Autonomous Quadrotors Using Attack Logic Under Navigational Guidance

Senior Design 1 Fall 2012

Group 4

Yousef Al-Khalaf

Chris Culver

Shane Parker

Bianca Wood

Table of Contents

1.0 Executive Summary

Autonomous Quadrotors Using Attack Logic Under Navigational Guidance, or A.Q.U.A.L.U.N.G., was a project that is described fairly well by the title. Specifically, this project endeavored to design a team of four quadrotors, along with all of the artificial intelligence required for autonomous behavior that would be able to play competitively against human opponents. The quadrotors must have been able to act and respond in real-time, as well as make intelligent tactical decisions to provide a genuine challenge for their human counterparts. This research has potential to make significant contributions to the advancement of autonomous robotics and practical application of AI techniques. In order to carry this project through to fruition, many diverse systems have been designed and brought together in an efficient manner. It is the intention of this report to lay down a specific design plan and describe the completion and integration of the multitude of facets of the project.

While covered in greater depth later in this document, a brief explanation of the motivations behind this project should be provided. The number of contributing motivations for this project was large, but the main influence was the personal fascination with creating something that could fly of its own accord. Simply put, the A.Q.U.A.L.U.N.G. project was fun to work on.

Creating an autonomous robotic system, such as the team of quadrotors used in the A.Q.U.A.L.U.N.G. project, required the implementation of dozens of technologies and the design of a coherent means of combining them. To facilitate an organized approach, every component of the project was matched to one of four subsystems. These subsystems were:

1. Artificial Intelligence
2. Power Management
3. Flight Control
4. Computer Vision

Each subsystem was interconnected with the others in a manner that was straightforward, making their integration a more manageable task than trying to connect of the dozens of systems to each other individually. The general organization was that every subsystem fed information to the AI system, which used this information to make real-time tactical decisions. Then the AI subsystem sent navigation and fire control commands to the flight control subsystem, or a shutoff command to the power management subsystem when applicable. Additionally the AI subsystem assisted the Computer Vision system by providing information about the environment, such as expected edges given an individual quadrotor's position and orientation. An exhaustive description of the design criteria and decisions made for each of these subsystems is covered later in this report.

2.0 Project Description

2.1 Project Motivation

There were many factors in the motivation for this project, ranging from personal to practical. The practical motivations for this project revolved around how it can be used and applied. The personal motivations were responsible for choosing this specific project, as opposed to one of many others that may also have had practical applications.

The most direct practical application for this project is for entertainment. The quadrotors are designed to play laser tag competitively against human opponents, which allows for a novel experience for the players. Laser tag is primarily an entertainment oriented activity, and the added component of playing against small, agile, aerial robots, as opposed to other human players, would increase the choices players have in their experience. This increase in the available choices would allow for prolonged enjoyment, due to it removing the potential for monotony in fighting the same opponents every time. This effect could be further increased with future research and development on this project by creating differing “personalities” or play styles in the quadrotors.

In addition to diversifying the opponent choices players have in laser tag, the addition of robotic opponents allows for the creation of a different story for the players to enter into. A typical laser tag scenario is merely a skirmish between two waring factions in whatever environment is being simulated by the arena they are competing in. Competing against these quadrotor robotic opponents would allow for additional scenarios to be played out, such as a machine uprising and a fight for the survival of the human race.

Another practical motivation for this project was in the potential application of police or military use. While there are distinct differences in the objectives and methods of police and military units, for our purposes, these differences are negligible. The first of these potential applications is in training. In a training situation, the quadrotors would be set to merely observe the combat for After Action Review, as training directly against the robots would not be as applicable as training against other human opponents. However, having the quadrotors follow the combat and record the actions of the combatants would allow for more thorough review than the placement of static cameras or the use of human observers, who can only follow the combatants. The Artificial Intelligence in the A.Q.U.A.L.U.N.G. project is designed to be able to predict the flow of combat and intercept human combatants, which means that it would have practical use in monitoring

the combat of a training team, to record video from angles that give reviewers data that static camera placement couldn't provide.

The quadrotors in the project have also laid some early groundwork for active use of quadrotors in a police or military environment. With some modifications, the robots would be able to go into a hostile environment and scout without risking human life. Once finding any hostile opposition, they would be able to relay this information back to either the police or military and either continue searching or maintain contact. Given the fact that the quadrotors are designed to operate in a simulated combat environment it would be perfectly reasonable to adapt them to a real one. Of course there would need to be some additional work done in order for them to maintain equal effectiveness in an unknown, or less well known environment, but the concepts of maintaining awareness of an opponent's position, and avoiding detection and damage are both applicable.

The final practical application for this project was not quite as direct as the previous options above, but it was tied closely to the most recent one. This project can be used to make early progress in the ability for the military not only to deploy Unmanned Aerial Vehicles, which have no personnel on board, but are operated by a flight team from a control center, but also to deploy Autonomous Aerial Vehicles, that operate without any human involvement. These vehicles could then operate and perform a given task without the risk of operator error, whether due to fatigue, distractions, or any other reason. Of course this application is still some time out, as any system must be as close to perfect as is possible before it can be deployed for autonomous military use, but without the early steps toward this end being made, that goal will never be attained.

In addition to the potential future applications for this research that constitutes the practical motivations for this project, there were several personal motivations behind the decisions to pursue research in the A.Q.U.A.L.U.N.G. project. The first of these personal motivations tied in very closely with the practical applications. Each member in the group felt that this project was the most significant undertaking in their academic career, and wished for this undertaking to have genuine, substantial purpose, as opposed to merely being another assignment completed for grade alone. This desire stemmed from the fact that each member was approaching graduation with a degree that states they are qualified to enter the profession of engineers, and a facet of the definition of a profession is that it is "a calling... which has for its primary purpose the rendering of a public service." This means that engineers work to provide for society as a whole, and do not strive solely for personal rewards and positive critiques of performance.

Another aspect of personal motivation for the A.Q.U.A.L.U.N.G. project was also traceable back to the definition of a profession, that is, "a calling requiring specialized knowledge". Not only was it desired for the project to be useful, but also to be challenging. Each member desired to work on a project that is novel, and demanded the use of the "specialized knowledge" that had been accrued during their undergraduate academic career, not simply a project that "reinvented the wheel", so to speak. During

the undergraduate career of many engineering students, the only projects that are undertaken are ones that have already been done before. There is often already a known solution, and the student is taught by following in the footsteps of those who have come before them and worked on the project, learning the research and development process by repeating it, step for step. While this is occasionally the case for professional engineers, as they continue their education, often times they are required work on something new, to explore uncharted territory. Therefore, part of the motivation for this project was to gain experience in working on a problem that had not been solved, to make footsteps for others to follow.

In addition to being motivated to work on a project that required specialized knowledge that had already been learned, it was desired to work on a project that allowed for growth into areas that each group member had not worked in before. Each member was motivated to learn new topics during work on the A.Q.U.A.L.U.N.G. project and broaden their field of expertise. The trials that life may bring in the future are unknown, thus having a more diverse set of skills and experiences often times proves to be highly beneficial.

The final personal motivation for the A.Q.U.A.L.U.N.G. project was the sheer fun of it. Each member of the group is fascinated by artificial intelligence, inspired by robotics, or passionate about aeronautics. This means that all practical applications aside, all motivations for personal improvement disregarded, and all desires to demonstrate skill and proficiency put on hold, each member wished to be a part of this project for the simple joy of working with a product of artificial intelligence, robotics, and aeronautics. Whether a certified pilot in love with flying, or nostalgic about a similar project influencing the decision to become an engineer, a hobbyist tinkerer always working on pet projects, or just wanting to make something come to life and fly around the room, no member of this group would dream of passing up having worked on this project.

