our choice for the master device is the Raspberry pi 4, and for the slave device we make use of the Atmega 328p chip. The Raspberry pi 4 offers a lot of processing power compared to other MCUs it competes against, while still being power efficient. In order to minimize incompatibility, we also wanted to use a microprocessor for our custom board that was from the same family of processors as our master board. While we could have technically utilized any processor that uses the same communication protocols as our master board, by choosing a microprocessor that is the same architecture and family as the master board, it allows us to program it in the same coding languages and access the same databases as our development integrated circuit master board.

For the custom PCB portion of the project our circuit board was designed with the various pin-outs needed to supply power to the various components that are being controlled by the master device.

3.4.3 Battery

Because we are utilizing a dual board design, we wanted to have a separate power source for each of the individual boards. While there were ways to power both boards with the same source using voltage regulators, we worried that it would complicate the design if one component failed leaving the entire project without power. Because our Raspberry pi 4 is acting as the master device and is controlling all of the motors, and having to send and receive data from the secondary device, it will require substantially more power than our secondary slave device will. To power our main board, we have decided to use a two-cell Lithium Phosphate battery that is designed to work in high performance remote control cars. We needed to choose a battery that would handle the high current loads of the motors as they try to track a target. The motors will be drawing significantly more power than all other parts of our project combined and were the main component that needed to be accounted for. The battery we have chosen to go with is rated at 7.6V and has a capacity of 2000mAh. The battery also has a C-rate of 20C, and can handle a peak burst current of over 50 amps. While we are not expecting our project to draw anywhere near that much power, it allows us to take liberties with picking the most optimal style of motor for our design.

To power our custom PCB, we wanted to pick a battery that would not require as much regulation and we wanted to have the power connector be as simple as we could possibly make it to maximize the options for powering the slave device and Pixycam 2.0. We decided to utilize a portable battery bank made by Anker Power to power the secondary MCU. The Anker Powercore has a much larger capacity at 10Ah and it is rated at 5 volts at 2.4 amps output. Together this will allow us to power anything up to twelve watts. With our MCU consuming less than half an amp, and the Pixycam 2.0 consuming only 140mA, we have more than enough leftover current headroom to power some other peripherals on the secondary board. Because the Atmega328p chip only requires an input voltage of five volts, we can utilize a single voltage regulator or rectifier circuit to drop the constant output voltage of the battery from 7.4 volts to the desired five volts.

3.4.4 Motors

For our motor selection we have decided to use the MG995R as our pan and tilt motor. This servo is powerful enough to bear the weight of the entire upper platform and rotate the gun enough for us to have a good range of motion. The MG995R also has enough stall torque in order to adjust the tilt of the gun and hold it in a fixed position. The MG995R has an operational voltage between 4.8V-6.0V so we do not need to use a separate voltage regulator in order to feed these motors a different level of voltage than the rest of the components on our PCB. These motors will require a lot of current due to the amount of forces they are rated for. To compensate for this, we have decided to use a voltage regulator that will output a high enough current to satiate any potential current requirements we may have from these motors. We also have compensated for the increased current draw by using thicker traces in the VCC lines that feed the motors.

We have chosen the SG92R servo motors to operate the firing trigger of our Nerf gun, and the safety switch on the gun. The safety switch of the gun requires a force of approximately 88 grams in order to fully depress the trigger and allow the gun to fire. The safety trigger also has a shorter pull length than the main firing trigger at only 1.2cm. The main firing trigger has a pull length of 2.8cm and requires approximately 213 grams of force to fire a dart. The SG92R servo also is capable of running at 5V and only has a peak current draw of 650mA.

3.4.5 Range Detection Devices

When researching range detection technologies we had originally considered using ultrasonic sensors to measure the distance from our turret to any potential targets. Ultrasonic sensors are both very affordable and have incredibly low power consumption. However, due to their relative inaccuracy at detecting smaller objects or distinguishing an object from other things in its environment, we ultimately decided to look into other more advanced means of range detection. We decided to pursue LIDAR technology as it has become significantly more affordable, offers much higher levels of accuracy and environment/target distinction, and retains a low power usage.

The specific LIDAR module we are going to use in our project is the SMARTLAM TP-solar. The SMARTLAM LIDAR module has a standard five volt input and only draws 80mA of current, resulting in a power draw of 0.4 watts. The SMARTLAM also has an operation range of 0.2-12m. Because we are range limited by our projectile launcher to around ten meters, we did not require the extended range that more premium LIDAR modules provide. Having LIDAR range detection on our turret will allow us to track potential targets and only have our turret fire on a target once it is within our turret's most effective range of 5-8m.

3.4.6 Firing Mechanisms

Because of the complexity of having to design our own firing mechanism, we wanted to source one that was already developed and that could be easily implemented into our project. The Nerf platform by Hasbro provides an effective solution to demonstrating the effectiveness of our design, while keeping the project nonlethal and minimizing potential injury during testing. We are utilize the magazine and the firing mechanism from the Nerf Doomlands blaster as it is motorized and will allow for continuous firing without having to manually chamber a round after every shot.

4. Related Standards and Design Constraints

4.1 Standards

4.1.1 Relevant Standards

UN/DOT 38.3 5th Edition, Amendment 1

UN/DOT 38.3 is a safety standard set by the United Nations Department of Transportation that regulates the testing around the transportation of all lithium metal and lithium ion cell batteries. This standard sets the conditions of rigorous testing that must be passed in order for a lithium cell battery to be deemed legal for transport. Lithium batteries can be a potential fire or explosive hazard if their enclosures or cells are ruptured and therefore must pass a series of intensive mechanical, environmental, and electrical tests to minimize the risk of injury or accident during transport. These tests include altitude simulation, vibration, shock, impact, overcharge conditions, external short circuiting, forced discharge, and thermal testing. The testing associated with the UN/DOT 38.3 standard is typically performed by an accredited test lab such as MET labs due to liability issues.

IEC 62133

This is a de facto standard for international compliance of end-devices. Because this standard is for the compliance of end-devices, this means that the testing and regulations imposed by this standard are performed and held up to the electronic devices that will contain the lithium cell batteries. This means that the passing of the UN/DOT 38.3 testing standards is a core requirement to also satisfy the IEC 62133 standard.

UL 2054

This standard is a mandated U.S end-device testing standard with more rigorous testing than either the IEC 62133 or the UN/DOT 38.3 standards. The UL 2054 standard is made up of eighteen separate tests that stress all aspects of the lithium cell battery and the electronic device that it is enclosed in.

ASTM F963

According to this standard as said in *4.21.2.3 (Projectile Toys with Stored Energy)*, “Projectiles shall not have a kinetic energy per unit area of contact, also known as Kinetic Energy Density (KED) greater than 2500 J/m2 when tested”. This is set in place to reduce the risk of injury when playing with projectile toys. Because we intend to use the entire firing mechanism from the Nerf Doomlands blaster without any alterations, our turret will meet the standards of classification of a projectile toy with stored energy.

Firearm Related Standards

Our project is compliant with all ATF related firearm standards and regulations. As told in the ATF's firearm classification in the document *26 U.S.C. 5845; 27 CFR 479.11*, our turret cannot be classified or regulated as a firearm as per the ATF's definition of a firearm stating, "a device or weapon that is designed to expel a projectile by the action of an explosive". Although the Nerf gun resembles a firearm, it fires its projectiles using two rotating barrels that speed up the projectile as it is pushed between them.

4.1.2 Design Impact of Relevant Standards

Because we have chosen to source already developed products for the batteries and the projectile launcher, the standards listed above have not had any significant impact on our design. Had we decided to construct a battery or try to build a projectile launcher while adhering to the standards listed above, the design process would have taken significantly more time. Because we sourced products from manufacturers that are already required to adhere to these standards, they are required to build their products to meet the specifications of said standards. Because we are not altering these products in a way that changes the way they operate, these standards will hold up and will still be met. Had we built a battery ourselves and still tried to meet the UN/DOT 38.3, IEC62133, and UL 2054 standards; we would have had to design our battery to withstand the rigorous testing required by these standards. By sourcing the batteries from Venom and Anker, they have already tested and met any relevant standards for portability and end-device testing.

4.2 Realistic Design Constraints

4.2.1 Economic and Time Constraints

There are a few ways of implementing the facial recognition process. One of the most effective ways would be to incorporate a GPU to increase the number of calculations and have a better performance-based machine. The involvement of a GPU would require a PCI express slot to connect with the CPU and memory. Not only that there are restrictions on power as well as creating a system where the CPU is able to communicate with the GPU and utilize its resources effectively. With the addition of these components there will be an increased demand on capital as this would require a dedicated GPU, a reliable CPU, a bigger power supply, as well as the increase in weight to support these upgrades.

Utilizing the PCB for facial recognition would require the development of a computer. With the limited budget and the time required to create the project this is likely not feasible. Hence the use of an external device that will perform the calculations and allow the transmission of signals to the turret for operation.

Another aspect is the electrical power usage for the device to function. There are two ways that we can supply power to the device, and they are the standard household outlets that are able to provide around 120V AC as well as a rechargeable battery pack. The outlet to be of use would require an AC to DC converter, however it will increase the

chances of a power surge. On the other hand, a rechargeable battery would reduce the chances of a power surge. In turn it would have a limit charge that would have to be either monitored or we would have to manage and try to draw out the time of operation by managing power consumption.

The battery depending on the size can be costly as a bigger battery would result in more resources to use and handle the battery. In respect to mobility and design, we have decided to go with a battery powered system in order to reduce cost and increase mobility.

4.2.2 Environmental, Social, and Political Constraints

Environmental: The main impact our project will have environmentally is from its power source. Our automated nerf turret supports hot swappable battery packs that are Lithium-ion.
These batteries are much less toxic in the metals they contain as compared to those containing lead or cadmium. They contain elements such as cobalt, copper, nickel and iron. These types of batteries are quite costly to produce in its labor making them the least likely to be recycled, however they are safe for landfills. Extracting lithium requires a lot of resources and requires a substantial amount of water to do so. Recycling remains a more costly process than the extraction of the materials through mining. There would be more environmental constraints if the choice of battery we made contained more toxic metals or materials which would pollute the ecosystem or have harmful effects on humans or wildlife.

Political and Social: In many parts of the world, firearms have become something looked down upon in society while they remain necessary tools for citizens' self protection and military power. Our nerf turret meets all political restraints regarding laws, regulations and policies. Our nerf sentry turret resembles a firearm and is only used for testing and fun and is in no way intended to harm an individual. It will not be fired at any targets that are unwilling to participate nor in any vulnerable areas such as the eyes. There are no constraints socially or politically related to firearms here as we aren't using real firearms for the turret and we are of age to purchase nerf guns.

4.2.3 Ethical, Health, and Safety Constraints

Our Turret project is not inherently dangerous as we are essentially just automating a toy Nerf gun. Some potential safety constraints are that we should include a safety switch to prevent the turret from arming itself and firing before we intend it to. Anyone who will be positioned in front of the turret should also be required to wear safety goggles to reduce eye related injuries. Because our project contains a device that resembles a gun, we made sure to ask campus security to ensure we are able to bring the device on campus. Our turret should also not be fired at any unsuspecting individuals in order to reduce risk of accidental injury. Because of the volatility of the Lithium Phosphate battery, we should also mount our batteries away from any potential heat sources such as the two MCU processors, which could become hot to the touch after running for several

minutes of intense load. Also to prevent unwanted strain on the power system, the turret should not be run for longer than the ten minutes specified in the design constraints.

4.2.4 Manufacturability and Sustainability Constraints

Manufacturability: The automated nerf turret manufacturability constraints have been increased due to the current chip shortages which makes obtaining the hardware components of the PCB and obtaining MCU's and other electrical components more difficult to obtain. This means that we must be extra careful in the testing and utilizing of the electrical components as replacing them could prove difficult. This may also prompt the purchasing of greater quantities of the components in order to make sure we have the supply without needing an emergency purchase. Another constraint due to current circumstances is the supply chain, which exacerbates the problem of the chip shortage that has made an even greater increase in the need to plan ahead and properly perform testing as the time period of many components increase, as well as the demand increase which would bring more people to buying components in larger amounts and a decrease in supply will be detrimental to the project completion.

Sustainability: The nerf turret must be able to complete all its tasks without burning out or the software does not hit a condition in which the operation of the turret ends up in a non ending loop in order for the turret to be able to perform without fail for as long as the turret has power to operate. The turret will follow the RoHS compliance for any component that is offered and is compliant. RoHS compliance means that harmful substances are reduced, which include Admium, lead, mercury, hexavalent chromium, polybrominated biphenyls, polybrominated diphenyl ethers, bis phthalate, benzyl butyl phthalate, dibutyl phthalate, and diisobutyl phthalate in which all these substances must have less than 1000ppm except for cadmium which must be less than 100ppm. There are, however, components that may have exceptions based on its operations. There is also the exception of components that do not have RoHS and that are no longer available for purchase, in which case we had to make those exceptions and continue without the compliance. However, we expect most if not all components that we obtain to be RoHS compliant.

5. Project Hardware and Software Design

5.1 Overview of the Turret

The BRO-NS turret project will need to be able to fire at a target within 15 meters, with dual axis motion that covers roughly 90 degrees of vertical motion and about 135 degrees horizontal rotation. In this section, we will now be discussing and demonstrating the designs of the project to accomplish these requirements and complete the goal of the project which is to create an automated turret with wireless control as an option. A top view methodology of how the project will function is shown in figure 13.

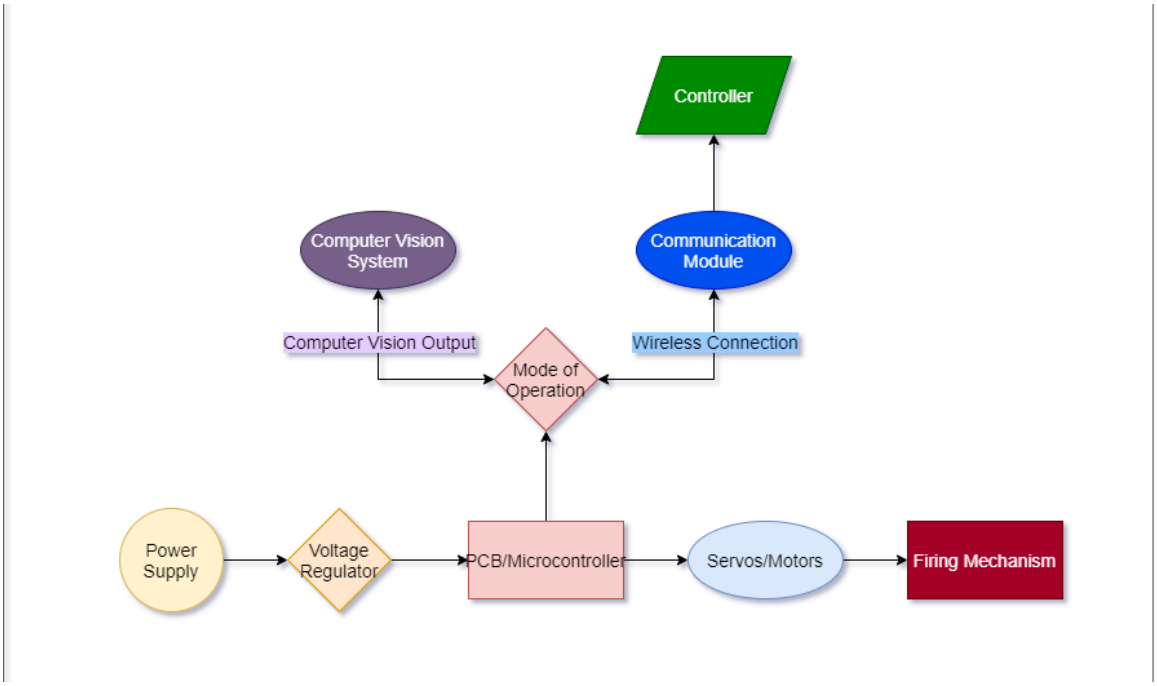


Figure 13: Project Design Block Diagram

The diagram shows the overall project design components and operations. The PCB holds the operation of all the components and has two modes of operation, the computer vision system and the wireless communication. The PCB is split into two separate MCU, one is the custom PCB with the Atmega 2560 and the other being the raspberry pi 4.

The computer vision system, controlled by the raspberry pi 4, should allow autonomous control by utilizing the camera to find the target when in its view and send data to the custom PCB which then controls the motor to move the dual axis chassis board to the target which finally sets the firing mechanism of the nerf gun to fire. The other operation is the wireless control which would set the computer vision system into an idle state and allow the user to control the turret. Each component and operating mode will have it's own personal section to provide a specific detailed explanation on how these operating modes and components will accomplish its specific task and how it will be designed in order to complete said task.

5.2 Motors

The motors are essential for the operation of the turret, since they are the mechanism in which the nerf gun to the target and fires at them. In order for the turret to have the ability to move to the target the motor needs to have the necessary torque to move the components. In order to find this torque, we look into the stall torque and the stall current. For both the motors in question, we find out the torque that will be required and will calculate how much weight can be rotated by using the equation to find the weight in which the motor can rotate.

Torque (in kg-cm) / distance (radius in cm) = Supported Weight (in kg)

The current of the motors at stall torque will not hinder the battery in the slightest as the current draw for either of the motors are roughly about 1-1.5 A. This means that the motors will not have a drastic impact on the powering of the turret and we can safely and effectively power them without worry. However, we do not want to reach the stall torque due to it reaching the limitations of its operation, as such we needed to utilize this equation and testing in order to figure out whether the motors we have acquired are sufficient for our needs without overexerting them by hitting the limits of its capabilities.

5.2.3 Horizontal Motion Motor

The motor for the movement of the firing mechanism in the horizontal axis is oriented in a vertical position, in which the horn that rotates the chassis board faces upward, perpendicular to the ground floor. The motor will rotate the chassis board directly and thus will utilize its full torque with an immediate response and with full torque. This will also diminish any variability in inefficiencies when involving torque over a distance, for example if we were to use a pulley method. However, even without any diminishing returns, the direct application of the motor will reduce the distance and form factor of the overall turret.

This motor is attached below the chassis board and is held by a platform offered by a stand that would make the motor be able to interface with the chassis board at just the right height to rotate the chassis board directly. The motor should have enough torque to be able to rotate the entire chassis board, however that will also need to be tested. If one motor is not enough to rotate the chassis board, we can add more motors and create a bigger platform offered by the stand. This will, however, increase the need for careful application of motors such that the motors must be in sync, but for the time being, one motor should suffice. This is due to the stall torque of the horizontal motion motor reaching 11kg-cm, which if directly connected, will utilize most of the torque available to it.

5.2.4 Vertical Motion Motor

The vertical motion motor will serve to move the chassis board with the nerf gun attached in the vertical axis when powered to do so. The motor is connected to the stand connected to the chassis board to rotate the chassis board vertically. The vertical motor is mounted below the horizontal motor and will rotate a section of the stand that will have the ability to pivot in a vertical position to move the entire chassis board in a vertical pattern. The microcontroller will have control over the operation and positioning of the motor using a PWM to power the motor in the position desired based on how the signal is provided. The motor has a 90 degree rotational range that will help with the navigation of the nerf gun to the target based on the microcontrollers command. We also positioned the motor in such a way that degree 0 is not perpendicular to the ground, but a bit higher sloped than the ground.

The reason we do not want the 0 degree of the turret to be perpendicular to the ground is because the firing of the ground does not have any useful applications, and because this turret is utilized on solid ground and not a turret based on flight we do not have any application of use to have to lower the turret much, other than an edge case. We placed the motor in which it does have some downward motion in case it is placed in a position where there is altitude and the target is below. This, however, is the edge case and the assumption will mostly be that the target is on the same ground as the turret.