2.2 Objectives and Goals

The objective of the A.Q.U.A.L.U.N.G. was to have four quadrotors play a game of laser tag against four human opponents. The four quadrotors worked as a team autonomously, navigating the course while talking to each other as to relay position and possible opponents back and forth from one quadrotor to another. It was important for the quadrotors to be able to pose a real and imminent threat to the human players in the game, as this makes the game entertaining for players and spectators alike. So to turn the quadrotors from moving targets into intelligent opponents a couple things had to happen. The quads had to be agile, fire with good accuracy and they had to be able to make decisions quickly and dynamically, making them unpredictably elusive.

The quadrotors did not have to map their own environment when searching and destroying, they navigated in a know environment that was pre mapped and programmed into the AI. This factor gave the quadrotors a leg up on the competition, if and only if the opponent had never navigated the game course before, making them an even more threatening force against the humans. If the human opponents had navigated the course before then they start the game with the same knowledge as the quadrotors, a mental map of the course. It was an objective to teach the quadrotors how to use the maps natural advantages, such as angles and niches that offer the most defense against an opponent sneaking up from behind, and other combat formations or combat techniques that would maximize the chances of victory for the robots.

Hunting down an intelligent enemy while that enemy, itself, is hunting you down is hard enough when you have all five senses working for you. Now imagine if the gift of sight were to be taken out of the equation, game over right? Well the quadrotors had to have the gift of sight if this is going to be a fair fight, and a fair fight is what this project was all about. So the objective of the video camera on each quadrotor was to, of course, capture images, the images the quadrotors were seeing, and send them back to the central computer. Although, like most other things on the quadrotors, the camera had to be a balance of resolution, weight and cost. If the camera did not have the resolution to be able to send back images that the Artificial Intelligence can decipher, than there would not even have been a point to having a camera in the first place. So the camera had to be able to show one object from another, without being too bulky or too expensive, as weight on the fly machine was incredibly important, just as much as weight on a college student's wallet is.

From the time light hits a human retina to the time it is translated into a visual perception is about one-tenth of a second. [1] From then on it depends on the human as to how long it takes to recognize, make a decision, and put that decision into action. Someone with more experience in real time combat will have trained the brain to do these actions a lot quicker than, let's say, four teenagers on a Friday night looking to have some fun (mainly the target audience for the project at hand). So the speed of the image processing for the A.Q.U.A.L.U.N.G. greatly depends on the intended audience of players. The objective was to cater to a lesser trained more relaxed combat environment than advanced military training, so the image processing needed to be somewhere in the vicinity of keeping up with normal human capabilities if it were to be on a competitive level.

For stabilization, the quadrotor used an Inertial Measurement Unit on a microprocessor to determine its acceleration, pitch, yaw and roll. For the quadrotor to be a successful laser tag playing machine it needed to be able to

self stabilize whilst navigating, shooting and evading. The objective was to have the quadrotors be able to altitude hold, hover in the same spot without moving, turn 360 degrees on a pin, and have full range of movement on the x, y and z axis. With all this freedom of movement it was possible to program the quadrotors to do advanced combinations, such as shooting a fixed or moving target whilst the quadrotor itself was moving. If the artificial intelligence evaluated too big of a threat from the enemy for a particular quadrotor, it was possible to not only move and shoot at the same time, but another advantage was that it could have successfully retreated out of harm's way as well. Advanced movement and stabilization was definitely a key in the effectiveness of the quadrotors tactical ability.

For keeping track of each of their positions, each quadrotor used dead-reckoning. Dead-reckoning is a method of estimating the position of an aircraft or a ship, as by applying to a previously determined position the course and distance traveled since. [2] So by using the Inertial Measurement Unit the quadrotors would have been able to calculate their positions as they moved throughout the game course. There is always error associated with tracking ones position this way, as the aircraft can drift in any direction depending on the environmental circumstances. There is also error associated with the math done by the IMU and Microprocessor which will be discussed later on in the Flight Control subsystem. So with these errors and miscalculations dead-reckoning would have had to report back the position of the quadrotors with a certain tolerance in mind. For example, if the sensors on the IMU could have reported back an accurate position within plus or minus X number of inches to its actual position, it would have been considered acceptable and could have been used with confidence. However if the error had become too great, and the microprocessor was saying that the aircraft is in one position, when it was really a foot away this is no good and cannot be trusted to accurately guide our aircraft through the course safely. So the objective for using dead-reckoning would have been to find this tolerance point and not exceed it, but also possibly have a check point where the quadrotor could reset its position every so often within the course so no large drift or error could have built up causing inaccurate readings.

In order for all of these tactical movements, retreating movements and images to be processed, there needed to be an intelligent system to make quick and important decisions. This needed to be done while not only thinking about the current situation but also while thinking ahead, just as commanders do in real life combat. The Artificial Intelligence that was programmed into the central computer took care of all of these decisions. The most important aspect of the artificial intelligence for the quadrotors was their ability to act unpredictably even though

they would be roaming the same course over and over. In the heat of battle, human players will have to be at the top of their game, triggering high cognitive activity. One of the most critical skills in intelligent decision making is the ability to spot existing or emerging patterns, although humans are not even aware they are doing it. [3] So in order to maximize the ability of the quadrotors to play competitively against humans, dynamic decision making was an absolute must. If the quadrotors wondered the same path over and over in search of its enemy, that pattern could easily be decoded and it would cause certain doom for the quadrotors.

Just as humans need food to convert to energy, the quadrotors needed an energy source to power them; this energy source was provided by batteries. Batteries are a very reliable steady source of energy for the short term and can easily be recharged for another short term use. This made them perfect for the application of powering everything needed on the quadrotors during a game of laser tag. The objective was to figure out how to maximize the energy output of the batteries while keeping their weight down. For example: it would have been easy to put five battery packs onto the quadrotor and power all the electronics for quite a while, but with five battery packs on it, would it be able to fly? Or maybe, only two battery packs are needed to power the electronics and lift the quadrotors, but it can only stay in the air for half a minute or so. So it was essential to find the right balance for the demands of laser tag combat.

For the quadrotors to be able to communicate with each other and the main computer, a wireless communication system had to be in place. Each quadrotor would have to have a transceiver on it that could send data back to the computer. The computer would also have to have a transceiver on it, but it had to be able to receive data from four transmitters at once, since of course there would be four quadrotors talking to it at all times throughout the game. The wireless communication system would have to be able to send quite a bit of data through as the quadrotors will be sending back and forth a couple things. They would be sending back to the computer their position in the field, which was determined by dead-reckoning, they would also be sending back the data from the video camera that is on board the quadrotors. All this information needed to be able to be sent back while the computer was sending information to the quadrotors. The computer would be telling the quadrotors where to move, how to move and how fast to do it, as well as telling the quadrotors to fire the laser that is attached to the bottom of each one. With so much critical information going back and forth, the wireless communication system was one of the most important aspects of the whole operation.

The brains of the operation, as a whole, was the central computer. This unit was in control of piecing all the components together, and making them all interact with each other as one. The objective of the central computer was to have the computing power to make this happen, but since the quadrotors would in combat as the computer gets information, it not only had to be able to compute the information and send out commands, it needed to do so in real time with zero lag. Without a fast processor the quadrotors could be getting commands to shoot way too late, the opponent could have already moved out of harm's way, or worse, shot down the quadrotor ending its reign of terror over the battle field before it even began.

2.3 Project Requirements and Specs of Hardware

The project requirements and specs were split into two categories hardware and software. For the hardware, the requirements ranged from physical characteristics of the parts, which could not be changed, to performance of the parts, which could have depended on a number of factors. The software side of the requirements dealt with manipulating code to reach certain goals within A.Q.U.A.L.U.N.G..

For physical characteristics of the hardware the frame will be first up, the frame needed to be strong enough to be able to hold all of the components that were placed onto each quadrotor, yet light enough to be able to be lifted off the ground. The more weight the quadrotor had on it, the bigger the frame needed to be, since more powerful motors would have be needed, larger propellers and to power it all a larger battery. For A.Q.U.A.L.U.N.G. the frame from motor to motor had a diameter of 36 inches. This was due to the fact that it had to successfully carry more than the normal load for a quadrotor, and in consequence the propellers needed to be a little larger. The propellers were 10 inches in diameter and the pitch was 4.5 inches.