5.3 Main Board Custom PCB

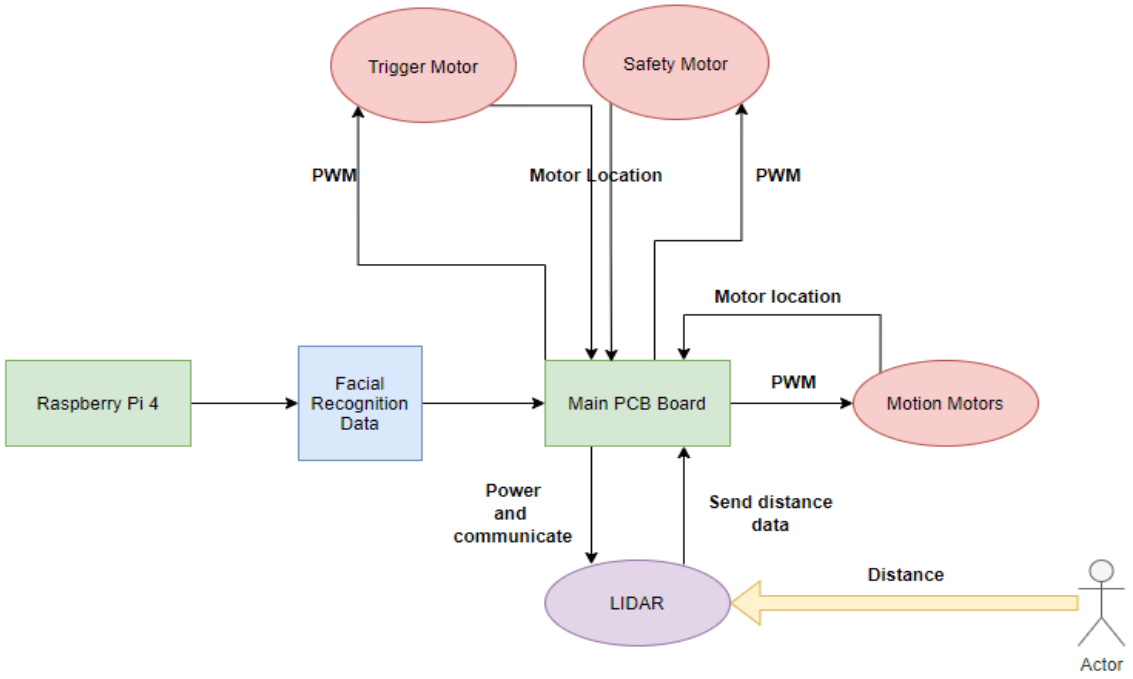


Figure 16: Main Board Block Diagram

The main custom PCB board is in charge of commanding the entire operation of the turret project, and specifically takes control of all motor functionality as well as the LIDAR peripheral. The main PCB will begin with starting the communication with the raspberry pi 4, which will then have its own operation and then return the communication with the data the main board requires; the facial recognition and location of the face based on the camera imagery. The data will be discussed in more details in the software section of the design. While the raspberry pi is running, the main board will be in idle sleep mode. Once the pi is finished processing the data, it will send it to the main board. The main board will wake up and take the information then it directs itself to the LIDAR.

The LIDAR will be used to find the distance between the target and the turret. This is essential because of the need to hit the target effectively, we must move the turret to a position in which that is feasible. Once the LIDAR senses the distance, the motion motors are then set to move the chassis board, with the nerf gun holstered, to the appropriate

angle of fire that the main board calculated to be the most precise location for a successful firing of the nerf dart. With the main board controlling the PWM and the range finder, there will be less latency issues and more coherent communication. Having the raspberry pi control the computer vision will add a bit of latency but that will be more beneficial as the raspberry pi will be able to work on the computer vision in parallel to the main board. The raspberry pi will also continue the computer vision operation concurrently to the main board in order to update the location of the target in case the target moves from one location to another. When there is movement the pi will cause an interrupt to the main board and it will give a new coordinate for the main board to drive the motors to moving.

The PWM is an important signal that the main board will be in charge of providing for the motors. Testing will have to be done in order to find the right duty cycles, however, there is a multitude of PWM modules in the Atmega 2560 to fit the safety and trigger motors, as well as the motion motors, both the vertical and horizontal axis motors.

5.4 Slave Board

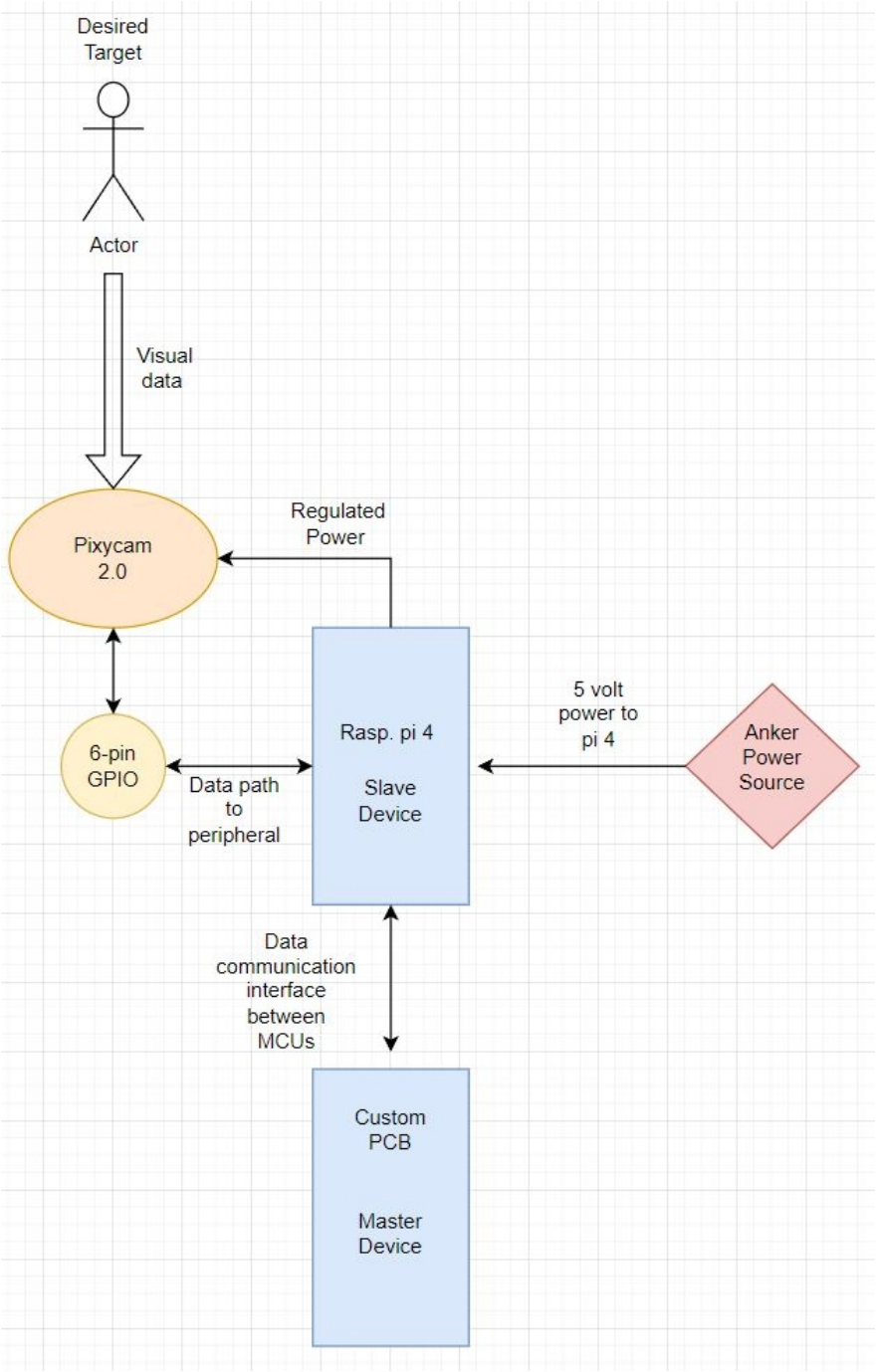


Figure 17: Connection Schematic for slave device

The figure above shows the basic connection diagram of our secondary 'slave' MCU. For power, the Anker power supply will supply the 5 volt input needed to power the Raspberry pi 4. The Anker battery bank can supply up to 2.4 amps of current with the Raspberry pi 4 only drawing up to 0.6 amps and the Pixycam only drawing 0.14

amps.The Raspberry pi 4 has built in voltage regulation that will allow the pi to send the required amount of power to the Pixycam. The data exchanged between the Pixycam and the MCU uses a wired interface through the micro-USB interface. The pi has a total of twenty six GPIO pins and the Pixycam uses a six pin connector to provide power to itself. Of the Raspberry Pi's forty available output pins, twenty six are GPIO, two are 5 volt supply, two are 3.3 volt supply, and seven are ground. The Raspberry pi 4 will also be tethered to the master MCU via a micro-USB cable.

The data path for the secondary MCU is as follows. The Pixycam has been programmed to interpret a specific color as “hostile”. When the Pixycam detects a hostile color, it will send the relevant data to the Raspberry pi 4. The pi 4 is then used to interface with the master MCU. Having the secondary MCU handle all of the visual data will relieve some of the workload that would otherwise be put onto the master device. We had also considered having the secondary MCU control the range detection of the target, but because we want our master MCU controlling all of our motors and movement, we wanted to minimize the amount of time that would be spent sending data between the two MCUs. Having too much load placed on a single MCU could potentially overwhelm the registers in the MCU resulting in the microcontroller being unable to keep up with the commands needed to execute the movement and tracking correctly.

The Pixycam is programmed using an included utility and only interfaces with the MCU to send the relevant data. The Pixycam is being powered by the regulated 5 volt micro-USB input coming from the Raspberry pi’s GPIO pins.

From the GPIO cable included with the Pixycam, the Pixycam will require six total pins of the Raspberry pi’s GPIO output.

The Pixycam uses the USB interface to transmit data to the Raspberry pi because the pi has more memory and can handle the larger memory footprint needed by the USB protocol. If we decide to use an MCU other than the Raspberry pi 4 the Pixycam will then have to use a serial data interface to communicate with the MCU instead of USB protocol. For instance, with the Arduino family of MCUs, the PixyCam uses an included cable to communicate with the Ardino using SPI. SPI is the preferred serial data interfacing method due to the amount of data that could be shared on the SPI interface. The Pixycam is able to detect and process images at fifty frames per second and can detect one hundred and thirty five objects per frame. When the image sensor is running at fifty hertz the Pixycam captures images at 640x400 resolution. The maximum resolution of the Pixycam can be set to 1280x800, but this only allows the Pixycam to record images at twenty five frames per second.

Using the larger resolution will make it easier for the Pixycam to detect smaller objects, or objects that are further away, but this increased resolution requires more CPU and memory resources to encode and send the data. When processing images at the 640x400 resolution, the Pixycam must be able to process and send roughly 1Mbits/sec of data through the serial interface. This is easily achievable through SPI as most Arduinos have a SPI clock rate of 1MHz, which can be increased if needed. When using a slower serial

data interface such as UART however, there may not be enough time for the Pixycam to fully process the images, and send the images across the UART interface before the next frame is detected and processed. To maximize the efficiency of the data processing only the newest data is processed and sent, resulting in older, unprocessed frames getting thrown out. The Pixycam is also programmed to prioritize larger objects, resulting in smaller objects potentially getting lost or undetected if the serial data interface cannot process the images quick enough.

The serial data interface can be manually selected using the included "Pixymon" software. The Pixymon software is a Windows and Linux executable program that allows for the changing of fundamental features of the Pixycam without having to change the data interface using coding functions.

5.5 Chassis Board

The chassis board has been designed and built on our own by obtaining parts such as platform surfaces, walls and a roof for housing and other components to build the chassis that will have space to separate each component while having openings for wires to connect the microcontrollers, motors, and the nerf gun together. The start of the design of the chassis board is a flat surface that will be the chassis boards' support and the platform that will connect all the different components together. We compartmentalized the chassis board with additions and separations of the chassis.

5.5.1 Microcontroller compartment

The main board microcontroller is housed by the chassis board surface and separates the two microcontrollers inside the housing with enough space to comfortably connect the components while also holding the microcontrollers in place. The microcontrollers must be holstered because the turret will be moved between different locations and it must be movable. The microcontrollers are housed with the horizontal motor as well. The motor is inside and rotates the chassis board for horizontal movement and as such will be placed at the center of the chassis, while the microcontrollers are placed on either side of the motor, with the custom main board connecting to the motor and controlling it. This housing will allow for the microcontrollers to have connections to other peripherals that are not going to be inside the housing but the housing must be able to protect the microcontroller from any form of weather so that in the case of rain or other hazardous weathers, the microcontrollers will not be affected.

5.5.2 Firing Mechanism holder

The holder of the firing mechanism is shaped to hold the nerf gun upside down in order to have the nerf gun magazine be able to be detached without having to go through difficulty. The holder is mounted above the chassis board's housing and as such the chassis board should have the means to support its weight along with the nerf gun, and the vertical rotation motor, which is situated underneath and in between the twin armatures that will hold the nerf gun in place. These armatures are fastened to the nerf

gun in a way that will produce little to no sway so that the nerf gun will not have any variability in firing, and will not become a hazard. The motor for the vertical rotation will be situated near enough to the nerf gun to rotate the nerf in the vertical axis to match what the microcontroller tells the motor to turn to.

5.6 Turret Stand

The turret stand is attached to the chassis board and will provide the support to keep it above the ground by being the platform where the chassis board will be held up. The stand will need to support the weight of the nerf gun, the chassis board, and the microcontrollers and their peripherals. The two micro motors weigh 14 grams, the motion motors weigh 62.41 grams each, and the nerf gun weighs 2lbs, or 907.1847, LIDAR is 4 grams, the pixycam is 10 grams, the batteries weigh 68.03886 grams for the LIPO and 354.369 grams for the Anker Power battery. The total weight the stand needs to support is approximately 1496.41 grams without taking into account the actual weight of the chassis board and the armature that will hold the nerf gun.

There are two different types of stands that we looked at in order to maintain the balance of the turret. The first topology was a tripod that would provide stability by finding the angles necessary for each leg to keep the balance and distribute the weight evenly. The other method was just a box stand that would be large enough in width and length to maintain the entire turret. The method we decided to use is the second option. A box like stand is far easier than to create a new tripod of our own, especially since we built the entire mechanical parts ourselves.

The height of the stand is about eight inches and has a width of about twelve inches. This is the preliminary model which will then be tested and attempted to see whether it is capable of maintaining the balance and weight for all aspects of the turret operation, from the horizontal and vertical movement, to the firing of the turret, and whether it will be able to maintain its connection to the chassis board.

5.8 Software Design

The software design for this project is a crucial part of this project as the facial recognition itself is only possible with the adjustment of the code. This section will go into depth about the different methods expanding on the previous methods that were mentioned before. By relying on vectors and cross referencing with the samples that were used to test and apply the results from those tests to recognize a face from the provided camera feed. From there the software will break into a few parts that will play alongside each other in order for the automated targeting system to work. These are broken up into three parts which are Facial Recognition, Motors, and the Distance Sensor.

5.8.1 Facial Recognition Software

Starting with the facial recognition side of the software we decided to use the OpenCV package. The way that the software will work is by detecting edges and comparing those results compared to the testing that was done with other pictures. This facial recognition software will be further broken down into two sections. One will look to see if there is a face detected and the second step will come into play once a face has been detected to cross reference with the database to identify a person.

5.8.1.1 Facial Recognition OpenCV

The main software library that we decided to use was the OpenCV which supports the execution of machine learning which is a beneficial aspect of facial recognition. In order to even achieve facial recognition and try to interpret what defines a face machine learning would be required meaning it has to be trained. Otherwise we would be using the software to match content for similarity, in which case a person who has shifted ever so slightly will not be flagged as a person. Therefore machine learning is essential for this because it will cycle through a database of faces with an algorithm which will adjust the parameters on how the numbers will perceive a face from the pixels.

When creating a program that can tell the difference between what a face is and what is not is very important. For this reason we have decided to use Python as the base for programming the microcontroller to detect faces. Python provides a myriad of seamless library integrations; some of these include dlib, face_recognition (facial recognition utility used with dlib), and of course the previously mentioned OpenCV. The other aspect that is a great aspect of python is its ability to NumPy which is great in mathematical operations. We perceive pictures as a color which include blue, red, white, and many others whereas technology perceives them as numbers. Therefore each color will have a predefined number assigned to it which is what is used along the NumPy operations. These numbers are inserted into some sort of array and cross referenced to the data created from the machine learning side to see if there is some sort of similarity.

5.8.1.2 Facial Monitoring

The Facial recognition process will need to utilize a camera in order to get visual information. Some forms of facial recognition require a photo however we are implementing a live feed to obtain frames which will be processed to obtain facial information if any. Due to the turret needing fast information to continuously track and identify people we have decided to use the Haar cascade. The Haar cascade relies on edge detection to identify objects which in this case will be faces of people. OpenCV is very beneficial in this aspect as it has pre-trained models which will make it easier to identify facial features such as a face or even eyes.

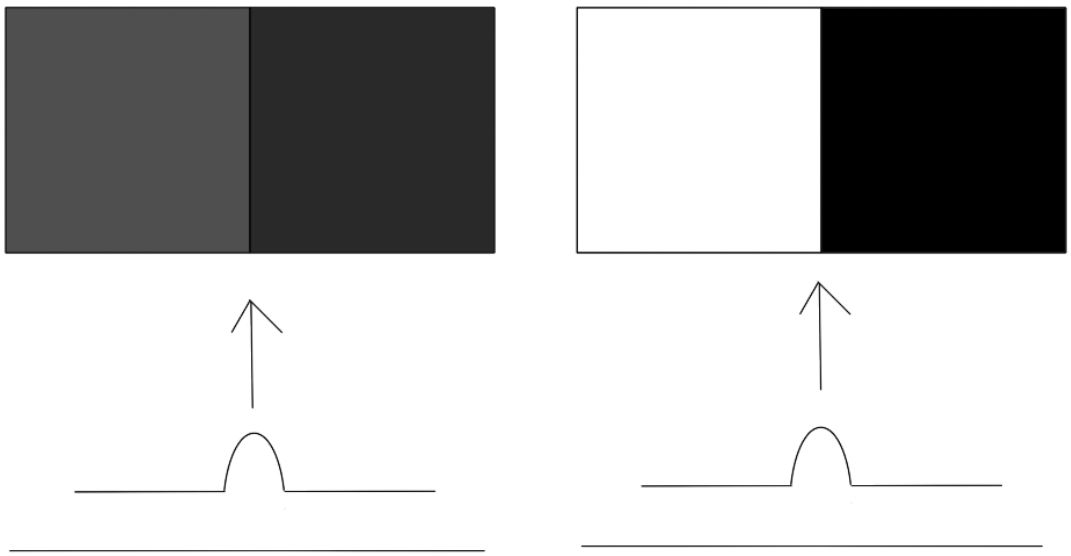


Figure 18: Edge Identification

The OpenCV trained models help look at these features by identifying edges. In order to identify these edges the camera will look at the color differences between different sections on viewable areas from the camera. Referring to the image of figure 18 we can see how an edge is defined. The color differences between a section is what is identified as a form of an edge as it marks the start of a new section. Now there are specific patterns that the code will be looking for which can help define a face.

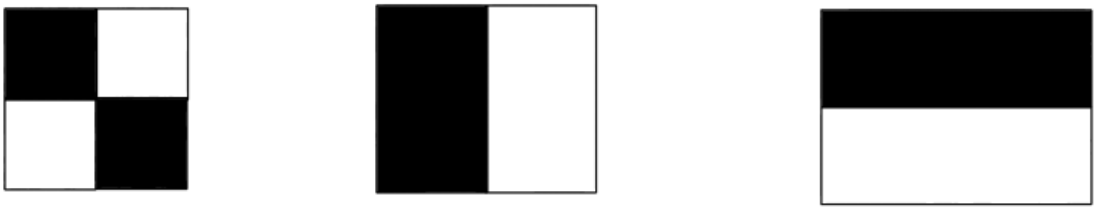


Figure 19: Defining an Edge

The pictures are cross referenced with defined edge sections. Some of these edges are shown in Figure 19 which shows how an edge can be shown. The figure to the far left references a diagonal line or a form of a curve, the middle references to a defined vertical line, the image to the right is a horizontal separation of colors. In order for the turret to even determine a face it needs to run some calculations with preset calculations from a database which are usually in the form of pictures. It is usually a three step process for setting up the determining factor on whether a face is present and where it is. The first

step is taking an input of pictures and letting the program or algorithm determine whether there is a face present in which case it will focus on it. The second is Taking in the results to see the accuracy of it and letting the algorithm redetermine its parameters for what is a face and what is not. The last step is using the final calculations to determine the pictures or live feed coming from the camera as a face or not.

Another step for facial monitoring is keeping the view of the face if it has been detected. In order to accomplish this the algorithm will do scans for the whole range of the camera's viewing angle and get the location of the face in regards to the camera's view. From there the next step would be to keep the face centered allowing the camera to have an easier time following people with the motors. This situation causes another problem where the turret algorithm will not know who to follow if there are multiple faces being detected. In order to resolve this issue we have decided to include a feature to determine if the person is worth tracking. In order to establish this worth we are going to establish a way to have facial identification.

5.8.1.3 Facial Identification

The facial recognition will be a bit tricky in this factor. We would not want the system to create a new identity each time a person is detected. If this was the case we would require a large enough storage device that would take in all the potential faces and store them for later use. This would be inefficient as storing information that will not be used would just be a buffer in the system that would slow it down on image processing and recognition. The reason for this is because the image received will be compared to each of the photos in that database. In order to keep the system running efficiently without using a myriad of resources and energy, we have decided to reference the images with potential targets only.