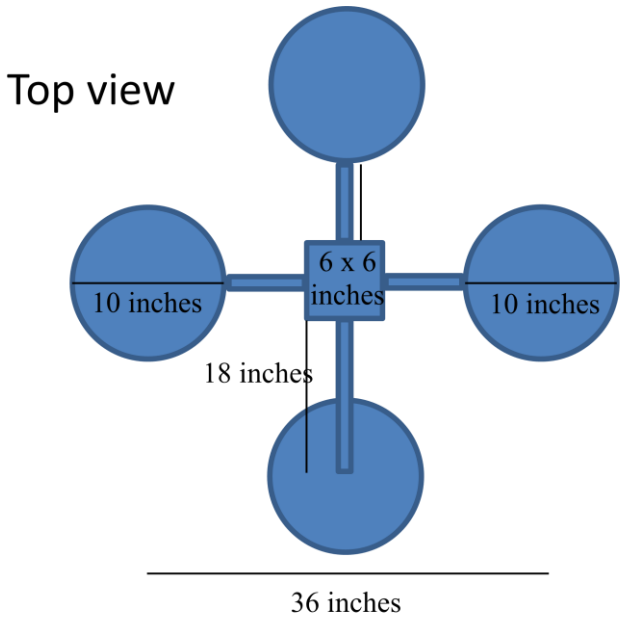

Figure 2.3.1 Quadrotor Dimensions

Figure 2.3.1 shows the dimension set up for the quadrotors, it shows some of the dimensions that were taken into consideration. It all starts with the size of the interface plate; here it is a 6 x 6 square in the middle of the quadrotor. This dimension of the plate depended on the size of the electronics, the size of the PCB, and the sizes of the camera, laser gun, and batteries. It all should be able to fit within this square, so if the physical dimensions of the microcontroller exceed 6 x 6, than a bigger interface plate would have been needed. Once the sizes of the interface plates were established it was possible to choose the length of the arms for the frame. The arm length was important in balancing the aircraft as well as keeping the propellers far enough away from the interface plates and all the wires for the ESCs. The arms had to be at least as long as the propellers radius to insure that the propellers were not hitting anything on the quadrotor, but it was even better to have the arm length to be about twice the radius of the propellers. This provides better balance and steering capabilities when the quadrotors are in dynamic flight. Now the propellers could be chosen, there were two dimensions that needed to be taken into consideration, the length of the propellers and the pitch of the propellers. This all depended on the weight of the quadrotor as a whole, and the power of the motors, a more in depth discussion on this topic will be discussed more in section 5.0.

For the motors, the characteristics that were important were the amps that it will draw, and the Kv rating. The amps matter because the batteries needed to be able to support whatever it drew, and the Kv rating tells how many RPMs/volt,

finding the perfect balance for these two ratings was the key to optimizing the quadrotors performance.

For the camera it there were multiple options available, all of which depended on the quality that was needed for the project. For A.Q.U.A.L.U.N.G. a minimum of 720p was the goal for video resolution. Size of the camera was also a factor, based upon the electronics dimensions the interface plate size affected how big the camera could be.

Laser gun and sensors also needed to be within a certain size limit depending upon the size of the interface plate as well.

In section 5.0 Design Summary of Hardware and Software, exact dimensions for hardware on a prototype will be examined and specified.

3.0 Research Related to Project Definition

Research in the field of robotic automation is expansive, with solutions being explored from every conceivable direction. As such, an exhaustive discussion on even a fraction of these projects is unfeasible, not to mention distracting from the focus of the A.Q.U.A.L.U.N.G. project presented in this report. However, any achievement worth mentioning is built upon the proverbial “shoulders of giants”, this project being no exception. The purpose of the topic at hand is to provide information about a few of the specific projects that have provided direct inspiration and applicable techniques used in the A.Q.U.A.L.U.N.G. project.

The first of these projects is the work of Dr. Vijay Kumar at the University of Pennsylvania. The original inspiration for working with quadrotors for this project comes directly as a result of the amazing demonstration of the possibilities of autonomous quadrotor robots in a presentation Dr. Kumar gave at TED. In this presentation, Dr. Kumar demonstrated quadrotors flying autonomously and performing various acrobatics in a dynamic environment, even flying through a hula hoop tossed into the air. In addition to showing a single quadrotor performing aerial acrobatics, Dr. Kumar showed quadrotors working together to perform various tasks, such as constructing a three dimensional frame. The effect left from viewing this video when it first came out was a desire to work with quadrotors, even if only as a hobby. As it worked out, each member of the group found that they shared this common fascination, and thus the A.Q.U.A.L.U.N.G. project was born. Dr. Kumar's research did not only contribute by inspiring the A.Q.U.A.L.U.N.G. project, but due to the related nature of the projects, it is an excellent source for information and direction.

Another project that has been researched is a previous senior design project at UCF. The Reconnaissance and Demolition Super Attack Tank project described in the Spring 2012 Senior Deisgn 1 class has been researched for the merits of computer vision solutions described therein. The motivation for researching a previous UCF senior design project is that the solutions discovered for a senior design project are designed for implementation. When it gets down to it, these projects give a strong argument for what will work and while it will perform as desired. This removes the task of sifting through research that is only concerned with advancing theory and provides no feasible suggestions for practical applications. Therefore, it is especially vital to verify the findings of previous senior design projects, as they provide that which is sometimes difficult to find in other research: a straightforward description of how to apply the knowledge for practical use.

Another senior design project, titled Multi-Agent System of Quadcopters was submitted in the Spring semester of 2011 at the University of Texas at Austin has been examined. The document submitted is not as thorough in describing the design and justification for

the decisions made as is required by UCF, but still has merit in detailing the experiences of a group working on a similar project.

4.0 Project Hardware and Software Design Details

4.1 Initial Design Architecture and related diagrams

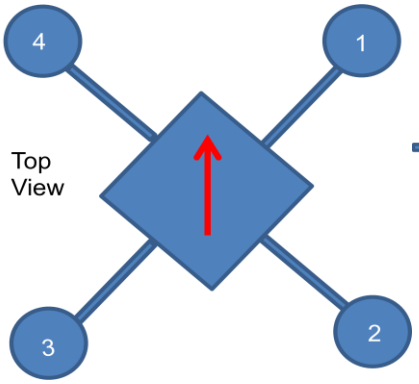

Figure 4.1.1

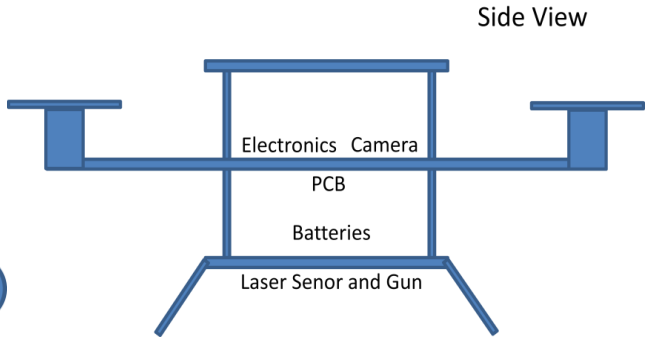

Figure 4.1.2

The architecture of the quadrotors looked like figures 4.1.1 and 4.1.2. Figure 4.1.1 is a simplistic view of the type of flight configuration that was be implemented in A.Q.U.A.L.U.N.G., in non-technical words it is the 'x' configuration instead of the '+' configuration. This type of configuration was chosen because of the stability aspect of the configuration. Visualizing movement is rather easy for this configuration as well, if forward motion is the goal all that is needed is the increase in speed of two motors simultaneously and the decrease in speed of the other two motors. For example, if the quadrotor wanted to move forward motors 1 and 4 would slow down while motors 2 and 3 would speed up causing a tilt forward. This configuration was a little more difficult to code however, it causes more stabile movement and less drastic of movement in the quadrotors flight pattern.