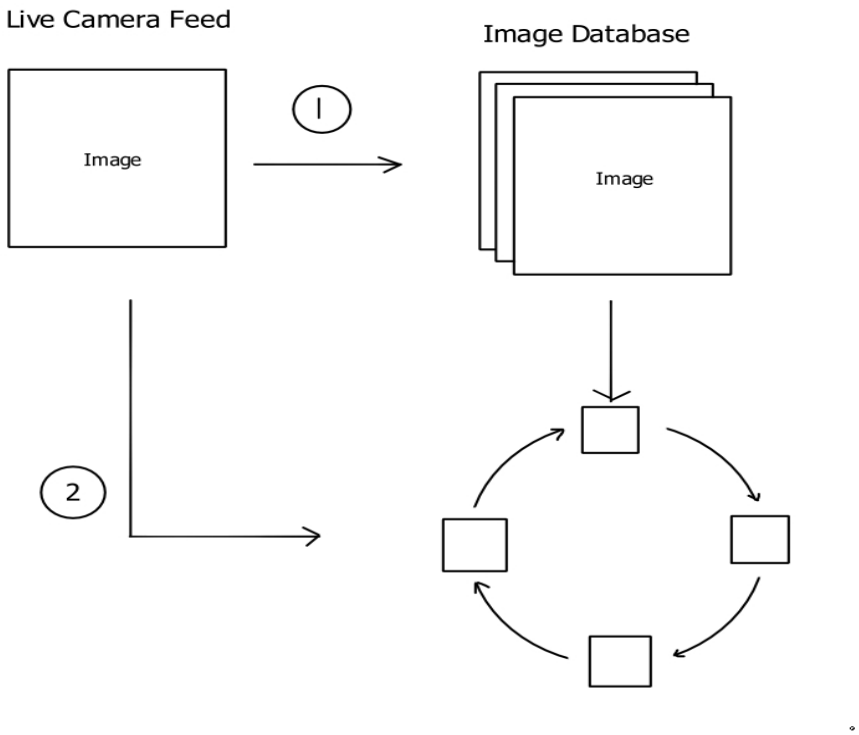


Figure 20: Image Recognition Process

Referring to Figure 20 we can see the process for how the image will be compared to the stored faces in the system. The process for identification starts from the live camera feed which will be scanned to see any faces in the feed. If there is a face detected then it will have to be stored in the system along with pre-stored images that we have determined as potential targets. The reason for this is because it allows for easier access and allows us to make sure that the picture will not be lost if there is some type of error that occurs while the turret is active meaning it can pick up from where it left off.

This will also create a temporary storage so that the camera will know that the person has already been processed. Therefore if the person appears again they will be ignored while the actual target will be followed with the use of the motors. The second step will be to isolate the image from the feed and cycle through the images in the database and compare them for the similarity. If there is one, a flag will be established so the turret will begin tracking that face instead of the others that are around it using the motors to help establish the face in the center of the viewing angle.

This process will take advantage of the database as the people that it needs to target increases so does the size of the database. Therefore we have decided to have a local database giving us complete control over our project through a local connection. The database is attached to the raspberry pi to handle the information along with the visual processing. On the other hand since the bluetooth processing is running with the development board, we also needed to provide access to the database as well as it will be communicating with the app on the android device.

5.8.2 Motor Software for Autonomous Control

5.8.2.1 Motion Motor

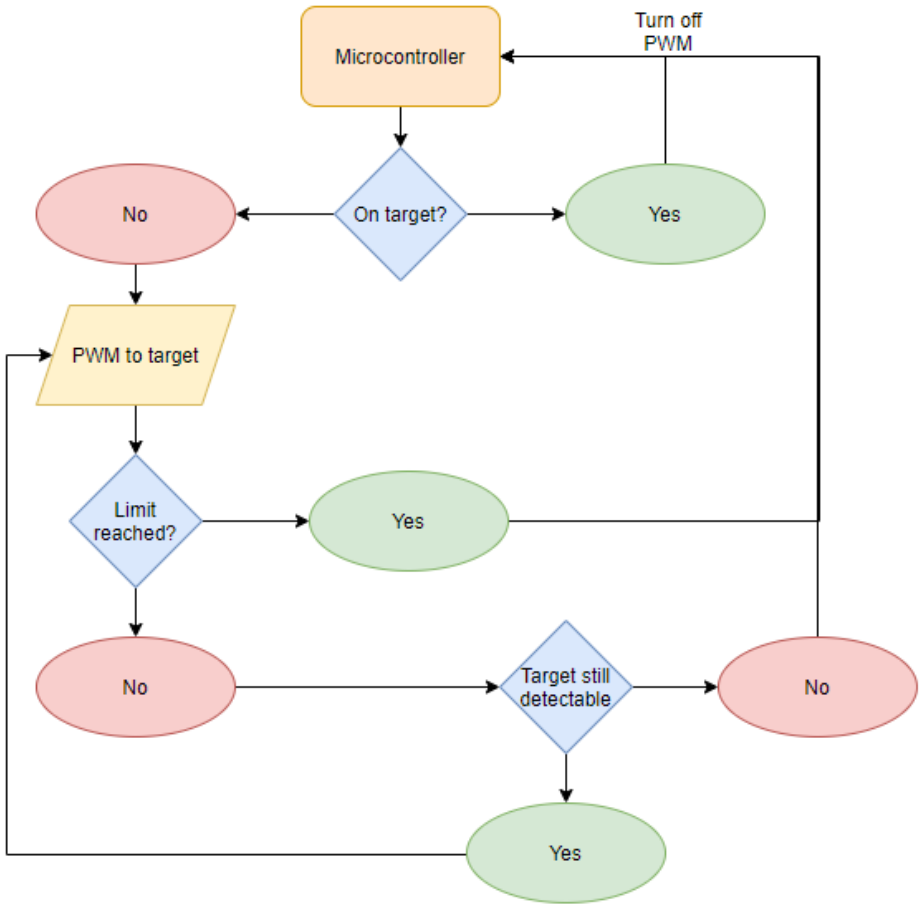


Figure 21: Motion Motor State Diagram For Autonomous Control

The motion motor state diagram describes the operation of the motors and the checks that are necessary for the operation of the motors. It starts with the microcontroller checking whether the turret is on the target, if yes then the microcontroller will not start the pulse width modulation signal, which would move the motor. If the turret is not on the target, then we set the pulse width modulation and have the microcontroller control the direction using both the horizontal and vertical motors in tandem in order to move the nerf gun to the target. There is however, a limit to how much the turret could move, which is around 120 degrees for the motion motors, as such we need to check on that limit and make the turret stop when that limit is reached.

The final check is whether the target is still detectable. If the camera loses sight of the target then the turret should stop all the motors and turn idle.

In order to get to the target, the pulse width modulation will have to be set for the horizontal and vertical case to get to the target by setting duty cycles that will set the speed for each motor. With the motors having a 0.16 second per 60 degrees with a voltage of 5V, we determined the speed of each motor to be set, with the direction, based on how far the motor has to go horizontally and vertically to reach the target. In order to achieve this, we needed to model a two dimensional coordinate.

One dimension is the vertical distance needed to reach the target, the second dimension is the horizontal distance needed to reach the target. The distance from the turret and the target may be needed so that the turret sends the nerf dart in such a way that it will reach the target by setting the turret aim higher than the target in which the dart will not drop earlier due to gravity. The higher aim distance was determined through testing and seeing the drop off of the darts between each distance in order to determine the need for corrective aim.

5.8.2.2 Safety Motor

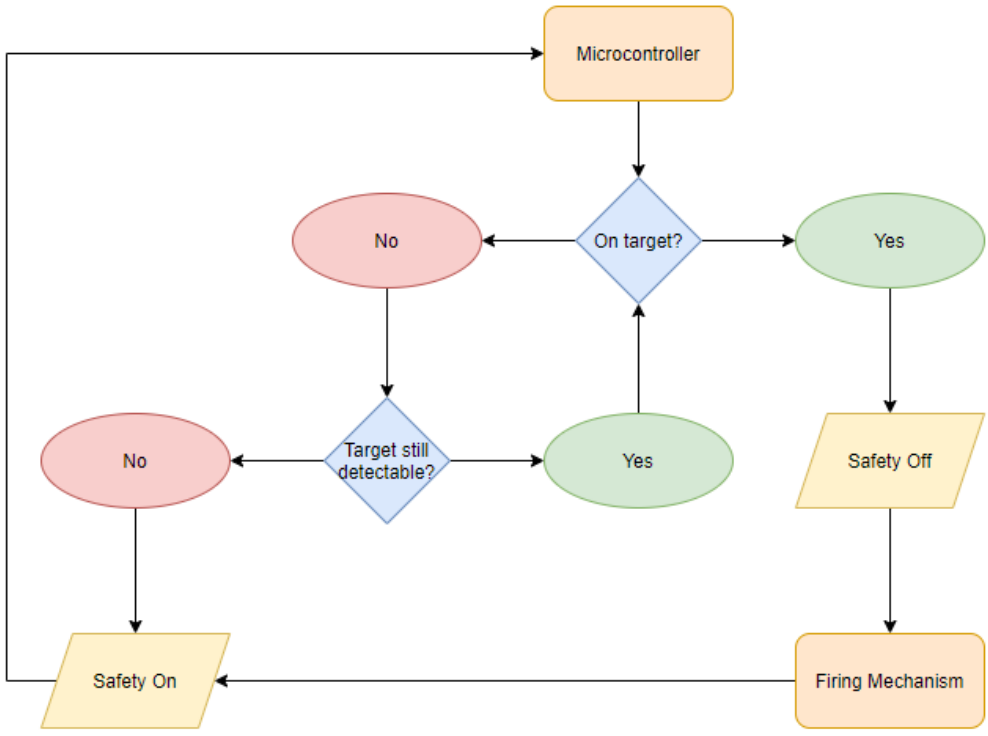


Figure 22: Safety Motor State Diagram for Autonomous Control

The safety motor will operate through the microcontroller by the following checks. The first check is if the turret has locked on to the target. The target acquisition is being handled by the computer vision system, the microcontroller will know when the target is

acquired. If the target is not locked on, then the next check will be if the target is even still detectable by the camera. If the condition is no, then the microcontroller will set the safety on and then exit. If the condition is yes, then we return to the on target check for as long as the target is detectable. Once the target is locked on, the yes condition will have the microcontroller set the safety off and then the firing mechanism conditionals must be met. After the firing mechanism finishes its checks, the safety will be reset back to on and the microcontroller will then continue the necessary operations afterwards.

5.8.2.3 Firing Motor

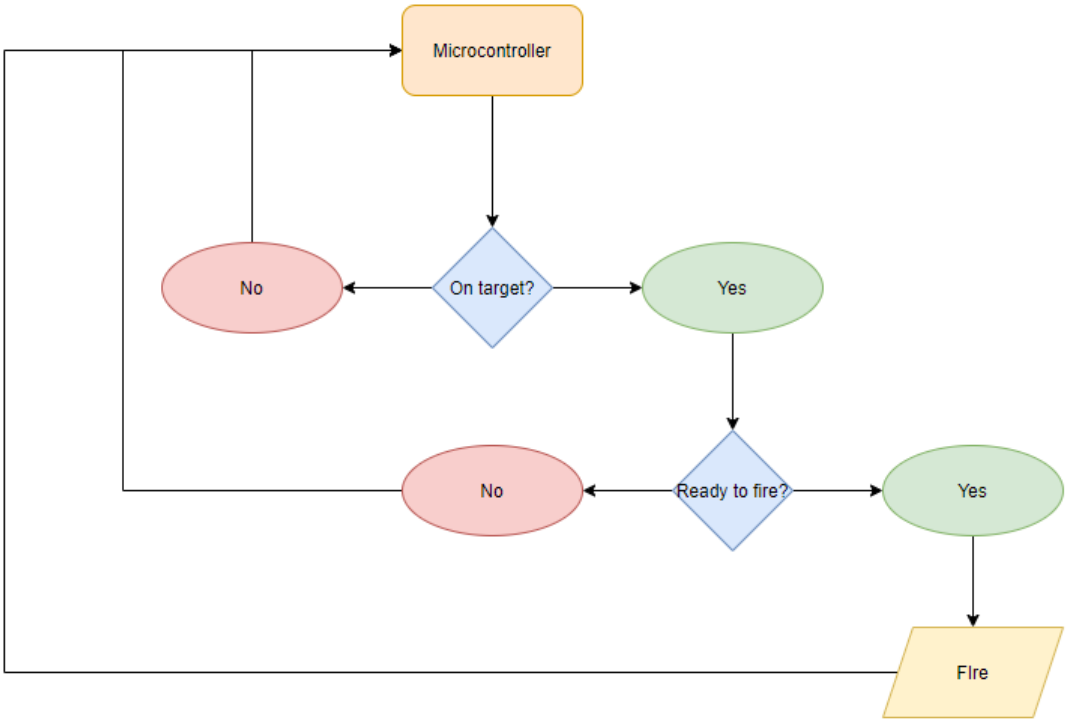


Figure 23: Firing Motor State Diagram

The firing motor state diagram is shown above. The state diagram explains the checks that the program must take into consideration before it begins its operation. The first check is whether the turret is on target. If it is not, the firing motor will wait on the motion motors to move to the target and if the target is no longer in the vicinity then the microcontroller will shut down the entire operation before even the motion motors get their checks in place. If the target still exists and the turret is now in position, the next check is if it is ready to fire. This check is important because the firing motor has to wait on the safety motor to turn the safety off so that the nerf gun can power on it's internal motors and allow for the turret to fire, otherwise the nerf gun will not fire even if the firing motor pushes the trigger. If the turret is ready to fire, which means that not only is

the target in position but the safety is off and the internal motors are running, then the turret is able to fire and will fire the dart at the target.

5.8.3 Distance Sensor Software

The software for the distance sensor not only needs to establish and run the distance sensor it also needs to help trigger the turret to fire. The first step would be to figure out when the sensor will be running in order to save energy and allow the maximum amount of operational time. This is important because if the sensor is constantly scanning if there is not a face present then it will just drain the battery with no real accomplishment. To counteract this we have decided to enable the sensor only when the turret has either picked up a potential target. This is also a crucial point for the motors to be active as they will be used to allow the sensors to stay on route and determine the distance.

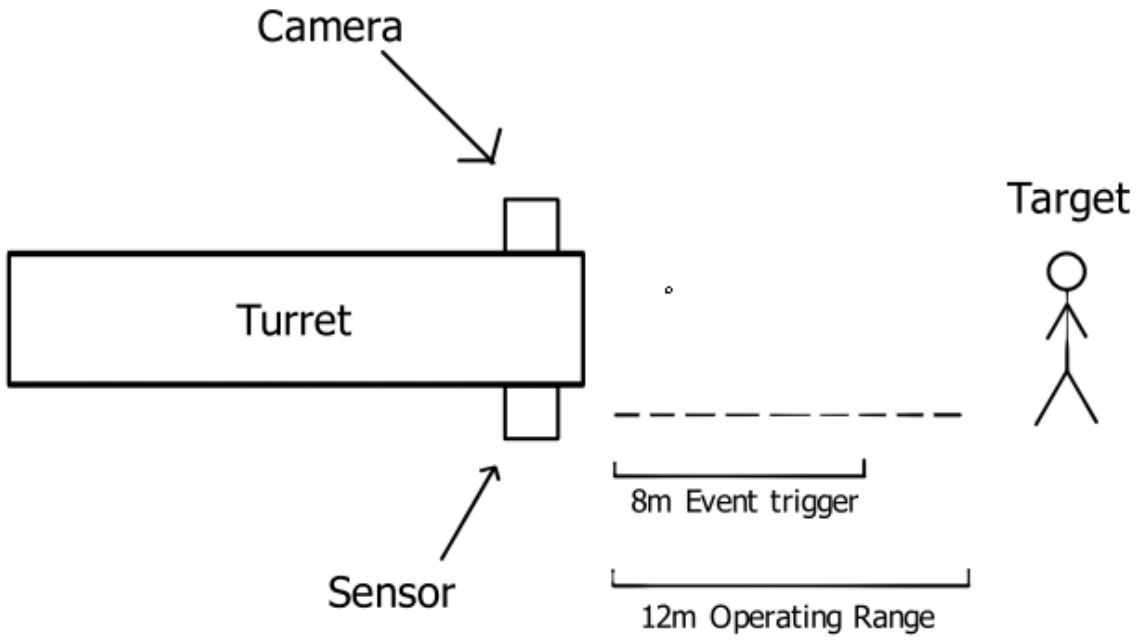


Figure 24: Sensor Trigger Event

The limitations of the sensor is the biggest factor in how we are going to use the sensor and what to do with the information on the distance. Given the limitations of the sensor we have decided to create an event trigger meaning a situation that will cause something to happen in which the turret will start firing. The event trigger was determined by the operational range of the sensor which is a maximum range of 12m. Even with the range being 12m the LIDAR systems are less efficient when it comes to objects that are soft.
In this case the objects of interest will be wearing fabric which will decrease the range so we have decided to decrease the trigger event to start at 8m. Once the camera has recognized the presence of a person it will track them and turn on the sensor. This way

when the person approaches and reaches around 8m it will trigger the turret to start firing, given that the person identified is an actual approved target.

5.8.4 App Design

The app is the first destination for the operation of the turret. It determines the operation mode of the turret which will be sent to the microcontroller. The app is the control center for the project and all of its features will be discussed in the next sections, while how it will operate will be discussed in the App Software Design.

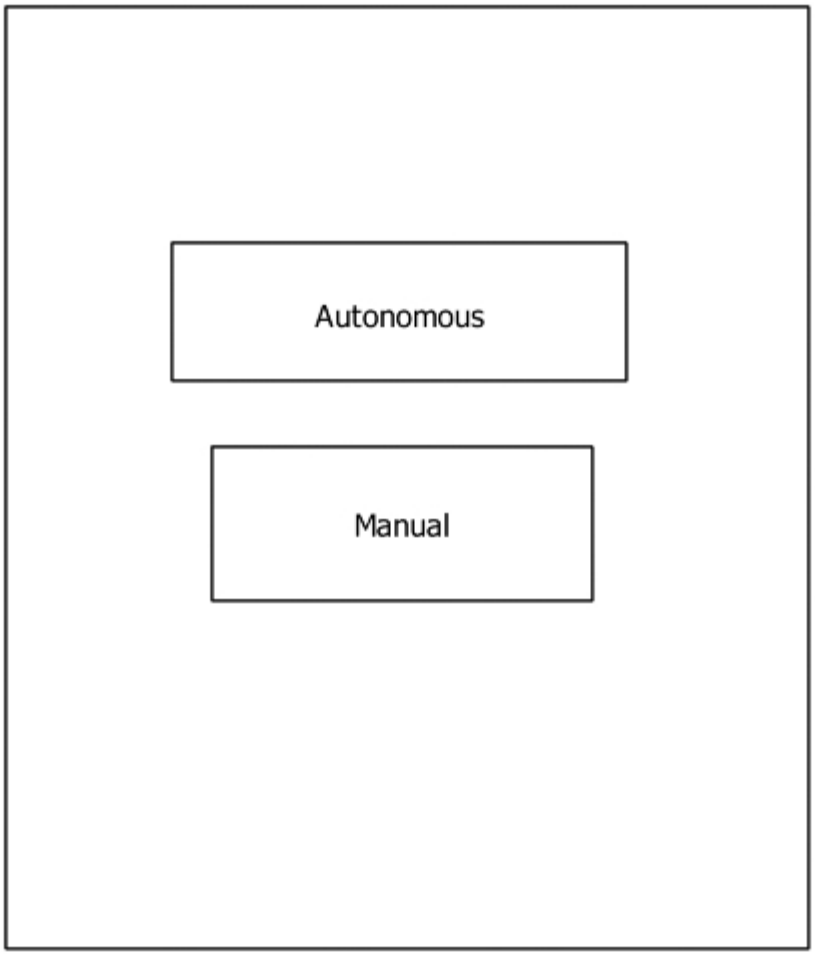


Figure 25: Operation Mode Select

In figure 25, we show the model of the operation mode select page of the android app once the application has connected to the turret through bluetooth. The user will be presented this page in order to determine the operation mode of the turret, since that will be the first thing that must be selected for the specific operation the user will like to use. Either choice will send the app to a different page which will be modeled to perform the duties that the operation select must act on. Since this project is not a final design, we did not require any sign in before this page, however the landing page will be a page that will

have bluetooth connectivity so that the phone or tablet will be able to start the pairing between itself and the microcontroller.

Once that pairing has been completed, then the operation mode select page will present itself to the user and the turret is ready to operate under whichever mode the user decides. The two modes, autonomous and manual, will send the user to their own pages and will have their own requirements to fulfill in order to operate, the Autonomous Mode Operation is shown in figure 25, while the Manual Mode Operation is shown in figure 26. These respective sections will describe their page orientation and describe their functionality.

5.8.4.1 Autonomous Mode

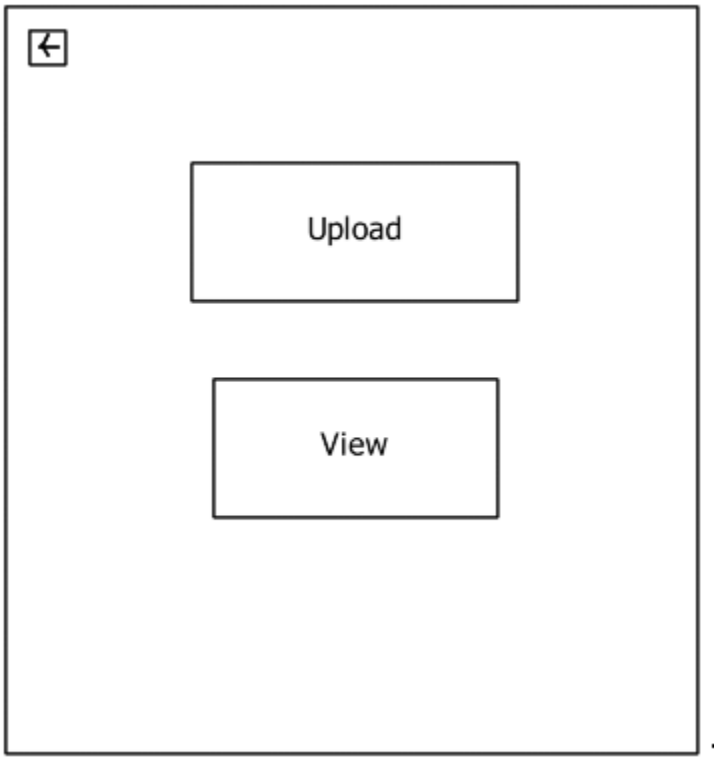


Figure 26:Autonomous Mode Operation Menu

The Autonomous Mode Operation shows three objects in the page. The top left button is the back button, and the necessity of the back button is to signal to the turret that the current operation is over and a new mode will be selected, whether it be the Autonomous Mode again, or if the user wants to use the Manual Mode. The next option is the upload which as the name implies, allows the user to place a new image. This feature is one of the features that was the reason for using the phone and tablet over the joystick controller, and it gives the turret a dynamic approach to finding the target rather than having to continually add an image to the computer vision system directly, instead sending it through bluetooth to the microcontroller and it will share this image to the computer

vision system to find the target the user wants. This operation will be discussed more in the App Software Design.

The view option has two options, the first is added in order to find a list of the targets currently in the system. When the view is pressed, the app will direct the user to the view page, shown in figure 27. The view mode will have the features required to have the autonomous turret function properly. First the app will display the images in a small sized location in which the user will be able to press the image and the app will resize the image so that the user can see it enlarged. These smaller sizes are useful in compacting the images placed to fit enough four images per line.

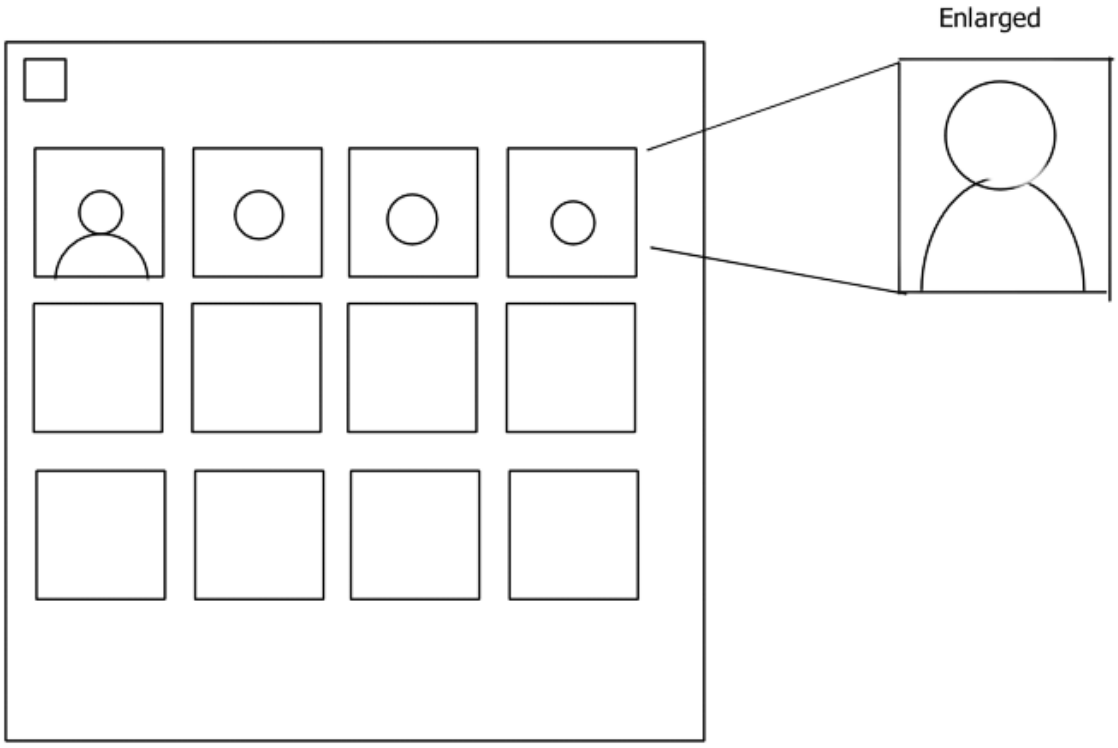


Figure 27: AutonomousMode Operation View

In the enlargement of the image, a few features will be given in order to manage and direct how the autonomous operation will be performed. The first feature would be the deletion of the image. It is an essential function in all database structures, the ability to remove an item once it has been added to the list. This will allow the user to be able to remove someone from the list of targets, whether the reason is that the supposed target has been “neutralized” or if the target is no longer on the target list. The next feature is the target on or off toggle. The reason for this feature would be that if the target is currently not being targeted but there is potential in retargeting that specific individual, we can have the image remain in the database and just determine whether the turret should engage the target or not at the given moment. This allows a dynamic approach for the turret that allows it to disengage when a target is either not a threat anymore or if the user simply does not want the turret to engage with the target for any other reason.

The view page as shown in the figure will have a three by four matrix image size, where these images will fill the page. The three by four will also be the maximum amount of images that will be allowed to be uploaded, in that way we limit the amount of images in the database and have the turret focus on these select images without having to bottleneck the performance of the turret any more than what it will have when looking through the images for the targets and finding whether the images in the database and the image in the camera match.

The second option for the view is shown in figure 28.

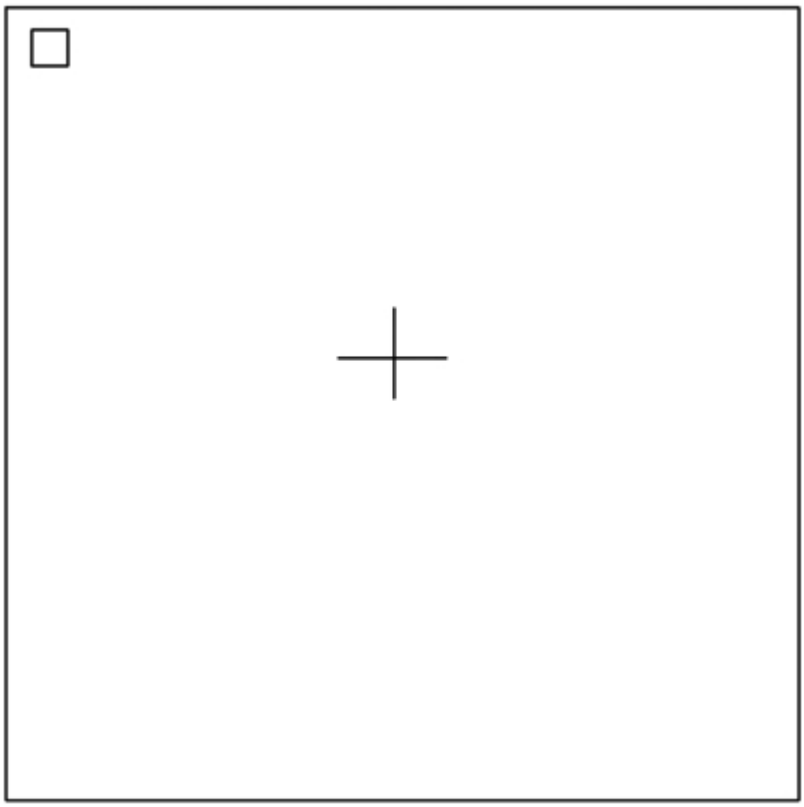


Figure 28: Autonomous Mode Operation Camera View

The Camera View option is the option for the user to have access to the camera view while using the app. This view shows a real time visual of what the camera sees. This way, the user is able to monitor the activities of the turret, as well as see it in operation. This is a valuable feature for examining whether the turret is operating as it should and even to shut it down if the computer vision were to make a mistake a quick exit of the autonomous mode will be asked when pressing the back button on this page. This will provide an interactive emergency plan for anyone who notices a problem or malfunction of the turret, or if the facial recognition mistaken an individual with similar features to the one in the database that is set as a target.

5.8.4.2 Manual Mode Operation

The Manual Mode Operation is selected when the user chooses the manual mode. This operation mode is used to allow the user to control the turret on their own, shutting down the computer vision to focus on the commands of the user to the turret and executing them without autonomy.

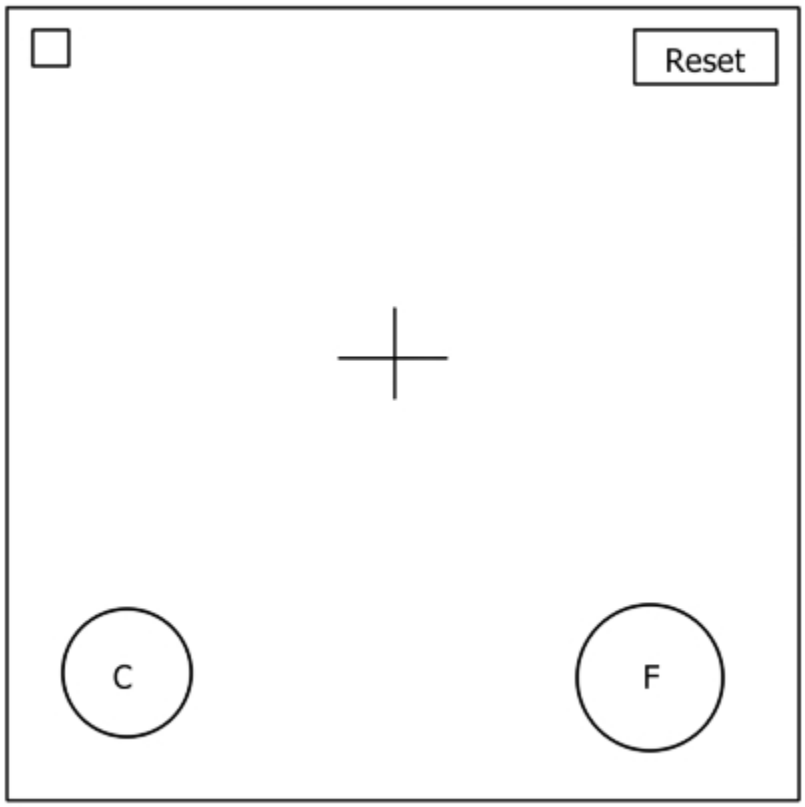


Figure 29: Manual Mode Operation

The manual mode is shown to have four key features, each one relevant to the manual operation of the turret. The first is the familiar back button feature described in the autonomous mode. It will revert the operation mode of the turret to the Operation Mode Select menu and will allow the user to reassign the turret mode of operation. At the bottom left of the figure, the circle with a C represents the control of the motion of the turret. The C stands for Controller, which will be a digital joystick that will translate the user joystick movement into the turret dual axis movement. In the bottom right, the circle with an F, which means Fire, does what the name implies. It is the button that will set the turret to fire the Nerf dart.
The final feature for the manual mode control is the Reset button. The Reset is used as the button to allow the turret to revert to the original position, where the turret is facing parallel to the chassis board and centered, which is called the zero position. The manual mode menu also shows the visual of the turret camera with the crosshair being the center

of the camera view. This feature will allow the user to be able to have manual control without having to be looking at the same location as the turret, but rather they can be in different locations for as long as the bluetooth can support the distance and connection between the itself and the controlling device, either phone or tablet.

5.9 App Software Design

The overall design will be an important factor on how everything is set up. This starts from how each component will communicate.

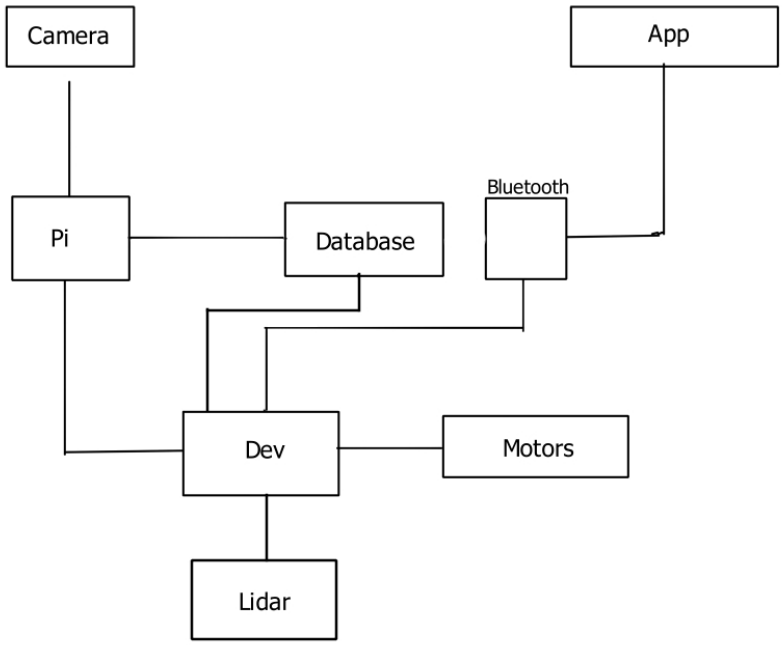


Figure 29: System Connection Process

The connectivity is shown in Figure 29 which is connected by a line connecting a module. The camera is connected to the Pi as the whole image processing will be accomplished by that board. In order to cross reference the images, we have decided to attach a database to the pi as it will have the fastest processing time. Now we are also using bluetooth to communicate the photos between the app and database for image uploading. This would mean that the development board would need access to the database as well since the bluetooth is attached to it. This can be done in two ways one being the accessing through the pi and the other form is by giving direct access to the database. We decided to give direct access as it would be the fastest and would not require the pi to process more information on top of the already heavy image processing.

The next step is the connection from the development board to everything else. Since the bluetooth is attached to the development board it will have a connection to the development board directly but will not be able to do much besides what will be programmed in the app itself. From there the information relayed from the pi will be

transferred to the Dev which will use the information to work the motors and use the information from the LIDAR to activate the turret. The turret is not in the figure due to it being controlled from the motor itself therefore the connection ending at the motor itself.

The next step after defining the component would be to set up the app using that information. As the app will only have access to the bluetooth which is attached to the development board each menu will access different information from each part of the system. These sections include the Database, Camera and motherboard combination, and the flag triggers for the photos.

5.9.1 App Database Connection

This section of the app software design will talk about the connection to the database itself. As previously discussed the app is connected to the development board through the bluetooth connection. Therefore the app itself will not be connected to anything but the board, however it will have access to the other components and the database through this connection alone. The app should be able to view what is available in the database therefore it will send a request to the board through bluetooth similar to a GET command in web development. This connection will also be allowed to do something similar to a POST command because we are also going to be providing the ability to upload the pictures to the database. Therefore the app will send the information to the bluetooth which will be delivered to the development board to be saved directly into the database. This is the same as the delete feature which will take the photos and remove them from the database.

5.9.2 App Manual Connection

The manual connection will need access to the camera. Like the database connection to the app this will also be limited in the app for how the connection is made. This limitation refers to the access for the raspberry pi which will be accessed only through the development board. Therefore, similar to the database connection, it will be connected through the bluetooth connection creating a GET command to retrieve the information from the camera, which will be sent to the raspberry pi which in turn will be sent to the development board which will be sending out the data through the bluetooth module. From there it will be a back and forth communication as the camera feed will be a retrieval and the controller access will be the send commands to control the motors.

5.9.3 App Flag Connection

The app flags are a component of the database connection. This will only be accessed through the automated mode from the view option that we have implemented. Ignoring the upload and delete functions as we discussed in the previous App Database Connection, rather than the database itself we will be discussing what will be a flag in the database itself which will correlate to what the turret will try and target. Therefore the database will have references to whether the picture is flagged or not possibly done through the boolean variables or an integer value of 0 or 1. This will not be a retrieval

request but a POST request which will send the information from the photos selected. This information will primarily be used by the raspberry pi which again will send the information to the development board to control the motors and use the LIDAR as a trigger to shoot.

5.10 Bluetooth Module

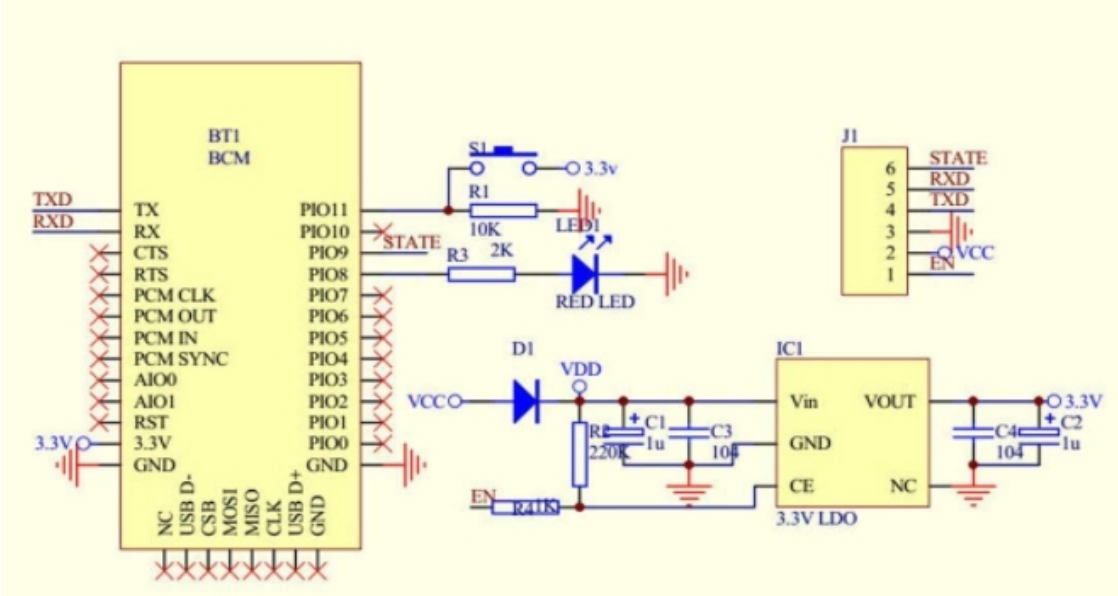


Figure 30: HC-05 Circuit Diagram

The JBtek HC-05 became our pick for the Bluetooth Module. It has the most streamlined setup and startup time. It is a widely used module and has found its way into applications such as home automation, consumer applications, wireless robots, communication between laptops, desktops and mobile phones as well as data logging. It was perfect for connecting to our Arduino Uno and allowing us to establish a connection with Android devices.

Given it has similar operating voltages as competitors as well as a similar effective connection range, it was an easy choice along with the fact it has a decent community support and online documentation to follow. There exist many android applications that are readily available. The 9600 baud rate allows for the ease of interfacing with any USART supported microcontroller. Command mode allows the configuration of default values. Within the two operating modes, one allows the default device settings to be modified, known as AT Command mode. The other is where data can be sent and received from other bluetooth devices, known as Data mode. It uses Serial Communication as its communication method and the master and slave mode can both be switched. By default, upon powering up, the key pin will enter into Data mode but it can be grounded to enter into Command mode.

We are able to connect this to the Arduino Uno and get up to speed rather quickly, running our mobile app for ease of use as a controller. The enable pin is used to set it to Data mode and or AT command mode. The AT command mode is set to high. The VCC pin connects to the 5V VCC pin of the Arduino. The ground pin connects to any of the three ground pins of the Arduino. The TX pin of the HC-05 connects to the RX pin of the Arduino. The RX pin of the HC-05 connects to the TX pin of the Arduino. The State pin is used to check if the bluetooth is working as intended.

5.9.1 Bluetooth Control Interface

The JBtek HC-05 Bluetooth module allows for its interfacing with android mobile devices. It uses serial communication allowing the communication between the module and other electronics. The 2.45GHz frequency band allows the exchange of files between small devices, such as phones, over the short-range wireless connection. The transfer of data can reach a rate of 1Mbps and the range that supports that transfer rate is within 10 meters.