In figure 4.1.3, the flowchart of all four main subsystems in the A.Q.U.A.L.U.N.G. project is displayed. Each subsystem was chosen due to its importance factor in the overall success of the project. All of the subsystems interact with each other in some way or another, whether talking back and forth or just providing power, each subsystem plays a role, and if one were to fail then the whole system would collapse. A complete and thorough discussion about how each subsystem talks to the next will be provided in the subsystem sections but for now, a rough overview will suffice. The power subsystem, of course, supplied power to the

entire system, varying with whatever each component needs. The quadrotors had the flight control (navigation) system and image processor on them, and wirelessly sent two pieces of information to the AI control system. It sent its current position deduced by dead-reckoning and it also sent a video stream. The AI control then had this information and decided what the quadrotor would do from there, if the video saw a threat it would tell the quadrotor to shoot, as well as tell the quadrotor its next position to move to. This cycle continues, making an autonomous laser tag playing team of quadrotors!

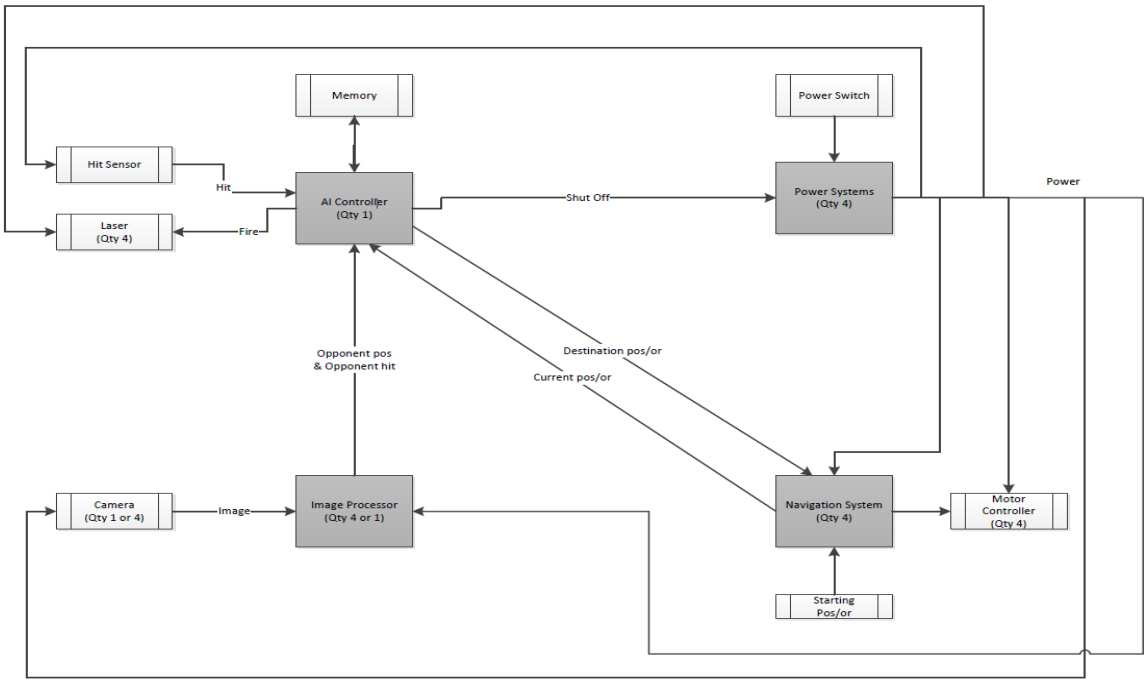

Figure 4.1.3 Main Subsystems Flow Chart

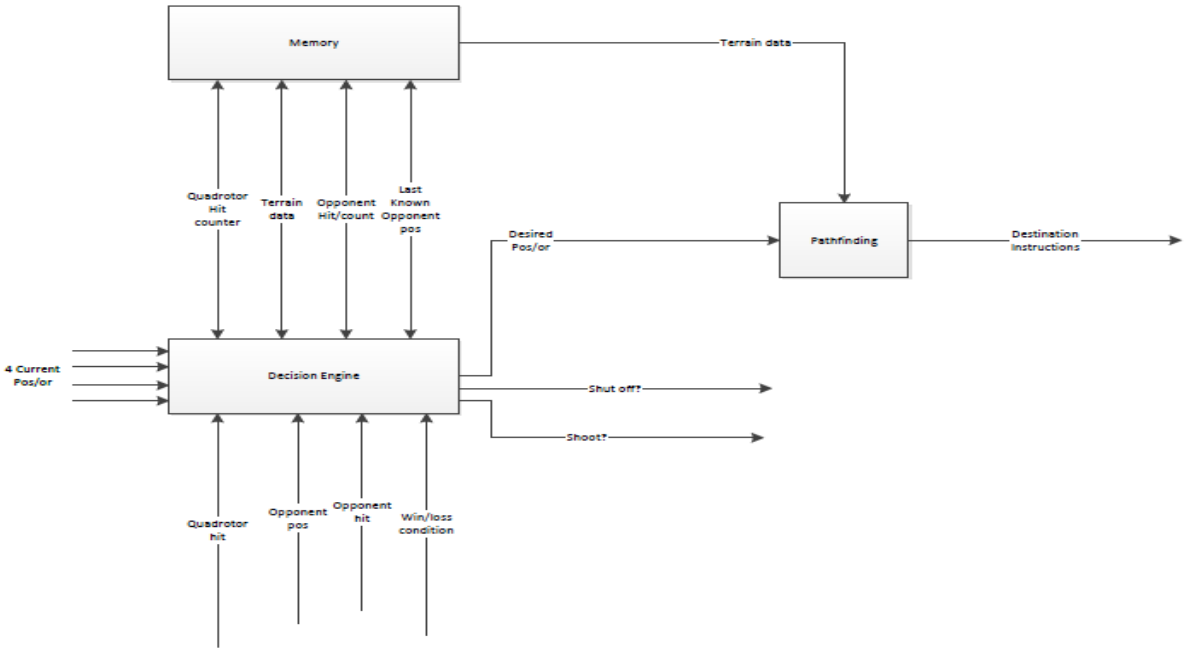

Figure 4.1.4 The AI Controller subsystem, covered extensively in section 4.2.

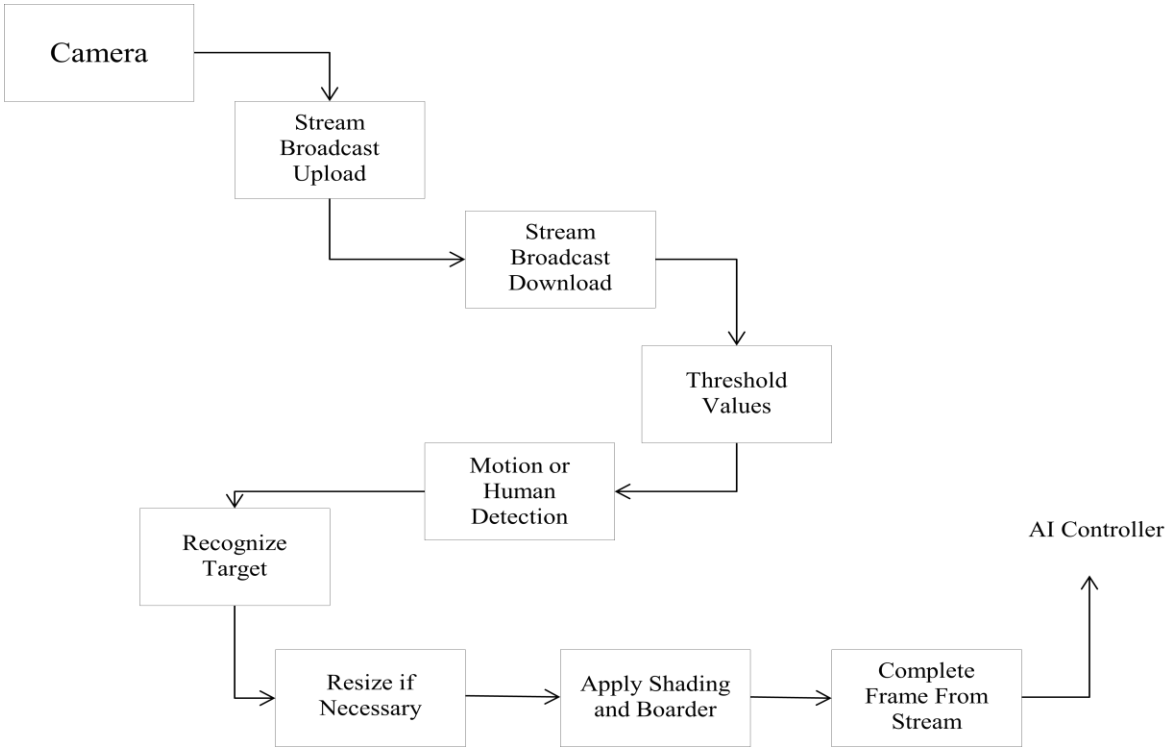

Figure 4.1.5 Image Processing Subsystem Flowchart, covered extensively in 4.5

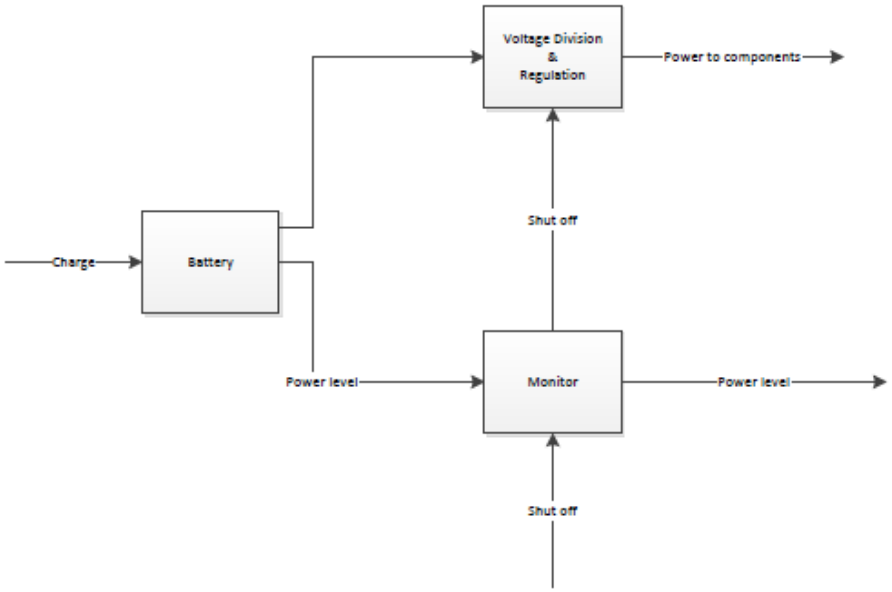

Figure 4.1.6 Power Systems Flow Chart, covered extensively in section 4.3

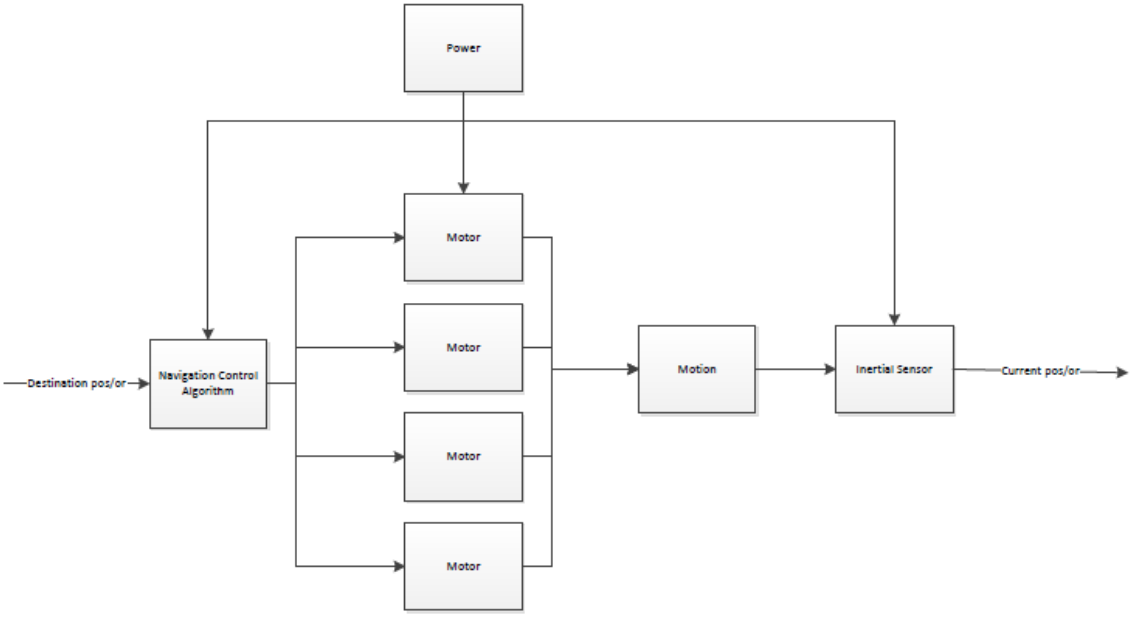

Figure 4.1.6 Flight Control Subsystem, covered extensively in section 4.4

For the AI subsystem in Fig. 4.1.4 the decision making portion is at the heart of it. The positions of the quadrotors would come into the system, the system would then look into its memory and find the best move to make based on multiple factors. It would also receive information from the camera and decide whether to shoot or not and possibly navigate based on sight, if it saw an enemy, it would turn toward it/ follow it and attack.

For the image processer, an image would be sent from the camera to the main system for detection, once motion was detected within the video stream it decided if it was human or not, if it was it would mark this as a target, it would do some conversions, and send out the stream to the proper authority on executing an attack.

The power system is a quite simple system that just disperses power throughout the whole quadrotor, to all its vital components. It starts at the batteries, goes through a printed circuit board where the proper conversions were made, either regulating, amplifying or not changing the voltage at all. Once the correct voltage was established it was sent out to all the parts.

The flight control subsystem took readings from the Inertial Measurement Unit and converted them into viable readings in order to stabilize the quadrotors as well as determine the quadrotors position using dead-reckoning. Once the position was established it sent the information to the AI and got back the next position, controls the motors and moves the quadrotors to the new position.

Just some quick overviews of the four different subsystems with flow charts to go along with them.

4.2 AI sub system

4.2.1 Introduction

The Artificial Intelligence subsystem was responsible for every decision made by the A.Q.U.A.L.U.N.G. project. Where to move, whether to shoot, how to evade, when to chase, who to aim at, what to do! The AI system needed to provide an appropriate answer to every one of these decisions. Each answer needed to be made intelligently. And most importantly, the Artificial Intelligence needed to make these determinations quickly. To add to this burden the artificial intelligence had an extraordinarily small

amount of data to work with, as compared with the data which the senses of human beings provide. In fact, each quadrotor only had one of the five senses, sight, and only barely at that. If this page is held 20 inches from the eye, the pixel equivalent size of this sheet is roughly 37 megapixels (Clark, 2009). Considering the capabilities of the human eye, it is fair to say the quadrotors were essentially blind as well. Therefore the A.Q.U.A.L.U.N.G. system needed to be able to make decisions with hardly any vision and only an educated guess of position, and have these decisions compete against human beings. An ambitious task to be sure, but nothing has ever been accomplished by backing down from a challenge.