The operations in the Master-Slave mode allows for data to be neither sent or received from an external source. The Command Mode allows you to communicate with the Bluetooth module with AT Commands so that you can take different settings and parameters and configure them. Such configurations to be modified exist like module name, Baud Rate and master/slave modes. The Master or Slave modes can be configured as a communication pair but as a prerequisite, we need enter the proper AT Commands after activating Command Mode. In Data Mode, communication with separate bluetooth devices is made possible, such as data transfers.

5.10 Summary of Design

The BRO-NS autonomous, with manual control override, turret has the major components necessary to operate in both the autonomous and the manual mode. In figure 31, all the components are shown and the arrows each pointing to the component that it provides either power, data, or commands. The app is the instrument that starts the operation off with first, the option of autonomous turret or manual control. If the choice was autonomous, we have the ability to add and delete faces in a database that will be the targets for the turret. It will allow access to the camera footage of its operations and the choice to target which individuals in the image database. The app communicates the choices and sends and receives data through the bluetooth module to and from the main PCB microcontroller.

The microcontroller will send all images in the database to the raspberry pi, which will hold the same database storage of images, and when changed in the app, the microcontroller will send the changes to the pi. The pi will process the visual data from the camera and send it to the microcontroller which will send the video feed to the app to obtain a real time view of what the turret camera sees. The main board will power the LIDAR in order to detect the distance of the target, and send the data back to the microcontroller which will process that data and use it to determine what location offset it

will use in order to fire the nerf dart successfully at the target. The final sections are the operation of the motors. As shown with the dashed lines, each motor is connected one after another. This is to indicate that the first condition for which the other motors will be able to operate is dependent on the previous. The motion motor must complete itself before the safety motor will be engaged and the firing motor cannot fire unless the safety is off which is dependent on the safety motor. Once all the conditions are met, then all motors are engaged and the nerf gun will then fire the nerf dart at the target.

Many of its capabilities have to work in tandem and the software development of this system required impeccable and sophisticated solutions to make each feature as powerful as possible while maintaining a system that is both coherent in its operation, while also being completely adaptable to the user and their needs and desires. From autonomous to manual control, changing of images in the database and having to commit those changes and have them permeate to the pi to keep both devices on the same page. Choosing whether someone in the database is a target or is just there as a precaution and quick reinstatement of threat, and the LIDAR and how it translates the distance to offset the turrets' position to focus on having the best accuracy we can afford in this system.

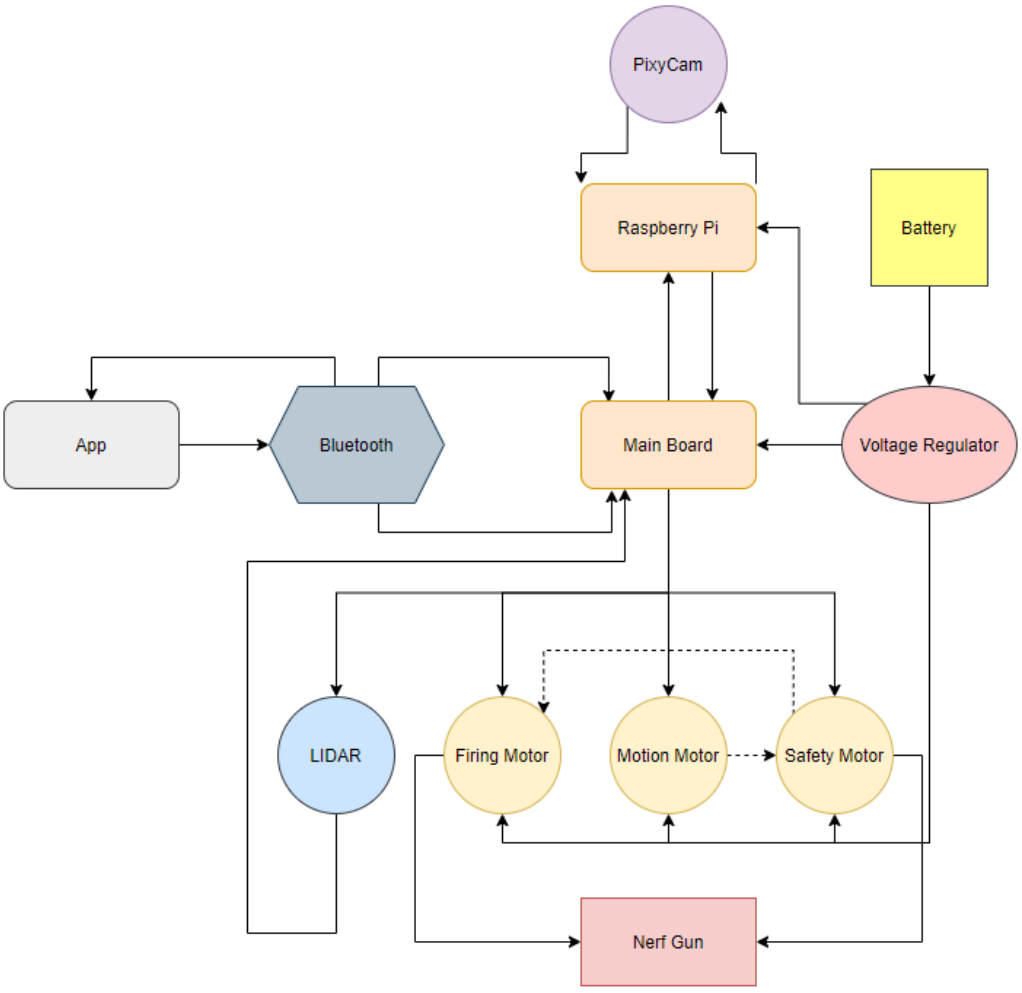


Figure 31: Overall Project Diagram

6. Project Prototype Construction and Coding

6.1 Integrated Schematics

For the custom circuit schematic, all work was done in AutoCAD EAGLE software. We started our PCB design process by first looking at the datasheets associated with the various electrical components we wanted to include in our circuit. The component datasheet gives a lot of information about what connections need to be made, and what values of RLC components need to be used to ensure the circuit functions as intended. Ultra Librarian is another tool that was essential in designing our PCB as it allowed us to import parts that we want to use that are not included in the default EAGLE libraries. Our full PCB schematic is shown in the following figure.

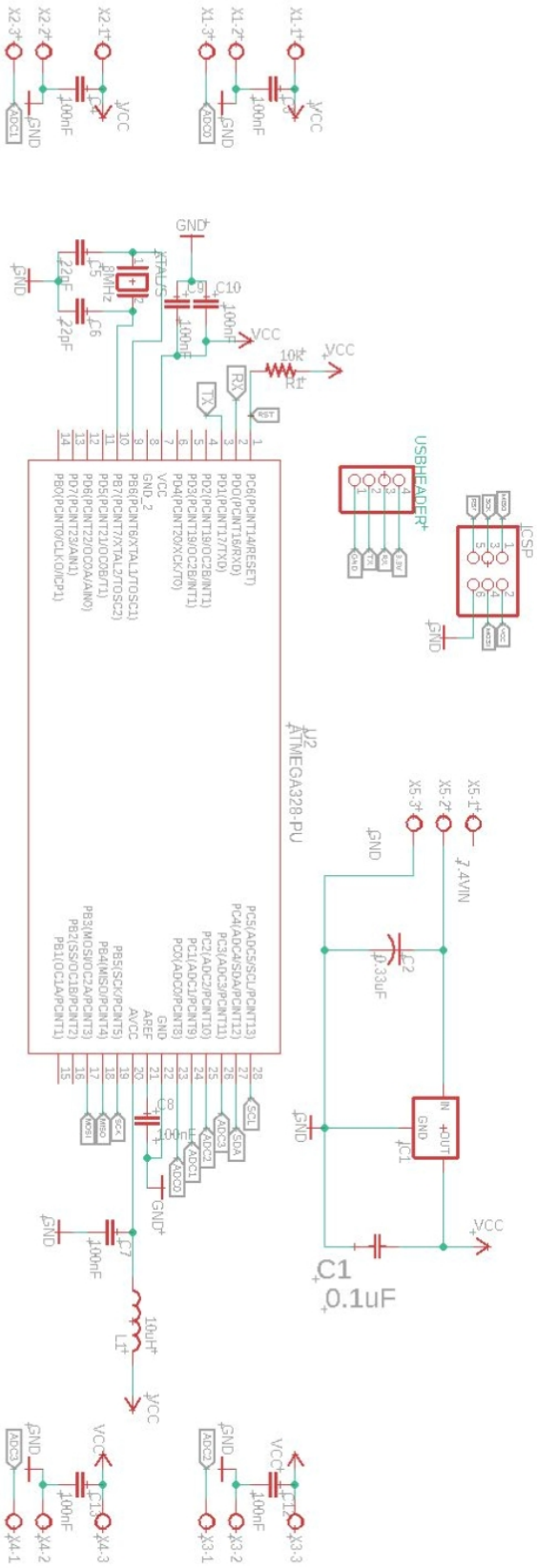


Figure 32: Full PCB Schematic

6.1.1 MCU Schematic

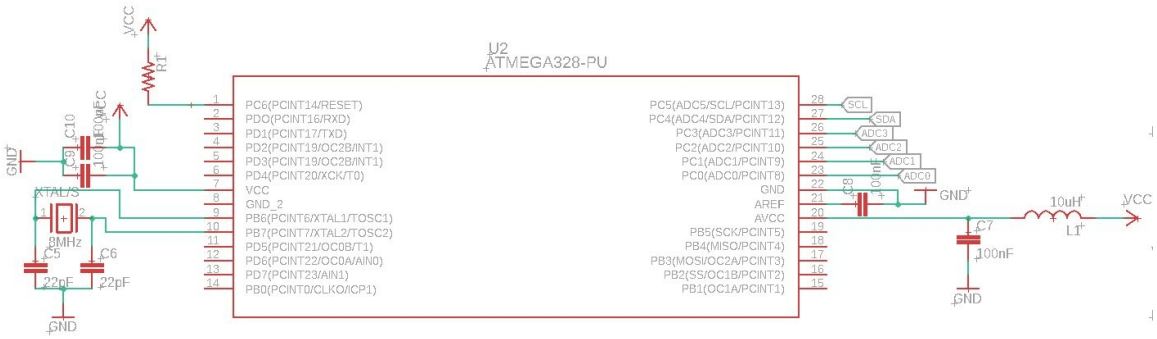


Figure 33: MCU Schematic

Figure 33 shows the MCU schematic and the various components attached to it. From pin 1, there is a pull-up resistor connected that will act as a reset. There are two 100nF capacitors tied to the VCC pin to help stabilize the input voltage. Below that, we have decided to implement an external crystal oscillator as it will control the clock rate of our MCU. The crystal oscillator will run at 8MHz and is connected to the MCUs XTAL pins. There are two 22pF capacitors connected between the oscillator and the ground to help reduce noise and keep the timing device running as accurately as possible. On the right side of the circuit are the various data pins and clock pins. SCL and SDA run to another external crystal oscillator that runs at a lower frequency of 32.768kHz. Pins ADC0-ADC3 are the analog data ports that will allow our sensors to send and receive data from our MCU. In the datasheet of our MCU, it explains that a low-pass filter needs to be connected to the AVCC pin on the MCU. It also explained that the low-pass filter could be implemented with a 10uH inductor and a 100nF capacitor connected to VCC.

6.1.3 Voltage Regulator

There were several different design decisions that had to be made when selecting a voltage control regulator for our custom PCB. Initially we tried finding a power source that would output the desired voltage we would need for our circuit components, that way we could omit the voltage regulator all together simplifying our integrated circuit schematic. If we could find a voltage source that would supply voltages within the allowed range of voltage inputs for our components, then we would not need to step down or step up the voltage to the various components on our board. Ultimately we decided not to do this because we would still have to include components such as bypass capacitors in order to help supply a constant reliable voltage level. The next issue we faced was finding a voltage regulator that would supply both the voltage and current that would be required by our motors and microcontroller. Many of the voltage regulators we looked at that could supply between 3.3-5.0V were designed for low current outputs and were limited to between 1-1.5A. After some more research we were able to find an adjustable voltage regulator that could supply up to 5A of current.

The voltage regulator we decided to implement in our project is the TI LM1084 5V voltage regulator. The main electrical components that we need to supply power to are the four motors we are using to aim our turret, and the microcontroller we are using. The input voltage range for our microcontroller is 2.7V-5.5V and our motors operate between 4.8V-6V. The LM1084 is rated for an input voltage range of 6.5V-35V and has a rated output of 5A with an adjustable voltage output between 1.2V-15V. To alter the output voltage of the regulator, the LM1084 datasheet explained that the values of two resistors tied to the output of the regulator needed to be altered. The values of the resistors were given in a table in the datasheet along with their corresponding output voltage. The datasheet also suggested using a tantalum capacitor tied between the output and the ground to act as a bypass capacitor and help the regulator be more efficient and have increased stability. The datasheet recommended using a tantalum capacitor as it is more efficient than most of the cheaper aluminum capacitors. Higher quality, high efficiency aluminum capacitors could be used but the tantalum capacitor was less expensive.

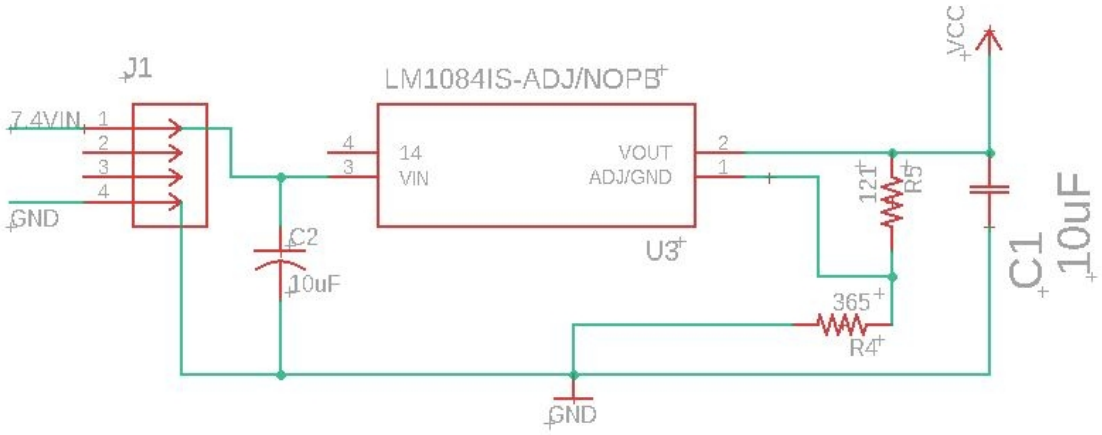


Figure 35: LM1084 Voltage regulator

From *figure 35*, the capacitor C1 is the tantalum bypass capacitor, C2 is the storage capacitor, and resistors R4 is responsible for adjusting the output voltage of the regulator. The connector J1 is a connection adapter that will allow our XT60 RC battery connector to connect to our PCB.

6.1.4 PCB Schematic

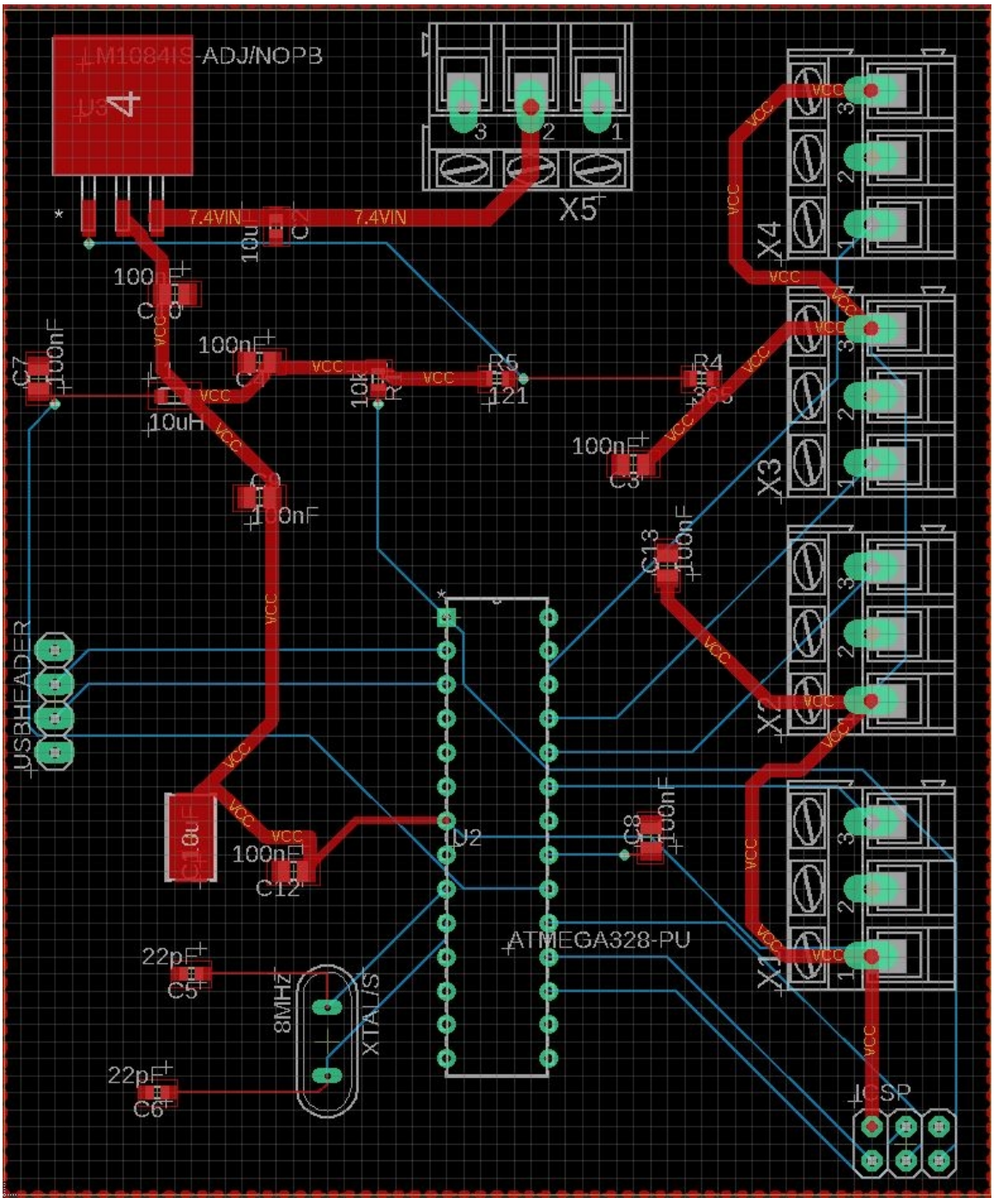


Figure 36: PCB Board Schematic

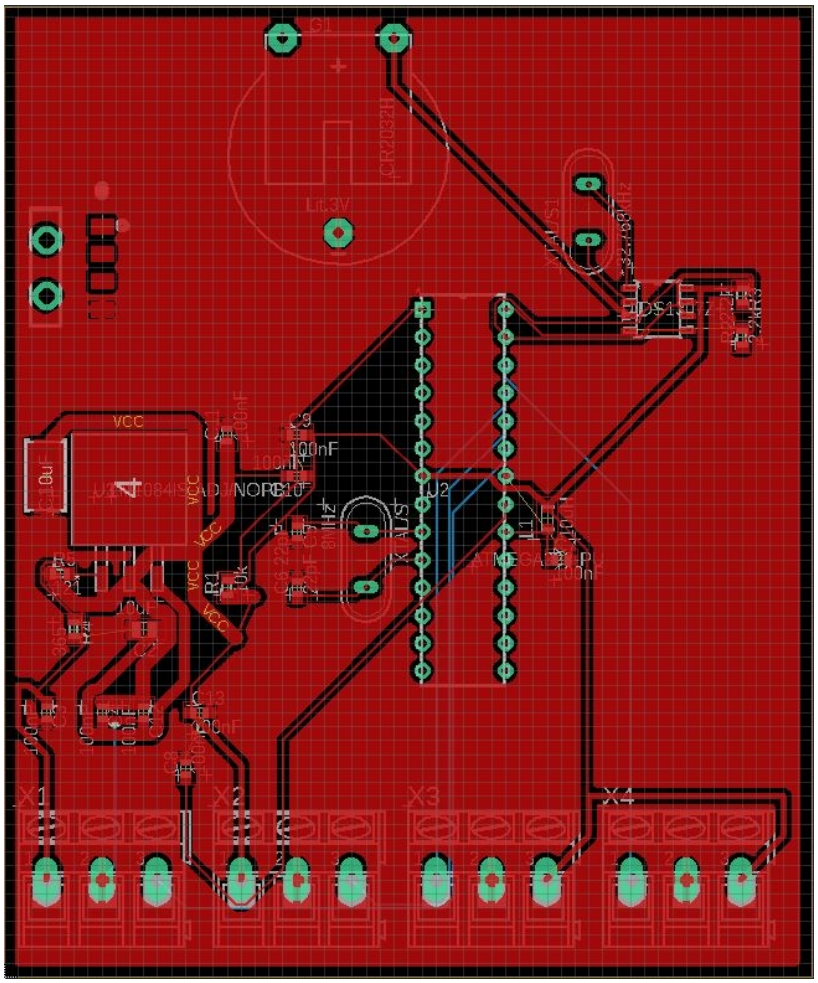


Figure 37: PCB Ground Plane

The first figure shows the final manufacturable PCB design after placing all of the components and traces on the circuit board. Our PCB measures 2.9”Lx3.5”W. When placing the components we wanted to make sure we had as little wasted space as possible because the PCB board manufacturers charge by the area size of the board. Critical components such as the MCU and the two separate timing modules were placed first. It is crucial that the two crystal oscillators are mounted close to their respective devices as longer traces can degrade the accuracy of the timers and impact the performance of our controllers. Next we wanted to ensure that our analog sensor connections would be mounted on the side of the board so that we could easily connect our motors and other sensors without having them interfere with some of the other circuit components. We wanted to try to mount our voltage input and our voltage regulator away from some of our more sensitive components to avoid any unnecessary noise.

The second figure shows the ground plane implemented on our PCB. Adding a ground plane allows all of our grounded connections to be grounded to the same point, this simplifies the overall design and increases the stability of our board.

When placing the traces on our PCB, we used a trace width calculator to make sure that our motors would be getting enough current without hurting the circuit. We ended up using 20mil trace widths to minimize the risk of potentially damaging our board.

6.2 PCB Vendor and Assembly

When researching PCB vendors and parts suppliers, we did not know if we wanted to try to assemble our PCB ourselves or if we were going to pay a company to assemble and solder the parts for us. If we decided to solder all of the parts ourselves, we would still have to pay a company to manufacture the actual PCB using the Gerber files exported from our PCB schematic and board design. Of the manufacturing companies we researched, OSHpark was the best option we could find. Oshpark has a very simple interface where the Gerber files or the entire EAGLE file can be uploaded to their website and they are able to give you an instant quote for the cost of the PCB manufacturing.

All PCB manufacturers we looked at charge by the square area of your PCB board, as well as its complexity, with boards with many VIAs and layers being more expensive. To lower the total cost of our board as much as possible, we looked at condensing the size of our board and eliminating as much unnecessary space as possible so that our overall board area would not be as large. Our PCB ended up being 2.9”x3.5”, and there is still enough room that the parts are not overly crowded. OSHpark’s quote for our board design was to build us three PCBs for $50.41. We would still be required to purchase all of our parts ourselves and we would have to spend considerable time soldering all of our components. The biggest issue we would face with assembling the board ourselves is that none of us have a lot of experience using a soldering gun and we are using parts that are incredibly small.

Because of our lack of experience with soldering we also researched PCB assembly companies. The cheaper companies we found were going to charge upwards of $250 per board. Seeing as we wanted to try to get two of our boards assembled at one time, this option was going to cost us roughly ten times as much as buying the boards and assembling it ourselves.

6.3 chassis Design

The chassis design was built using the materials available to us, we decided to use plywood and dowel struts to construct our chassis as they are affordable and lightweight.

The side view shows the general layout of the chassis board with the microcontrollers on opposite ends and the servo motor in the center to provide the rotation to the chassis board. The board is split into two parts that are clearly shown and divided into top and bottom. The bottom is the platform that will hold both the raspberry pi, the Atmega 328 and the horizontal motion motor. The top layer is a housing part that will shield the motor and the microcontrollers from either bad weather and provide a platform for the nerf gun

and the armatures that will hold the nerf gun in place. Due to this the top portion was designed to be able to withstand the weight of the armature, the nerf gun, and the motors that are being utilized or attached to the armature or nerf gun as a whole and not break. This is especially important because this component is the only thing that will divide the nerf gun and armature from the microcontroller and if it fails, the nerf gun and the attached motors can fall down on the microcontrollers and horizontal motors and damage it. The top and bottom parts of the chassis board are screwed together tightly so that it will also maintain steady, especially as the nerf gun and armature is being moved horizontally and vertically. Failure to do so will fling the nerf gun and the components inside of itself around and will certainly cause heavy damages to the components involved.

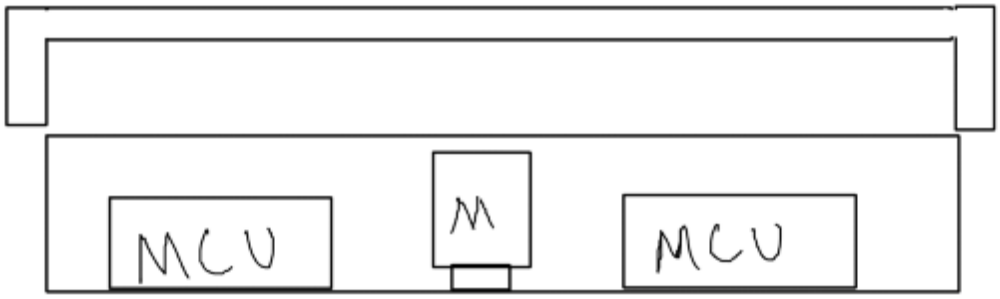


Figure 38: chassis Board Side View

Next is the top view. This is to give the perspective of view when looking down on the chassis board's bottom part. The motor was placed in the center to provide the best possible rotation evenly throughout the turret and the two microcontrollers are at opposite ends.

However, what the concept design does not show is that the chassis bottom surface must hold the microcontrollers and the motor in place so that they will not be sliding around while the turret is moved. The surface will fasten the motor in the center and is attached to the top portion of the chassis board to rotate it, and as such the motor is fixed in place, however the microcontrollers have no connection and as such we decided to compartmentalize these components to have separate sections on the chassis board. This is done by adding holes on the chassis board as well as on the custom pcb microcontroller so that we can screw the microcontroller and the chassis board together. The same has been done with the raspberry pi 4, which already has holes so that we can connect them either with the appropriate screws or small rods that will go through the holes and keep the component in place. The rods are tall enough to reach the top part of the chassis board to keep it steady and will have nuts in order to keep the microcontrollers from moving

from the surface upwards, maintaining in contact with the surface of the bottom part of the chassis board.

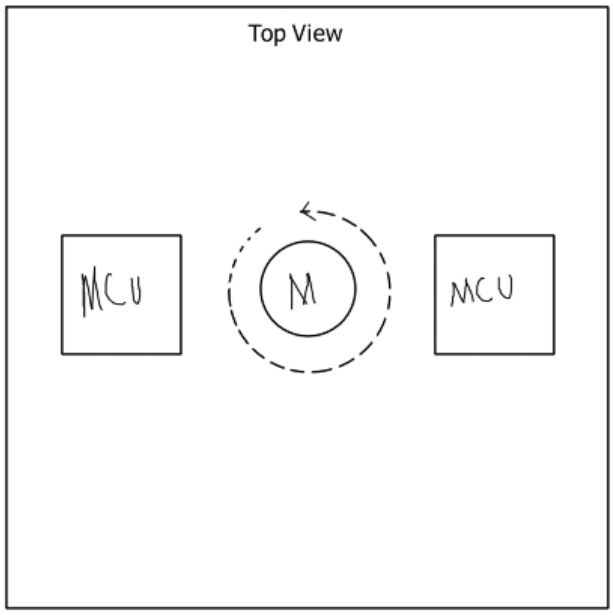


Figure 39: chassis Board Top View

6.4 Armature Design

The armature is the component that will hold the nerf gun in place and allow for the rotation of the nerf gun. The armature is important because it gives flexibility in the vertical rotation than if the nerf gun would be attached directly onto the chassis board.

The figure 40 demonstrates a concept design of the armature that will hold the nerf gun in place and maintain its connection to the chassis board. In order to rotate in a vertical fashion, we set the vertical motor above the top part of the chassis board which will be fastened into place and connected to the base of the armature. The base of the armature will be designed to be able to hold its place but also rotate when the vertical motion motor moves it. The way it will work is there will be static walls right next to the armature base that will be hold the armature in place and the armature arm itself will be connected via rods that will keep the armature and the wall connect but allow the motor to move the armature in a vertical axis. The figure shows the turret as the T, the armatures as Pi, the rotating platform as RP, and the platform, which is the chassis board, as P. The vertical motion motor will be fastened on the center of the rotating platform. The rotation of the turret will be the rotating platform will move the top section and the vertical motor will move the armature up and down, which will translate to the nerf gun move down or up at an angel that will move its range of firing to the designated location chosen by the user or through the autonomous system.

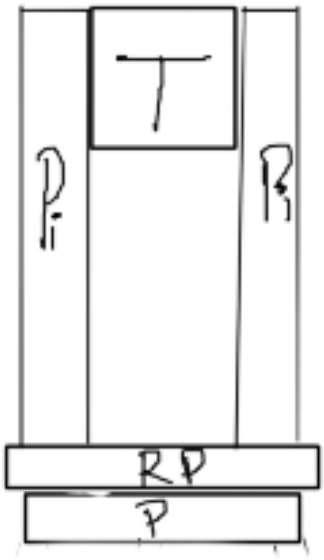


Figure 40: Nerf Gun Holder

6.5 Final Coding Plan

The overall coding plan will have to take into account the overall system. The system includes the bluetooth, remote control of the turret, an app to access the control of the turret, accessing the camera to see who is coming and comparing the database to see if the person is actually a target, and the firing event using the LIDAR to trigger the firing mechanism. As we have already discussed the plan for how the code will work, we will be going over the communication of all the items together rather than explaining how each option will work by themselves, and what they do.

The code will essentially have two separate working parts that will work independently of each other. The first part is the remote access part which will be controlled through a wireless device which we have decided to be a phone, and the other part is the automation which will operate on its own according to the parameters that we have provided.

6.5.1 Automation Coding Plan

The automation part of the code will utilize all the components to validate and cause the trigger event of shooting at them. These components include the distance measuring sensor, the camera, facial recognition and identification software (including raspberry pi programming), motor operation, and the atmega328 for using the rest of the components. The primary aspect of the automation is the identification of the face as the turret will not work by itself without it. The secondary aspect is the facial identification process which

will be the trigger to start the distance processing event. The next part of this would be the trigger event once the target reaches a certain distance that is less than the operating range of 12m. These components are the main components that work with the automation part of this but without the support of the motors they will probably not hit the mark and so the camera will be supported by motors to keep the turret on track with the target. The automation coding plan will start with the app design and functionality. While the app is being developed we also needed to have the main board microcontroller be designed and built in order to also begin programming the board and designing how the microcontroller and the app will communicate. It is because of this that it is essential to have the bluetooth module that will be paired to the app and will be directly connected to the main board microcontroller to program these two devices to connect to one another. To do this pairing we must first have an app that has the main menu with at least the choice to transition from that main menu to the automation page, then have the automation page designed and then program and modify the upload an image to the microcontroller. Then we connected the microcontroller to the computer and figured out how to find the image file that was saved in it and display it in order to be determined that this works. Afterwards, we began with the development of the database in the app and also in the raspberry pi, since they must match. For this to happen, we began programming the pi to have the communication necessary with the main board that will allow us to send images from the main board to the raspberry pi, which we then connect to the computer and look to see if the images match and are of the same quality. The database will then be established and we programmed the communication between the raspberry pi and the android app through the bluetooth and the main board microcontroller, and make sure that this series of connections are working. Then we test whether the database changes will also change the database in the raspberry pi in the same way. Now that the database is finished, we can then begin with the raspberry pi and the pixy cams' facial recognition by using the database that was just built. We had to look into many cases that will cause problems such as what happens when there are two faces that are in the database, as well as look into a priority queue which will determine that if there are similar faces that the camera detects in the system, it will prioritize the highest priority target. The communications between these components will be desired around the third week into senior design 2 so that we may be ahead of time since this portion. Then we looked into facial recognition programming and tried out different methods and tested which method will ensure the best results with finding the right target, choosing the highest priority target first, possibly by using priority queues or other techniques.

Once this has been developed, we incorporated the microcontrollers and the mechanical components together to form the turret, only then can we start developing the communication between the facial recognition and the motors and how they will be able to move the turret to the position of the target and track them, which will lead us to developing the motor software in the main board to have the checks for when it is a good time to fire. This will require quite a bit of time as we have to program the firing of the turret with cases such as what happens when the target moves at the exact time the dart is fired. Is the software fast enough to determine movement and cause the stalling of the firing or will the dart fired be fast enough to where even if the target moves at the moment of firing, the dart will still hit them. These questions that are critical to the

performance of the turret will have to be answered then and a solution will be needed as soon as we reach this point. After this, we made the finishing touches on whatever else is needed for the autonomous control, such as using the distance sensor to find out how far the target is and add any aiming corrections, and if there are any more useful additions or upgrades that will make the turret better. Once all these critical components are finished we then began with the next mode of operation, the remote control.

6.5.2 Remote Control Coding Plan

The second part of the code is the remote access and control. This remote access will be possible through the bluetooth module that we plan on using to control the turret from a range. Unlike the automation coding plan which relies on detecting the presence and determining when to shoot by itself, this remote access will be reliant on the user of the app. The app will provide a UI where the user can access the camera's view and follow the individuals through the app's controls. From the app the user is also able to fire the turret through the press of a UI button which will in turn activate the motor to press the button to activate the turret and then to start firing.

Due to the programming of the autonomous mode being completed in a working form, we can then begin on the manual control due to how simpler this mode is. Essentially, we first had to design and program the manual page and construct the UI with the digital joystick, a fire button, and the display to look at the camera in real time. Due to real time display already being incorporated in the autonomous mode programming, this will just be added easily. The most difficult and longest portion would be the design of the controlling of the motors manually due to having to determine the specifications of how this will work. The joystick will be used so that the user can control the motors in a two dimensional format with the up and down on the joystick being the vertical and the left and right being horizontal and anything in between will be moved according to its direction. This will require the most testing and the longest portion of the remote control which is why we were working on this portion in the beginning. Once all the UI is designed and can communicate with the microcontroller. Once that is completed, the rest is just fixes and upgrades to the program for whenever we can, possibly add additional features that could incorporate the LIDAR sensor if desired.

6.5.3 Low Power Modes Plan

One of the most critical programming designs has been programmed during the critical designs that create all the operations of the turret, and that program is the low power modes. The low power mode will allow the turret to last longer without the need to recharge because it will shut down all the unnecessary devices that are not being used and will send it into sleep mode. This will take quite a bit of time added onto the plan due to the need of shutting down the peripherals and as such we must be careful with programming them as we do not want a low power mode when the program is in a critical operation. This is the reason why we were working in parallel with the code for

the operation of the turret and the low power modes that will switch the peripherals and microcontrollers on and off depending on the usage.

7. Project Prototype Testing Plan

7.1 Hardware Specific Testing

7.1.1 Projectile Launcher - Nerf Gun

We performed the Nerf gun testing so that we would have a better idea of what the realistic operational range of our turret was going to be. It was important that we find the maximum effective range of the foam projectiles, what the relative accuracy is over various ranges, how fast the projectiles travel when being fired, and how much bullet drop should be accounted for when aiming our turret.

For all of the tests with the exception of the 10m range test and the maximum distance test, the Nerf gun was positioned at a set height and fired from a fixed position to minimize the discrepancies between the different tests. The tests were also performed indoors to minimize the effect of wind and other environmental factors. The specific tests that were performed are as follows; maximum range test, projectile velocity, accuracy at 5m firing approximately one dart a second, accuracy at 5m firing approximately three darts a second, accuracy at 10m firing one dart a second, accuracy at 10m firing three darts a second, and bullet drop at 5m and 10m intervals.

For the projectile velocity test, the Nerf gun was fired at a target 10m away. A stopwatch was used to approximate the time of flight over the 10m distance; when the trigger was pulled releasing a projectile, the stopwatch was started, when the projectile hit the target the timer was stopped. While this is a very rudimentary way of testing the speed of the projectiles, we were able to achieve five separate results all within a few hundredths of a second between each other, with three of the results within 0.01seconds of each other. When the five measurements were averaged, the time of flight was approximately 0.32 seconds for the dart to travel 10m. This gives us a projectile velocity of around 30m/s.

For the maximum projectile range testing, the Nerf gun was held approximately four feet off of the ground with the barrel of the gun parallel to the floor so that no launch angle was added to the projectile. Five darts were then fired and their distances were measured from the end of the barrel to the spot at which they landed. After averaging the results together, we achieved an approximate maximum range of 12.2m.

For the bullet drop tests, the gun was positioned at a set height at a set distance away from a wall with a tape measurer measuring the height. When the projectile would hit the wall, the height at which it struck the wall was recorded. Those heights were then averages and subtracted from the fixed height of the barrel, giving us the average drop of the projectile over the set distance. This test was performed at 5m and 10m, where five darts were fired at both distances.

The results for the 5m test are shown below.

Barrel Height	Dart 1	Dart 2	Dart 3	Dart 4	Dart 5	Average	Bullet Drop
25”	12”	17”	11”	17”	9”	13.2”	11.8”

Table 12: Bullet Drop Data at 5m from target

The results for the 10m test are shown below.

Barrel Height	Dart 1	Dart 2	Dart 3	Dart 4	Dart 5	Average	Bullet Drop
42”	7.5”	14”	9.5”	26”	14”	14.2”	27.8”

Table 13: Bullet Drop Data at 10m from target

For accuracy testing the entire ten round magazine was fired at a human sized target at 5m away, and then at 10m away. This test was performed three times at each range and the percentage of hits to total darts fired was calculated. The darts were then fired at a faster rate to see if there was any influence on the accuracy of the gun.

	Magazine 1	Magazine 2	Magazine 3
5 Meters: 1 dart/sec	8/10	9/10	10/10
5 Meters: 3 darts/sec	10/10	8/10	10/10
10 Meters: 1 dart/sec	10/10	6/10	2/10
10 Meters: 3 darts/sec	8/10	6/10	5/10

Table 14*: Accuracy Table*

The Nerf gun was very accurate at a range of 5m, but the accuracy dropped off dramatically by 10m. One thing to note was that with the gun aiming dead center of the human sized target, almost all of the darts that missed, missed to the right side of the target. We expect this is due to a slight rotational force that is being exerted on the dart by the two roller motors that are used to propel the dart out of the barrel. This discrepancy can easily be accounted for by aiming slightly left of the target.

7.1.2 MG995R Servo

The MG995R servo motor has been tested to determine if the motor has the torque needed for the operation of the movements. This testing required not only the weight of the nerf gun but also of the other motors and the mechanical components such as the chassis board and the armatures. This means that we had to test the torque of this motor with weight that is an approximation to what we believed would be the total weight the motor will need in order to perform its duties. We needed to make sure that the torque does not reach the stall torque because the stall torque is the maximum torque that the motor can hold while being still, therefore we must ensure that the torque for this motor is enough to move the platforms for both the platform surface of the chassis board and the rotating base component of the armatures.

The plan to achieve these outcomes is to first test whether the motor is able to rotate the nerf gun on its own. To do this we had to make a setup that would hold the nerf gun in place, fastened so that it will not move, then have the motor spin the surface the nerf gun is placed in and see if it moves. Then, the next testing was performed after the mechanical design of the chassis and armature were done and completed. We then once again tested it but in an actual complete turret design test and saw if the motor worked under that topology.

The vertical motion motor testing was similar to the horizontal, however, since they don't require specifications or intended roles we were able to do less intensive testing due to the vertical motion motor only needing to rotate the armature base which holds the arms that are attached to the nerf gun. This means that unlike the horizontal motion motor which needs to be able to rotate the entire rotating platform that holds most of the turret together, the vertical motor will only need to worry about the weight of the armature components and the nerf gun. However, the motors will still be testing in the same way, first with a model and temporary solution in order to get an approximation of how well the motor works with respect to weight and then the actual testing of the motor with a finished turret design to see if that approximation was correct.

7.2.5 chassis Board Testing

The chassis board required testing as well, especially due to the fact that these components were designed and built from the ground up for this specific project. This means that we had to design it with the specifications that will be the requirements to maintain a smooth and sturdy turret system. The chassis is built in two parts, the surface platform and the top platform. The surface platform will have to be tested to be able to hold onto the components that will be inside of it, the two microcontrollers and the horizontal motion motor. It should also be able to contain the rotation of the motor and hold it in place, which means we had to test the chassis with the motor itself to see if the chassis surface platform moves as the motor,fastened to the chassis, rotates an object. There may be some rotation if the chassis was just laying on the floor while allowing the motor to spin an object, which is why for this test we also fastened the chassis board as it will be fastened with the stand and the top platform together. The next part that was

required to be tested were the compartments that hold the microcontrollers in place. This part is vital as the microcontrollers must be able to maintain their position and in no way move around since it can damage them, either the board itself or the connections to it.