To accomplish the task of competing against human opponents, the AI subsystem was divided into two layers: the high level decision-making, and the low level calculations. An example of these two layers interacting can be shown with the action of moving from point A to point B. First, the upper-level grabbed information from a low-level tactical evaluation of the environment that was always running. Then, the upper-level layer decided which of the suggested locations to move to. The upper-level sent this target location to a path finding algorithm, which returned the best path from the current location to the target location. This path was given to the quadrotor, which then moved to the target location. Approaching the design of the system, this manner lent itself to the design of modular code that could be independently debugged if irregularities were discovered.

It is worth noting that many of the solutions to the various Artificial Intelligence problems the A.Q.U.A.L.U.N.G. project faced are primitive in nature. If ample resources, the most important of which being time, were available, many of these solutions would have been replaced or enhanced by machine learning techniques such as evolutionary computation or neural networks. This was unfortunately not feasible given the time-scale and resources available, as machine learning techniques require an abundance of data to sort through and adapt to, as well as time to test and learn.

Lastly, the purpose of this design documentation is not to teach programming or proper coding techniques. Therefore, the specific means of the implementation of a solution is not described. Instead, the purpose is to present the available solutions and the reasons for or against having implemented any given solution.

4.2.2 Path Finding

In any Artificial Intelligence application in which there is interaction with the world, whether real or virtual, path finding is arguably to most crucial aspect. In the application of this project, a failure in path finding would have substantially more devastating results than the failure of any other Artificial Intelligence component. If the quadrotors were unable to evade an opponent, they were simply eliminated from the competition once they are defeated. Similarly, if they were unable to acquire a target and defeat

opponents, they would merely be unable to win the competition. However, if the quadrotors could not find a safe path through the environment, they would almost certainly crash into an obstacle causing damage to themselves, and, if that obstacle happened to be a human opponent, they could have caused injury to that person.

While there were a few possible alternatives, the path finding algorithm that was used in this critical application is the A* algorithm. A* is a best-first search method commonly employed in path finding solutions. (Lester, 2005) The algorithm sorts through a given representation of the area through which the path must be found, and returns the path with the shortest distance. The algorithm employs a heuristic to improve its performance, but the heuristic must be chosen carefully, as underestimating makes the algorithm less efficient, and overestimating can cause it to not return the shortest path. (Lester, 2005) There are three common representations of the environment that are used with the A* algorithm: a grid representation, a way point graph, and a navigation mesh. To demonstrate the general differences and capabilities of each representation, they have all be applied to an example environment, shown in Figure 4.2.2.1.

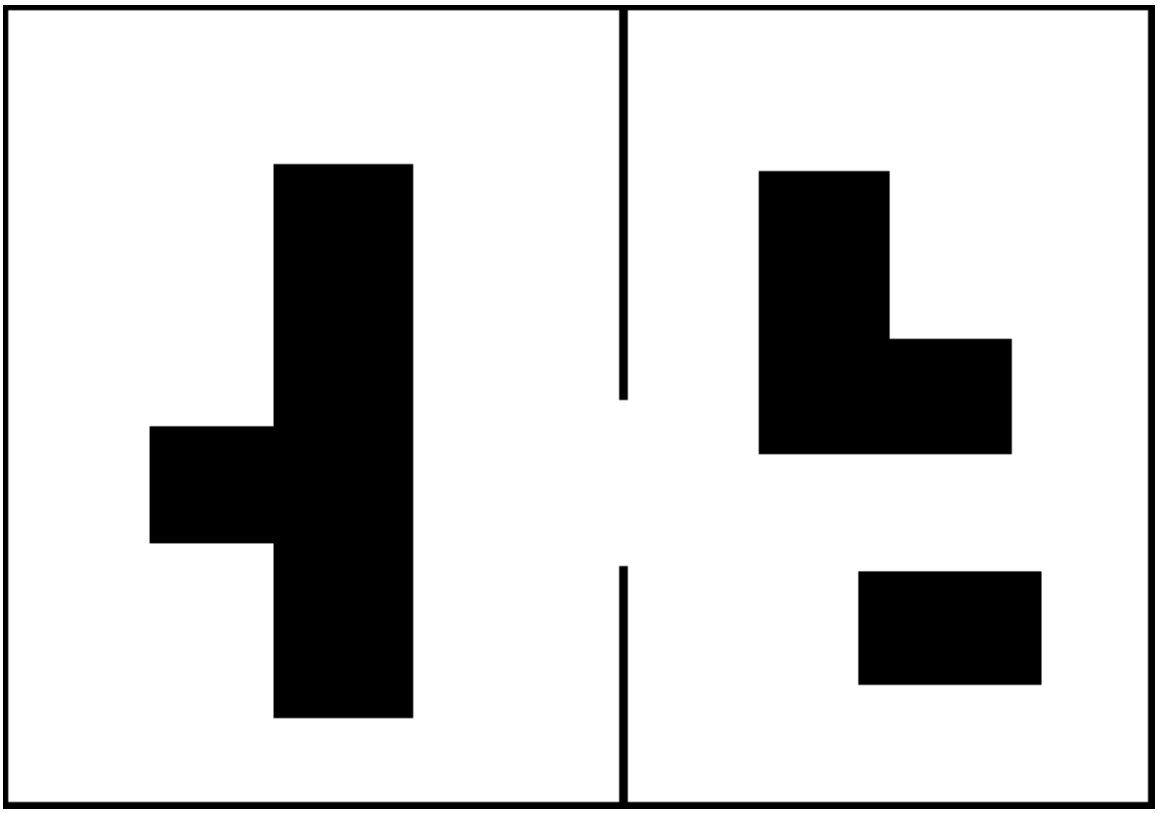

Figure 4.2.2.1: Simple Example Environment

For the sake of simplicity the example environment is shown in two dimensions. This allows for clear demonstration of each representation on the environment. None of the underlying principles of the A* algorithm are altered by the presence, or lack, of a third dimension.

An interruption feature has been included, primarily to avoid collision with a human opponent if they moved into the path of the quadrotor. The quadrotors were able to be given an updated path before they finished following a path they were previosuly on; the system remained dynamic at all times. This allowed for the quadrotors to safely navigate in the presence of dynamic obstacles, as well as update their position objectives in response to the behavior of their opponents.

A grid representation would have laid a grid over the entire area and label each resulting cube as either clear or an obstacle. The center of each cube would have been denoted as a node. A path would then be decided on from one node on the grid to the next, including diagonals. This method of representing the environment is shown in

Figure 4.2.2.2.