The test itself was put together with the chassis board and its housing with a test case model of a microcontroller and moving the chassis board around in high speeds to see if the model can be fastened and maintain its position without any damages when moved around in fast speeds, or even placed upside down. The top portion of the chassis board will also need to be able to withstand bad weather conditions that aren't severe such as a hurricane, but more of raining and windy. To test this, we chose a rainy or windy and tested it out naturally. This test also used model microcontrollers so as to not accidentally damage the constructed devices and having to rebuild and pay for them again. The final test for the chassis board is the ability to rotate horizontally. This test requires that the chassis board be built and have the motor directly interface with the board and start moving it around. The test must also be simulated with weights so that it can properly represent the actual operation of the turret with the nerf gun, motors, and armature situated and fastened at the top of the chassis board.

7.2.6 Stand Testing

The stand needed testing to make sure that after being attached to the chassis board, that it was able to support and keep it off the ground. We needed to test its strength against the weight of the nerf gun, chassis board, and the microcontrollers along with their peripherals. The stand in its shape of a pole will need to be tested to see of the height and width we create in the preliminary model will be capable of maintaining the balance and weight for all of the components along with other aspects such as the turret operation, containing different directional movements as well as firing of the nerf turret. It needs to be able to retain a reliable connection to the chassis board as well.

7.2 Software Specific Testing

7.2.1 PixyCam Testing

We performed the Pixycam testing so that we could establish an idea of how well the camera operates and the operational range and its limitations. It's important to have these ranges determined as it would help us have a proper test to see how well it works.

Now the tests were done in a closed location without optimal sunlight which did affect the results of the testing. But we did find out a couple of things with one of these including how to add a color to the system to track, and the second being the actual operational capacity and things that may hinder it. As these tests were done indoors it did hinder the device due to lighting location and the overall dark area caused colors to be somewhat similar when shadows were cast on it. This may have also been a factor when the range of identification is involved.

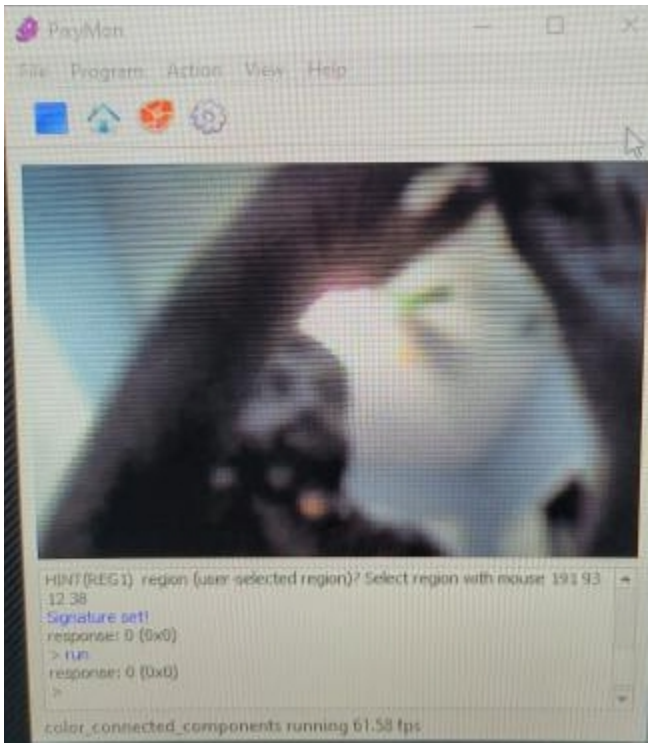


Figure 41: Pixycam testing

The testing involved specifying a color through the software (also able to set through hardware by pressing the button on the camera and holding), this can be done by going to the “Action” tab and selecting the one of the available signatures or overwriting a previously set one. From there the next step would be to select the color which is done by going into the camera view section of the app and clicking and dragging on the desired location. Once the color has been chosen and saved you can click the blue play button (now blue square in Figure 41) to let the camera resume its feed.

Now that we have covered how the colors are able to be set we can look into how it performed. As shown in Figure x the overall quality of the camera feed was not optimal but this may be due to the lighting situation. The light from the ceiling fan is behind the object being targeted therefore that object is casting a shadow having an overall dark appearance making it almost unable to distinguish color as they appear dark. However thanks to the 60 fps of the camera it was able to easily track the color and give feedback in the software (PixyMon). The software will display a box around the color and move it around as the object itself moves in the camera's view. The downside of this was the shadow being cast from the light source but if a light colored object is specified it makes it easier for the camera to distinguish between the others. Therefore instead of having one big box or multiple small boxes there is one small box around the item that we were tracking.

7.2.2 Motor Software Testing

We performed testing with a motor provided by a small kit to test out the different ways that we could utilize them. One aspect of the motors that we needed to determine was defining the rotation speed, having a pause one the initial rotation degree has been reached and whether we are able to define how much it should rotate.

From the overall testing we were able to determine and decide how we were going to utilize the motors. We were able to control the motors rotation speed by adding a small delay in between each cycle of rotation until the point was reached in which case we could have kept it going into an infinite loop. However we decided to test whether we could simulate a trigger push and release. This showed incitement to an issue that may arise in the programming or mounting of the motors. Each time the motor is turned off or given some circumstance the calibration is changed to the point where the motor will either not go to the specified degree or that specified degree is too close to the starting point of the motor which causes it to push into an immovable object. This can cause other issues as the turret may get damaged from long term use of this. In order to fix this we would have to either recalibrate the motor each time the turret turns on or make sure it shuts down in the right position once it is properly calibrated.

The other test that we did was the degree of rotation. This would help us determine how much to go and plays a major role in the calibration end of the turret trigger. In order to simulate a trigger we had to do a button push simulation where there is the initial press and a pause before the motor goes back and waits for a signal. This was accomplished by setting an initial degree (i) and then having it rotate 90 degrees by having a loop that increases the value of i to 90 and refreshing the motors stance each time it was incremented. From there it would reach the specified 90 degrees and then reset to 0 by changing the value of i and refreshing the motor once again.

7.2.3 App UI Testing

The UI testing will involve a two part testing procedure to see if the app itself works. The testing will include the testing of the UI to see if the app without the access to the turret or the database will work by itself and take us to the proposed sites. From there we need to check if the UI is able to enable or disable the feature flags so that when we do link it to the turret it will be able to be controlled.

7.2.9.1 Main Menu Testing

The main menu part of the testing is an app testing procedure mainly. The access will also be a three part procedure as the main menu will branch to different different pages. These pages for the most part will not be operational at this point, until the full UI is set up.

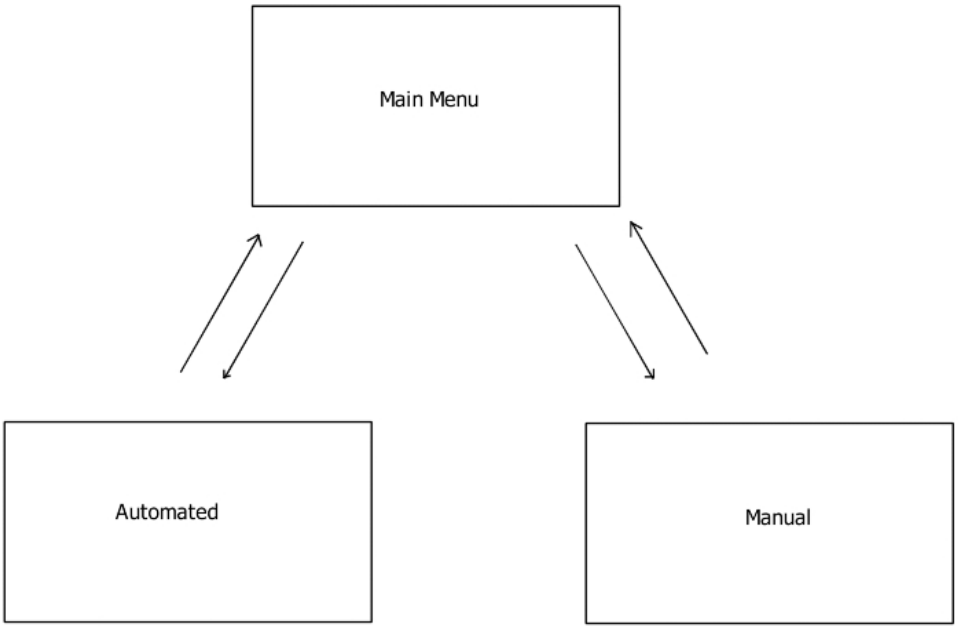


Figure 42: UI Navigation Procedure

As shown in Figure 42 we can see the proper procedure on how the page change is supposed to work. The main factor that we need to accomplish is that they will be able to navigate to and from each page. There will also be subsections to the Automated page which will also work in the same way as the procedure shown in the figure.

7.2.9.1 Remote Control Testing

The remote control testing will not only have the ability to control the various actions of the turret but it also needs to be able to navigate back to the main page. For the actions that the app needs to implement for the turret is to be able to view what the camera is able to see, move the turret left to right as well as going up and down, having the ability to shoot, and the last ability to reset its position back to the original. In order to test all these we did single tests starting with if we were able to view the camera directly from the app instead of having to view directly from the screen. The next step would be to test whether we are able to control it from the control interface in the app by moving it left right as well as up and down, and the last feature to reset the position back to normal. From there we would need to test the shooting capability from the app which would be a simple press of a UI button and see if the turret shoots.

7.2.9.1 Automated Controls Testing

Other than the user interface testing for the automated control page we needed to test for whether the automation part will be able to trigger the turret. These triggers will include whether the turret targets someone or not. The first step to this process is seeing if we have a connection to the database, to do this we had to view options to see if the

connection has been established to the database. If the connection is set up then the pictures will be available to see in the viewing page of the app. From there we can delete, raise the flag for a picture to be a target, or add a fresh person or target to the database. If the picture that is uploaded is targeted by the turret then this will be a success.

7.2.4 Bluetooth Testing

The bluetooth testing will involve connecting the proper pins to one another on the microcontroller and on the bluetooth module itself. After that we then connected the Arduino to the PC with the upload cable. The ino file needs to be uploaded to the Arduino. We checked if a stable connection was available to our smartphone after pairing. Via a bluetooth terminal app, we had LED's signal to test the functionality. From there we needed to test the nerf sentry's movement once coded. The next step is to test the data transmission of facial recognition and facial identification by the camera through bluetooth to display on the smartphone's screen.

7.2.5 Facial Recognition Testing

The facial recognition testing will not really involve the turret itself. This test will try to see if the raspberry pi is able to identify a face through the camera and display it to the screen. We made a few people viewable to the camera and see if the person appears with a box on the screen, after that was accomplished we needed to test if we are able to get coordinates and follow the face. In order to do so we first plugged the motors into the pi for the time being so that we can see if the information can be transferred into the motors to rotate the views. The view must center the faces on the view so the camera can be aligned with the target to initiate firing. The next step would be to move the motors over to the development and to pass the information over. The information is supposed to move the turrets around and see if we are able to do that with the same test of having someone tracked by the camera.

7.2.6 Facial Identification Testing

Facial recognition is one part of the facial identification process. The identification is figuring out if a person is identifiable or not and if they are then who would that person be. To test this process we worked off the facial recognition testing process, once all of that has been accomplished we needed to establish whether a person is a target or not and only track the people that are flagged. Testing this will require two people or pictures of faces and move them around once we have allowed the recognition process to identify them we can change the settings to one of them being a target. Then we ran the process of moving the faces around and see if we are able to follow the people that are the target rather than the person that is not one.

7.2.7 Database Testing

As the database is the primary way that we were able to proceed with the facial identification part of the turret this became very important to make sure it is working. In

order to test and see if this is working we then had to do a two part test to see how it is storing images and whether we have access to these images through the use of the bluetooth.

7.2.8.1 Storage Testing

The storage testing part will consist of a writing and reading part of the turret. Once the turret recognition was working we then used a temporary storage access to see if the turret is able to write to the database. Once the turret finds a face they will immediately write the picture to the database in which case we then access it locally to see if it was able to write and save it. The next step of this process is to see if we are able to read from the storage device from the turret so we had to name the picture in the database so that once it is accessed it will show up through the software view and reveal it on screen with the name on it. From there we restored the database access to the turret and allowed it to run. From there we could also see from a screen whether the turret is able to identify a name from the database and put a face to it.

7.2.8.2 Remote Storage Access Testing

The remote storage access was a similar form of testing but rather than testing and taking out the database to see if anything was written we were able to view it remotely from the app that we designed ourselves. The app was able to access and view and write to the database remotely through the bluetooth module. The testing starts from the remote access to the database itself to see if we are even able to see if the database is being accessed so we had the turret running and identifying a new face to see if the database updates and we are able to view this from the app. From there we were able to establish a target from the app and see if the turret will fire at that specified person. The next step would be to either delete the person or remove them from the targeted list and to reassess the turret’s targeting system to see if it will fire or not.

7.2.9 MicroController Testing

The first step in testing the MicroController was to connect it via USB cable to the PC. This connection via USB will enable us to send the program we wish to run on the computer with our Arduino and have it be able to send data back to the computer. A test program was the next necessary step to ensure the MicroController board is working properly and the program is running from the Arduino IDE.

7.2.11 Distance Measuring Testing

The distance measuring test is a simple task of seeing if it is operating at first. The second step was to establish a connection from the LIDAR system to see if we can get continuous readings from the device and display it on to the screen. Once that is done the next step would be to figure out a way to turn the device on and off from the programming and make it run only from the microcontrollers on command which can be accomplished by setting flags and seeing if the device's laser is operational. This was the

base requirements for testing the last step would be to see if we can read data from the LIDAR and see if the data taken can be used to trigger the firing of the turret.

8. Administrative Content

8.1 Milestone Discussion

Task	Assignee	Due Date
Preliminary Project Research	All Members	9/30/2021
Assign Roles for Research and Design	All Members	9/30/2021
Senior Design 1 Assignments		
Initial Document	All Members	9/17/2021
Updated Divide and Conquer	All Members	10/1/2021
60 Page Draft	All Members	11/5/2021
100 Page Draft	All Members	11/19/2021
Final Draft	All Members	12/7/2021
Senior Design 1 Component Research		
Motors, Actuators, Servos	Juan	11/23/2021
Wireless Communication Methods	Daanish	11/23/2021
Power Delivery	Kyle	11/23/2021
Circuit Design	Juan / Kyle	11/23/2021
Microprocessors	Juan	11/23/2021
Firing Mechanism	Ross	11/23/2021
Camera System	Juan	11/23/2021
Computer Vision Technologies	Daanish	11/23/2021
chassis Design	Juan	12/17/2021
Wiring Diagram	Kyle	12/17/2021
Research Programming Language for MCU		12/17/2021
PCB Design	Juan	1/14/2021
Winter Break		
PCB Mockup	Juan / Kyle	1/14/2022
chassis Mockup	Juan	1/28/2022
Begin Writing Code		1/28/2022
Senior Design 2 Deadlines		

Begin Order of Parts	All Members	1/28/2022
Assemble Custom PCB	Juan / Kyle	2/11/2022
Assemble Turret Controls	Daanish / Ross	2/11/2022
Build Prototype	All Members	3/11/2022
Finish Coding of Turret		3/25/2022
Test Prototype	All Members	4/1/2022
Finalize Assembly	All Members	4/1/2022
Presentation	All Members	TBD
Final Report	All Members	TBD
Final Presentation	All Members	TBD

Table 15: Project Milestones

Throughout the course of Senior Design 1, our group was required to research various technologies to incorporate into our project. We were then able to make a decision as to which technologies we think would help us best complete our project given our design specifications. It was our goal to have our custom PCB design almost completely finished before the end of the semester, and we want to have a good idea as to how we would like our chassis to look.

Because our Senior Design 1 course takes place in the fall, we had three weeks off of classes for winter break. During this time we would have liked to put any finishing touches on our custom PCB and we started gathering materials to use for our chassis. We were anticipating that the PCB design will require the most time to develop and troubleshoot therefore we are starting the design early. We also developed some of the programming that we used for the MCUs to control the various motors and other electrical components in our turret.

For Senior Design 2, most of our deadlines are centered around the assembly of our turret and ensuring all of the parts are in working order. The last month and a half of Senior Design 2 will consist mostly of testing and presenting our design.

8.2 Budget and Finance Discussion

Quantity	Component	Value	Manufacturer	Price (from)
3	Battery Housing	CR2032H	FDK	1.39
3	XT60 Connector	PRT-12638		3.23
3	Timing Device	DS1307Z+	MAXIM INTEGRATED PRODUCTS	2.45
3	MCU	ATMEGA328-PU	MICROCHIP	2.37
3	Voltage Regulator	LM1084IS-ADJ/NOPB	TEXAS INSTRUMENTS	3.53
12	3 pin Analog Header			1.26
3	Crystal Oscillator	8MHz	TXC	0.36
3	Crystal Oscillator	32.768kHz	ABRACON	0.16
3	PCB from Oshpark			50.41
1	Nerf Blaster			20.84
1	LIDAR			27.68
2	Arduino Unos			20
1	Pixycam			60
1	7.4V LIPO Battery			29.9
1	LiPO charger			24.95
2	SG92R Servo			5.95
1	MG995R Servo			13
1	JBtek HC-05			11.99
			Total:	326.93

Table 16: Electronic Component Part-out

The total budget of our project can be broken down into several categories to help determine what the final cost will be. These categories are; PCB, circuit components, chassis, sensors, motors, and firing mechanism. We want to design our project to be as inexpensive to make as possible, and we have made several design decisions in order to cut costs and simplify our design. One of the first decisions we made was to use a Nerf gun as our firing mechanism. This saved us the money and time of having to design and build one ourselves. The Nerf blaster we purchased comes with its own internal circuitry and motors needed to fire the gun without having to manually load a new round in the chamber after firing.

Another cost saving measure we have taken is to assemble our custom PCB ourselves. By having just the boards manufactured and then soldering all of our parts on our own, we are able to order three blank PCBs, and the parts needed to populate them for only $87.40. This would give us enough spare parts to assemble three complete boards, and if we messed up a component or damaged a board, we would have two backups. If we wanted to pay a company to assemble three boards for us it would cost somewhere between $600-900 to get three fully assembled boards.

The largest thing that is left unaccounted for in our budget is the cost of materials to build the chassis of our turret. Some of the options for building our chassis include getting a 3-D printed design made out of plastic resin, building the housing out of plyboard, or building the chassis out of extruded aluminum. The cheapest option for us would be to just buy a ply board from a hardware store and cut out the different panels we would need to make a housing. In total we expect our budget to total approximately $400 after purchasing the various materials needed to construct the chassis.

Appendices:

Citations:

1- "Lithium-Ion Battery." Clean Energy Institute, 25 Sept. 2020, https://www.cei.washington.edu/education/science-of-solar/battery-technology/.

2- Keith Araujo - Epec, LLC. "Battery Cell Comparison." Epec Engineered Technologies - Build to Print Electronics, https://www.epectec.com/batteries/cell-comparison.html.

3- "Whats the Difference between Nickel Cadmium (Nicad), Nickel-Metal Hydride (Nimh), and Lithium Ion (Li-Ion)?" Battery Universe Online Battery and Accessories Sales since 1999, https://www.batteryuniverse.com/help/battery-chemistries.

4- CPU vs GPU Performance of Deep Learning Based Face ... https://gvis.unileon.es/wp-content/uploads/2019/12/CPU-vs-GPU-performance-of-deep-learning-based-face-detectors-using-resized-images-in-forensic-applications.pdf.

5-Wiki:v2:teach_pixy_an_object_2 [Documentation], https://docs.pixycam.com/wiki/doku.php?id=wiki%3Av2%3Ateach_pixy_an_object_2.

6-Muneeb, et al. "Face Detection with PixyCam (1 or 2)." Pixycam, 10 Oct. 2018, https://forum.pixycam.com/t/face-detection-with-pixycam-1-or-2/5857/3.

7-"Python: Haar Cascades for Object Detection." GeeksforGeeks, 6 May 2021, https://www.geeksforgeeks.org/python-haar-cascades-for-object-detection/.

8-Jack, Srujan. "Face Detection Using Dlib Hog." Medium, MLCrunch, 17 July 2020, https://medium.com/mlcrunch/face-detection-using-dlib-hog-198414837945.

9-Behera, Girija Shankar. "Face Detection with Haar Cascade." Medium, Towards Data Science, 29 Dec. 2020, https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08#:~:text=So%20what%20is%20Haar%20Cascade,Simple%20Features%E2%80%9D%20published%20in%202001.

10-"Face Detection on a Raspberry Pi." UCL Connected Environments, 13 Nov. 2020, https://connected-environments.org/making/face-detection-on-a-raspberry-pi/.

11-Paper - Carnegie Mellon School of Computer Science. https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf.

12-ZeroZero 90311 gold badge88 silver badges1515 bronze badges, et al. "How Much Energy Does the Raspberry Pi Consume in a Day?" Raspberry Pi Stack Exchange, 1 Apr. 1961, https://raspberrypi.stackexchange.com/questions/5033/how-much-energy-does-the-raspberry-pi-consume-in-a-day.

13-"Power Consumption Benchmarks." Power Consumption Benchmarks | Raspberry Pi Dramble, https://www.pidramble.com/wiki/benchmarks/power-consumption.

14-Wiki:v2:powering_pixy, https://docs.pixycam.com/wiki/doku.php?id=wiki%3Av2%3Apowering_pixy.

15-Yashwant. "Various Techniques Used for Face Recognition." OpenGenus IQ: Computing Expertise & Legacy, OpenGenus IQ: Computing Expertise & Legacy, 10 Sept. 2019, https://iq.opengenus.org/techniques-for-face-recognition/.

16-Person. "UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter." UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices, https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html.

17-Duncan. "8-Bit Computer: UART Transceiver for Breadboard Computer." 8-Bit Computer: UART Transceiver for Breadboard Computer, Blogger, 28 Dec. 2020, https://shepherdingelectrons.blogspot.com/2020/07/uart-transceiver-for-breadboard-computer.html.

18-Person. "I2C Primer: What Is I2C? (Part 1)." I2C Primer: What Is I2C? (Part 1) | Analog Devices, https://www.analog.com/en/technical-articles/i2c-primer-what-is-i2c-part-1.html.

19-Anusha. "Basics of Serial Peripheral Interface (SPI)." Electronics Hub, 6 Oct. 2021, https://www.electronicshub.org/basics-serial-peripheral-interface-spi/.

20-ago, Shawn 2 years, et al. "Types of Distance Sensors and How to Select One?" Latest Open Tech From Seeed, 29 June 2021, https://www.seeedstudio.com/blog/2019/12/23/distance-sensors-types-and-selection-guide/.

21-Jost, Danny. "What Is an IR Sensor?" FierceElectronics, 30 July 2019, https://www.fierceelectronics.com/sensors/what-ir-sensor.

22-"Time-of-Flight Principle: Technologies and Advantages." Terabee, 23 June 2021, https://www.terabee.com/time-of-flight-principle/.

23-Sam, Abdullah. "Tof VS LIDAR: What's the Difference?" Notes Read, 21 Nov. 2020, https://notesread.com/tof-vs-lidar-whats-the-difference/.

24- Real Python. "Face Recognition with Python, in under 25 Lines of Code." Real Python, Real Python, 5 Mar. 2019, https://realpython.com/face-recognition-with-python/.

25-"HC-05 - Bluetooth Module." Components101, https://components101.com/wireless/hc-05-bluetooth-module.

26- HC-05 Bluetooth Module. https://www.gme.cz/data/attachments/dsh.772-148.1.pdf.

27-"Difference between Python and C++." GeeksforGeeks, 18 Oct. 2021, https://www.geeksforgeeks.org/difference-between-python-and-c/#:~:text=Python%20leads%20to%20one%20conclusion,data%20analysis%20and%20machine%20learning.

28-"All about HC-05 Bluetooth Module: Connection with Android." GeeksforGeeks, 28 Oct. 2021, https://www.geeksforgeeks.org/all-about-hc-05-bluetooth-module-connection-with-android/.

29- Robert Half. "4 Advantages of Object-Oriented Programming." 4 Advantages of OOP | Robert Half, Robert Half, 4 Dec. 2021, https://www.roberthalf.com/blog/salaries-and-skills/4-advantages-of-object-oriented-programming.

30- "Difference between C and C++." GeeksforGeeks, 15 Feb. 2021, https://www.geeksforgeeks.org/difference-between-c-and-c/.

31- says:, Isaac. "Top 5 Best Bluetooth Modules for Arduino [2021 Updated Review]." Solderingironguide, 31 Mar. 2021, https://solderingironguide.com/best-bluetooth-modules-for-arduino/.

32- Gill, Navdeep Singh. "Kotlin vs Java: Which Is Better for Android App Development?" XenonStack, XenonStack, 10 Aug. 2021, https://www.xenonstack.com/blog/kotlin-andriod.

33- Gill, Navdeep Singh. "Kotlin vs Java: Which Is Better for Android App Development?" XenonStack, XenonStack, 10 Aug. 2021, https://www.xenonstack.com/blog/kotlin-andriod.

Appendix B - Datasheets

This section is the source for the specifications of the microcontrollers, giving the electrical characteristics of each of the microcontroller that was researched and the peripherals, both in what each microcontroller offered and the amount that they offer.

3 Device Comparison

Table 3-1 summarizes the features of the MSP432P401x microcontrollers.

Table 3-1. Device Comparison(1)

DEVICE	FLASH (KB)	SRAM (KB)	ADC14 (Channels)	COMP_E0 (Channels)	COMP_E1 (Channels)	Timer_A(2)	eUSCI		20-mA DRIVE I/O	TOTAL I/Os	PACKAGE
							CHANNEL A: UART, IrDA, SPI	CHANNEL B: SPI, I²C			
MSP432P401RIPZ	256	64	24 ext, 2 int	8	8	5, 5, 5, 5	4	4	4	84	100 PZ
MSP432P401MIPZ	128	32	24 ext, 2 int	8	8	5, 5, 5, 5	4	4	4	84	100 PZ
MSP432P401RIZXH	256	64	16 ext, 2 int	6	8	5, 5, 5	3	4	4	64	80 ZXH
MSP432P401MIZXH	128	32	16 ext, 2 int	6	8	5, 5, 5	3	4	4	64	80 ZXH
MSP432P401RIRGC	256	64	12 ext, 2 int	2	4	5, 5, 5	3	3	4	48	64 RGC
MSP432P401MIRGC	128	32	12 ext, 2 int	2	4	5, 5, 5	3	3	4	48	64 RGC

(1) For the most current part, package, and ordering information for all available devices, see the *Package Option Addendum* in Section 9, or see the TI website at www.ti.com.
(2) Each number in the sequence represents an instantiation of Timer_A with its associated number of capture/compare registers and PWM output generators available. For example, a number sequence of 3, 5 would represent two instantiations of Timer_A, the first instantiation having 3 and the second instantiation having 5 capture/compare registers and PWM output generators, respectively.

Figure A.1 MSP432 Peripheral Sheet

5 Specifications

5.1 Absolute Maximum Ratings(1)

over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

	MIN	MAX	UNIT
Voltage applied at DVCC and AVCC pins to V_{SS}	–0.3	4.17	V
Voltage difference between DVCC and AVCC pins(2)		±0.3	V
Voltage applied to any pin(3)	–0.3	V_{CC} + 0.3 V (4.17 V MAX)	V
Diode current at any device pin		±2	mA
Storage temperature, T_{stg}(4)	–40	125	°C
Maximum junction temperature, T_J		95	°C

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions* is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) Voltage differences between DVCC and AVCC exceeding the specified limits may cause malfunction of the device.
(3) All voltages referenced to V_{SS}.
(4) Higher temperature may be applied during board soldering according to the current JEDEC J-STD-020 specification with peak reflow temperatures not higher than classified on the device label on the shipping boxes or reels.

5.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1) (2)	±1000	V
		Charged-device model (CDM), per JEDEC specification JESD22-C101(3)	±250	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process. Pins listed as ±1000 V may actually have higher performance.
(2) All pins except DVSS3 pass HBM up to ±1000 V. The DVSS3 pin is used for TI internal test purposes. Connect the DVSS3 pin to supply ground on the customer application board.
(3) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process. Pins listed as ±250 V may actually have higher performance.

5.3 Recommended Operating Conditions

Typical data are based on V_{CC} = 3.0 V, T_A = 25°C (unless otherwise noted)

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage range at all DVCC and AVCC pins (1) (2) (3)	At power-up (with internal V_{CC} supervision)	1.71		3.7	V
		Normal operation with internal V_{CC} supervision	1.71		3.7	
		Normal operation without internal V_{CC} supervision	1.62		3.7	
V_{SS}	Supply voltage on all DVSS and AVSS pins			0		V
I_{INRUSH}	Inrush current into the V_{CC} pins(4)				100	mA
f_{MCLK}	Frequency of the CPU and AHB clock in the system(5)		0		48	MHz
T_A	Operating free-air temperature		–40		85	°C
T_J	Operating junction temperature		–40		85	°C

(1) TI recommends powering AV_{CC} and DV_{CC} from the same source. A maximum difference of ±0.1 V between AV_{CC} and DV_{CC} can be tolerated during power up and operation. See Section 5.4 for decoupling capacitor recommendations.
(2) Supply voltage must not change faster than 1 V/ms. Faster changes can cause the VCCDET to trigger a reset even within the recommended supply voltage range.
(3) Modules may have a different supply voltage range specification. See the specification of the respective module in this data sheet.
(4) Does not include I/O currents (driven by application requirements).
(5) Operating frequency may require the flash to be accessed with wait states. See Section 5.8 for further details.

Figure B.2 MSP432 Electrical Specifications

2.2 Comparison Between ATmega1281/2561 and ATmega640/1280/2560

Each device in the ATmega640/1280/1281/2560/2561 family differs only in memory size and number of pins. Table 2-1 summarizes the different configurations for the six devices.

Table 2-1. Configuration Summary

Device	Flash	EEPROM	RAM	General Purpose I/O pins	16 bits resolution PWM channels	Serial USARTs	ADC Channels
ATmega640	64KB	4KB	8KB	86	12	4	16
ATmega1280	128KB	4KB	8KB	86	12	4	16
ATmega1281	128KB	4KB	8KB	54	6	2	8
ATmega2560	256KB	4KB	8KB	86	12	4	16
ATmega2561	256KB	4KB	8KB	54	6	2	8

2.3 Pin Descriptions

2.3.1 VCC

Digital supply voltage.

2.3.2 GND

Ground.

2.3.3 Port A (PA7..PA0)

Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on page 75.

2.3.4 Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B has better driving capabilities than the other ports.

Port B also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on page 76.

2.3.5 Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of special features of the ATmega640/1280/1281/2560/2561 as listed on page 79.

Figure B.3 Atmega2560 Peripheral Sheet

11. Power Management and Sleep Modes

Sleep modes enable the application to shut down unused modules in the MCU, thereby saving power. The AVR provides various sleep modes allowing the user to tailor the power consumption to the application's requirements.

11.1 Sleep Modes

Figure 10-1 on page 39 presents the different clock systems in the ATmega640/1280/1281/2560/2561, and their distribution. The figure is helpful in selecting an appropriate sleep mode. Table 11-1 shows the different sleep modes and their wake-up sources.

Table 11-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains					Oscillators		Wake-up Sources						
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Osc Enabled	INT7:0 and Pin Change	TWI Address Match	Timer2	SPM/ EEPROM Ready	ADC	WDT Interrupt	Other I/O
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X
ADCNRM				X	X	X	X(2)	X(3)	X	X(2)	X	X	X	
Power-down								X(3)	X				X	
Power-save					X		X(2)	X(3)	X	X			X	
Standby(1)						X		X(3)	X				X	
Extended Standby					X(2)	X	X(2)	X(3)	X	X			X	

Note: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT7:4, only level interrupt.

To enter any of the sleep modes, the SE bit in "SMCR – Sleep Mode Control Register" on page 54 must be written to logic one and a SLEEP instruction must be executed. The SM2, SM1, and SM0 bits in the SMCR Register select which sleep mode will be activated by the SLEEP instruction. See Table 11-2 on page 54 for a summary.

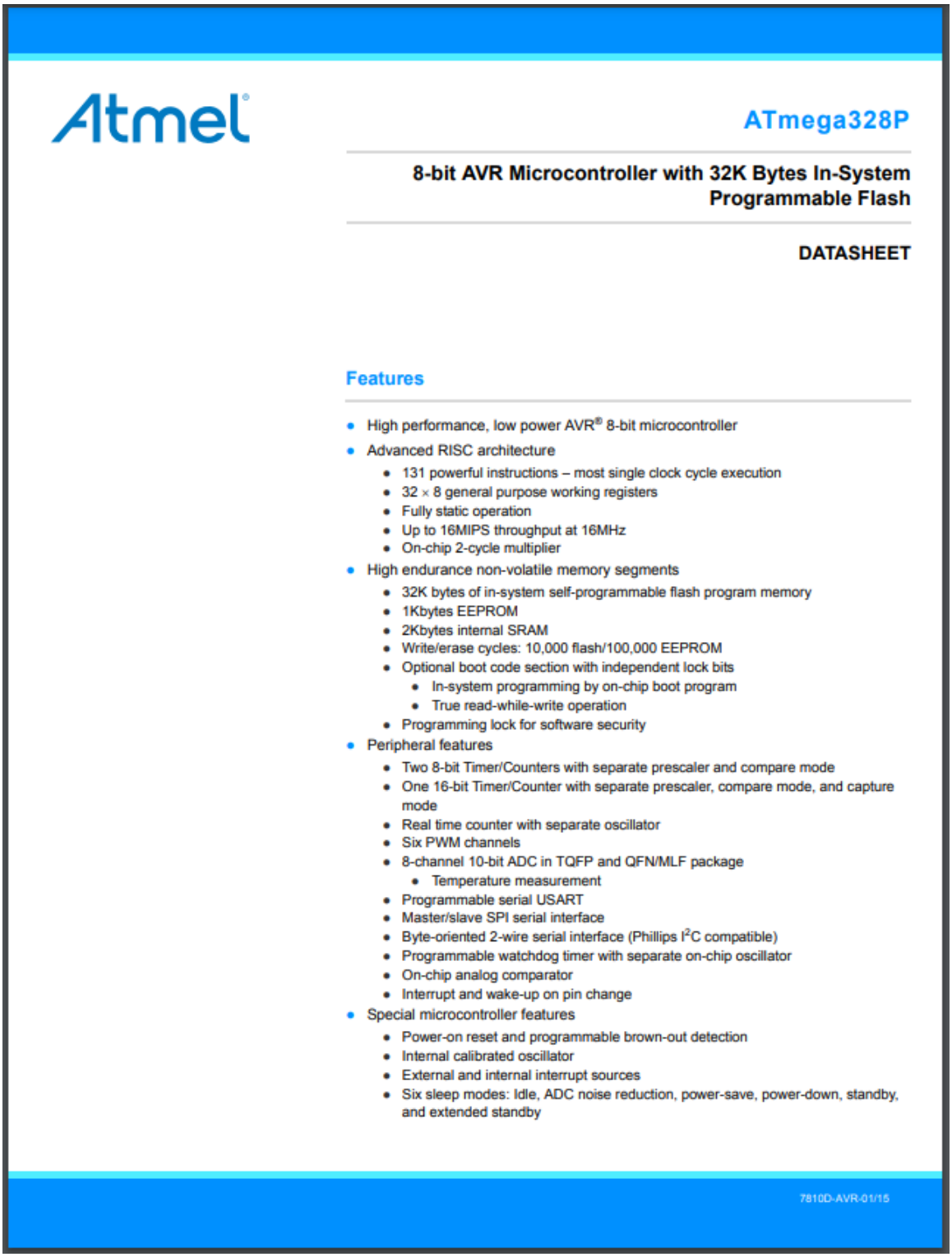
If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU wakes up. The MCU is then halted for four cycles in addition to the start-up time, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the Register File and SRAM are unaltered when the device wakes up from sleep. If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset Vector.

11.2 Idle Mode

When the SM2:0 bits are written to 000, the SLEEP instruction makes the MCU enter Idle mode, stopping the CPU but allowing the SPI, USART, Analog Comparator, ADC, 2-wire Serial Interface, Timer/Counters, Watchdog, and the interrupt system to continue operating. This sleep mode basically halts clk_{CPU} and clk_{FLASH}, while allowing the other clocks to run.

Idle mode enables the MCU to wake up from external triggered interrupts as well as internal ones like the Timer Overflow and USART Transmit Complete interrupts. If wake-up from the Analog Comparator interrupt is not required, the Analog Comparator can be powered down by setting the ACD bit in the Analog Comparator Control and Status Register – ACSR. This will reduce power consumption in Idle mode. If the ADC is enabled, a conversion starts automatically when this mode is entered.

Figure B.4 Atmega2560 Power Management Sheet



Atmel

ATmega328P

8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash

DATASHEET

Features

- High performance, low power AVR® 8-bit microcontroller
- Advanced RISC architecture
  - 131 powerful instructions – most single clock cycle execution
  - 32 × 8 general purpose working registers
  - Fully static operation
  - Up to 16MIPS throughput at 16MHz
  - On-chip 2-cycle multiplier
- High endurance non-volatile memory segments
  - 32K bytes of in-system self-programmable flash program memory
  - 1Kbytes EEPROM
  - 2Kbytes internal SRAM
  - Write/erase cycles: 10,000 flash/100,000 EEPROM
  - Optional boot code section with independent lock bits
    - In-system programming by on-chip boot program
    - True read-while-write operation
  - Programming lock for software security
- Peripheral features
  - Two 8-bit Timer/Counters with separate prescaler and compare mode
  - One 16-bit Timer/Counter with separate prescaler, compare mode, and capture mode
  - Real time counter with separate oscillator
  - Six PWM channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature measurement
  - Programmable serial USART
  - Master/slave SPI serial interface
  - Byte-oriented 2-wire serial interface (Phillips I^2C compatible)
  - Programmable watchdog timer with separate on-chip oscillator
  - On-chip analog comparator
  - Interrupt and wake-up on pin change
- Special microcontroller features
  - Power-on reset and programmable brown-out detection
  - Internal calibrated oscillator
  - External and internal interrupt sources
  - Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby, and extended standby

7810D-AVR-01/15

Figure B.5 Atmega328 Peripheral Sheet

28. Electrical Characteristics

All DC/AC characteristics contained in this datasheet are based on characterization of Atmel® ATmega328P AVR® microcontroller manufactured in an automotive process technology.

28.1 Absolute Maximum Ratings

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Parameters	Min.	Typ.	Max.	Unit
Operating temperature	–55		+125	°C
Storage temperature	–65		+150	°C
Voltage on any pin except $\overline{\text{RESET}}$ with respect to ground	–0.5		V_{CC} + 0.5	V
Voltage on $\overline{\text{RESET}}$ with respect to ground	–0.5		+13.0	V
Maximum operating voltage		6.0		V
DC current per I/O pin		40.0		mA
DC current V_{CC} and GND pins		200.0		mA
Injection current at V_{CC} = 0V		±5.0(1)		mA
Injection current at V_{CC} = 5V		±1.0		mA

Note: 1. Maximum current per port = ±30mA

28.2 DC Characteristics

T_A = –40°C to +125°C, V_{CC} = 2.7V to 5.5V (unless otherwise noted)

Parameter	Condition	Symbol	Min.	Typ.	Max.	Unit
Input low voltage, except XTAL1 and $\overline{\text{RESET}}$ pin	V_{CC} = 2.7V to 5.5V	V_{IL}	–0.5		$0.3V_{CC}$(1)	V
Input high voltage, except XTAL1 and $\overline{\text{RESET}}$ pins	V_{CC} = 2.7V to 5.5V	V_{IH}	$0.6V_{CC}$(2)		V_{CC} + 0.5	V
Input low voltage, XTAL1 pin	V_{CC} = 2.7V to 5.5V	V_{IL1}	–0.5		$0.1V_{CC}$(1)	V
Input high voltage, XTAL1 pin	V_{CC} = 2.7V to 5.5V	V_{IH1}	$0.7V_{CC}$(2)		V_{CC} + 0.5	V

Notes:
1. "Max" means the highest value where the pin is guaranteed to be read as low
2. "Min" means the lowest value where the pin is guaranteed to be read as high
3. Although each I/O port can sink more than the test conditions (20mA at V_{CC} = 5V, 10mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   ATmega328P:
   1] The sum of all I_{OL}, for ports C0 - C5, should not exceed 100mA.
   2] The sum of all I_{OL}, for ports B0 - B5, D5 - D7, XTAL1, XTAL2 should not exceed 100mA.
   3] The sum of all I_{OL}, for ports D0 - D4, should not exceed 100mA.
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (20mA at V_{CC} = 5V, 10mA at V_{CC} = 3V) under steady state conditions (non-transient), the following must be observed:
   ATmega328P:
   1] The sum of all I_{OH}, for ports C0 - C5, D0- D4, should not exceed 150mA.
   2] The sum of all I_{OH}, for ports B0 - B5, D5 - D7, XTAL1, XTAL2 should not exceed 150mA.
   If II_{OH} exceeds the test condition, V_{OH} may exceed the related specification. Pins are not guaranteed to source current greater than the listed test condition.

Figure B.6 Atmega328 Electrical Specification Sheet