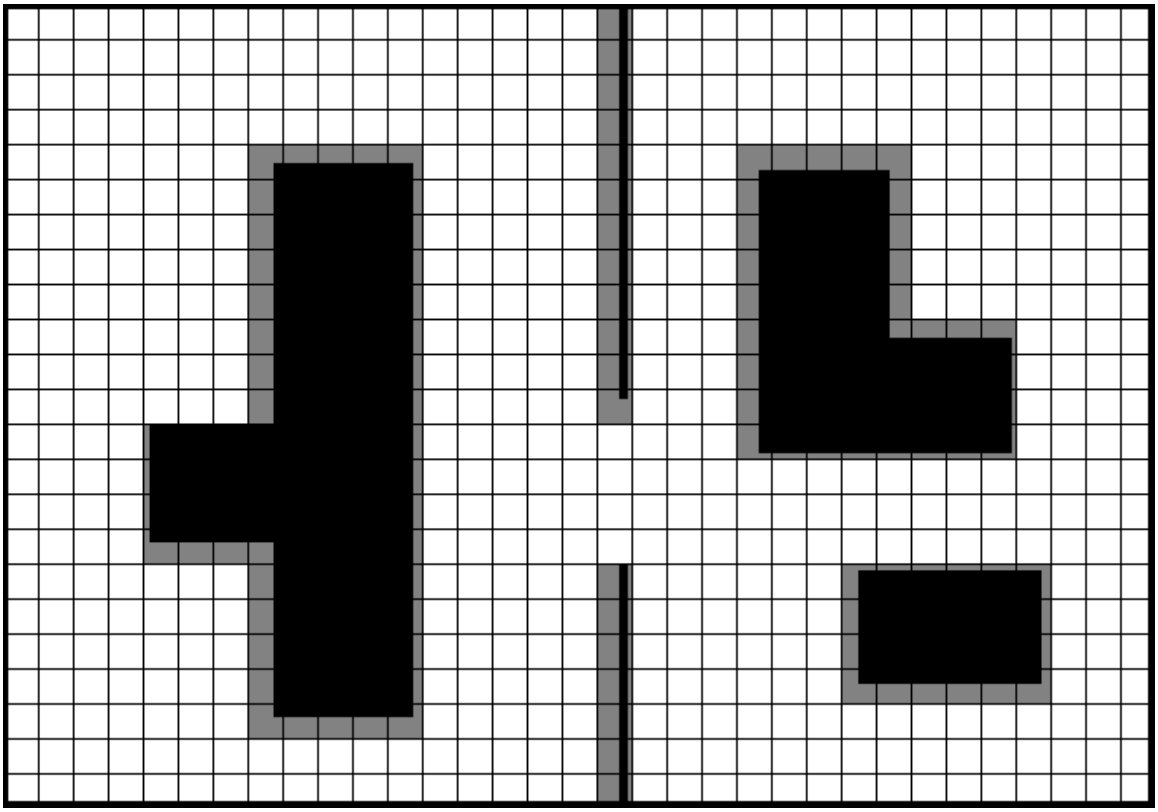

Figure 4.2.2.2: Grid Representation of Example Environment

Every square (or cube in three dimensions) that is partially obstructed by an obstacle is considered to be entirely an obstacle, shown by the gray colored squares.

The heuristic for the grid representation would not have been simply the Euclidean distance from the evaluated node to the target node, as movement within the grid would have been limited to adjacent nodes. This means that all motion would have happened along one of the three cardinal directions or a diagonal that results from an even combination of two or three of these directions. As such, the heuristic used would have been the sum of three values. The first value would have been the length of the least traveled cardinal direction multiplied by the square root of three. The second value would have been the difference between the lengths of the median and least traveled directions, multiplied by the square root of two. The final value would have been the difference of the length of the most traveled direction and the length of the median direction. This heuristic represents the minimum possible distance from the starting location to the target location if no obstacle had been present. An advantage to using a grid would have been that every location in the environment would have been accounted for and could have been navigated to, if the location to be navigated to hadn't been an obstacle itself. The most significant disadvantage to using a grid representation would have been the abundance of data, and thus reduced speed of searching through the space to find the best path. Consider a room that is 30 feet by 20 feet by 10 feet, and lay out a grid with 3 inch nodes. This will generate 384,000 nodes. In a worst case scenario, over half of these would have been examined, and many of them would have been examined more than once. Before any of the quadrotors would have been able to move, they must have had a path to follow, which means that they would have sit idle until a solution had been found. To counter this, there are a few techniques to either speed up the process itself, or reduce the search space. The first trick that can help with the problem of a large search space would have been to create two grids. (Lester, 2003) The second grid would have been composed of much larger cubes, thus having a greatly reduced search space. First, a path would have been found from the starting node to the ending node on the large grid, and then a path on the small grid would have been found, but only from the starting point to the large node two steps further on the first path. This micro-level path would have then been updated every time a new macro-level cube had been entered. This solution very nearly creates a navigational mesh, which will be discussed later. The next technique that can be applied to the problem of scale would have been the use of a binary heap to store the list of nodes to be searched. (Lester, 2003) A binary heap is a method of organizing all of the possible nodes into the list of nodes to be searched, such that the first item in the list is the most desirable next node to be examined. The way a binary heap works is to essentially heap every node into a pile with the condition that every node above the bottom row has two children, and is more desirable to the A* algorithm than either one. Figure 4.2.2.3 and Table 4.2.2.1 provide an example binary heap representation, and the associated array.

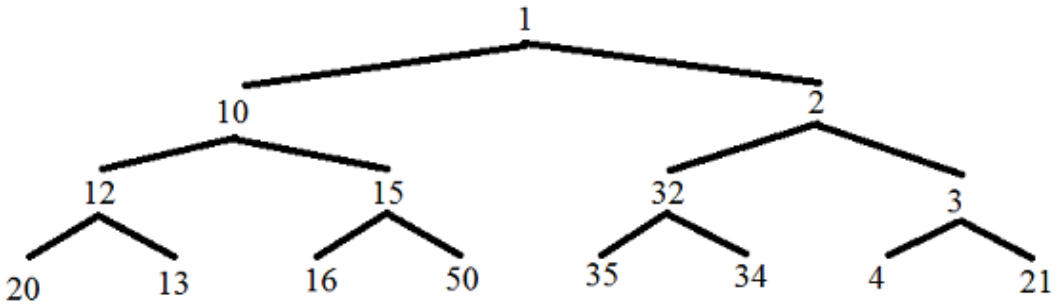

Figure 4.2.2.3: Binary Heap Example of F scores for A Algorithm*

Lower F scores are more desirable to the A* algorithm, the every parent is lower than either child. This example is one of many possible heaps for the randomly selected values.

Table 4.2.2.1: Binary Heap Example Array

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value	null	1	10	2	12	15	32	3	20	13	16	50	35	34	4	21

Values populate the array in increasing position when read from top to bottom, left to right from the heap. The positions of the two children are the parent's position multiplied by two, and this value plus one.

A binary heap would have also come with methods for re-organizing. These methods would have been called upon every time a node is added to the open or closed lists used by the A* algorithm, and are briefly described below. A binary heap could have been used in any representation of the environment used for the A* algorithm, but it would have had the most drastic effect in large search spaces. The phenomenon of exponentially diminishing returns with diminished size arises from the nature of the binary heap's structure and means by which it is organized. Every time a node would have been moved within the heap, it would have been compared with each successive parent or child, depending on the nature of the reorganization, until it would have been more desirable than either child, but less desirable than its parent. Specifically to add an item to the heap, it would have been placed at the end of the array and moved into the proper position. When the first item would have been removed from the heap, the last node would have been moved into the first position, and then moved into the proper position by successively swapping places with the most desirable of its children. Because of this, the maximum number of steps to find the correct place for a node would have been related to the base two logarithm of the size of the heap. An example of a path found through a grid representation is shown in Figure 4.2.2.4.

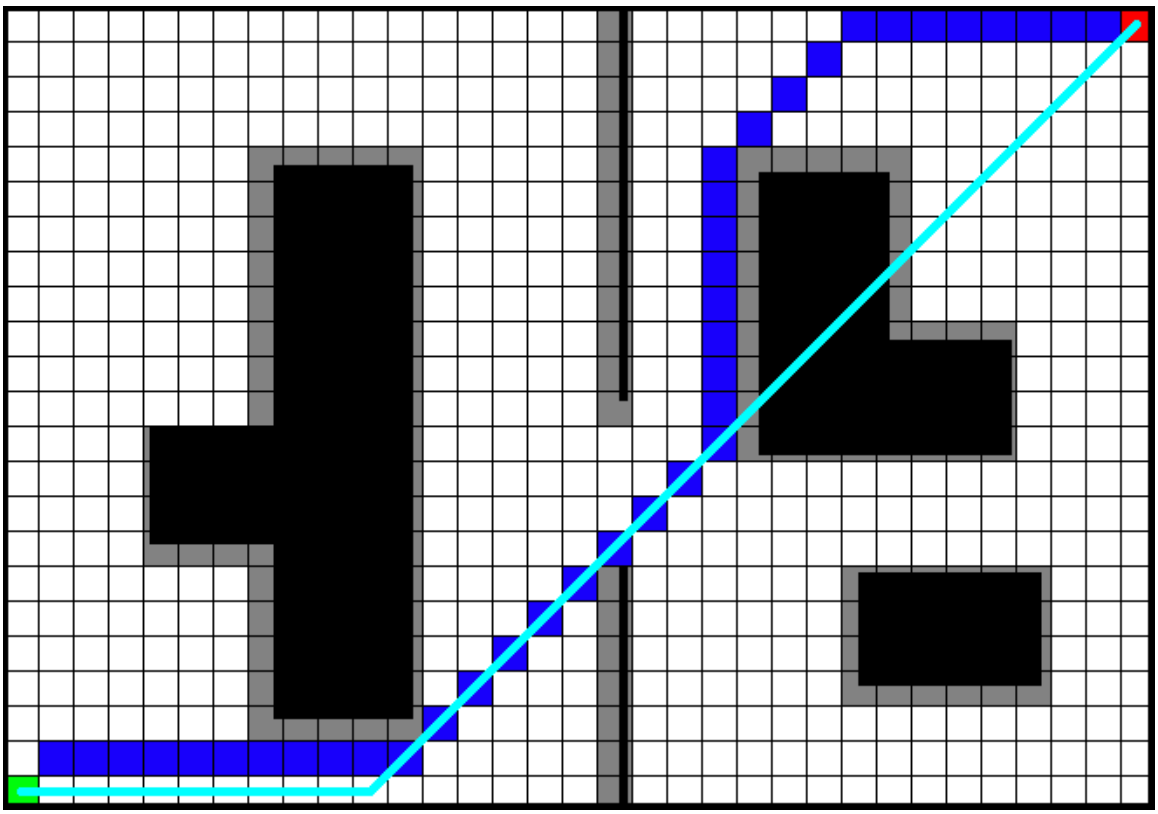

Figure 4.2.2.4: Example Path Using a Grid Representation

The green square (bottom left) is the starting location. The red square (top right) is the target location. The blue squares are the path returned by the A* algorithm. The teal line is a representation of the heuristic for the starting location (although the heuristic would not actually be calculated for the starting location).

The second possible representation of the environment that can be given to the A* would have been a way point graph. This representation would have required more work on the front end by the developer, as they would have been required to specifically place each node in the environment, at any location desired. Each node would have then been connected to its neighboring nodes by a straight line so long as that line did not pass through any obstacles. These lines then would have designated the paths for the A* algorithm to search. (AI-Blog.net, 2008) Thus, each node would have been assumed to be a valid path, so it would not have been necessary to specify whether or not it had been an obstacle, as required in the grid above. Instead, each node would have stored information about each line it would have been connected to. An example for this representation is shown in Figure 4.2.2.5.

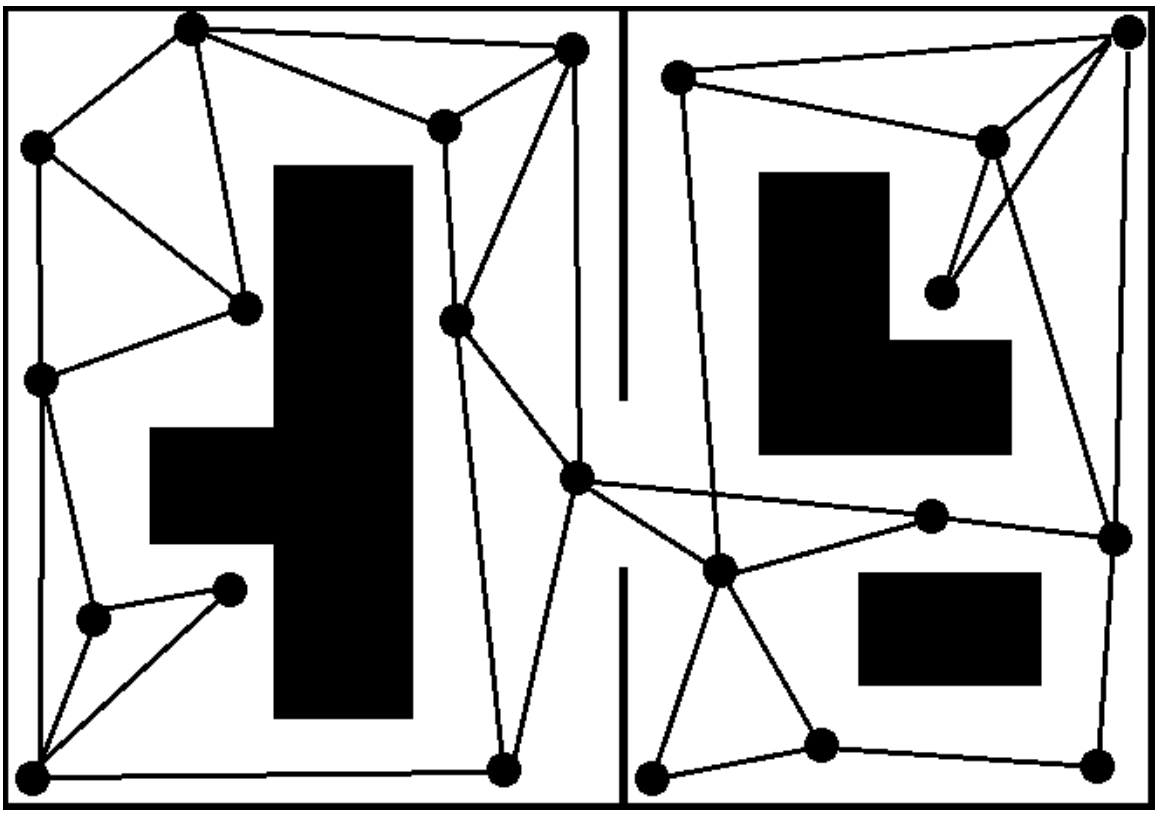

Figure 4.2.2.5: Way Point Graph Representation of Example Environment

This is a non-optimized example of a way point graph for the example environment. A proper model would have the nodes placed for better coverage, but the purpose of this demonstration is to show the general form of a way point graph.

The heuristic for this representation would have been merely the Euclidean distance from the node being evaluated to the target node. This heuristic, as before, would have represented the minimum possible distance from the evaluated node to the target node if a straight line could have been followed. Way point graphs would have allowed, and required, the developer to make choices between speed and flexibility. On one end of the spectrum, a way point graph would have become identical to the grid just described, with the added ability to connect to more than just nodes directly adjacent to it, but with more information, and thus space used. Additionally, an increased number of paths per node would have meant an increased amount of processing during each iteration of the A* algorithm, because the difficulty in solving a path finding problem has little to do with the length of the path, but rather lies within the number of possible paths to take. While doubling the complexity of the search space may have only cost a small amount of extra time per iteration, it could have added up to be a substantial increase with over 300,000 nodes, as in the example above. This burden would have been lessened with the use of a binary heap, but not done away with completely. On the other end of the complexity spectrum, a way point graph could have been used to basically draw a circuit through the environment. In this case the only decision to be made would have been which

direction to go around in, and no matter how long the circuit is, the decision could have been made almost instantaneously. Regardless of this speed, this level of complexity would have had little practical use in a dynamic combat system, as it would not have allowed for flexible, fluid movement through the environment. The last flaw that had been found with a way point graph was the fact that it has no built in method to avoid collision between two entities moving along converging paths, because the only information the A* algorithm has about the environment is the paths between nodes. This means that having one move to the side could have caused it to crash. While the issue of converging paths was unlikely given the movement strategies of the A.Q.U.A.L.U.N.G. project, it was not altogether impossible in a highly dynamic environment, such as laser tag combat. The easiest solution to this conundrum would have been to ensure that every line always has enough clearance in at least one direction to allow a quadrotor to stop and move aside for another and then get back on the path and resume its course. The advantage of a way point graph would have been the ability to meet in the middle of the complexity spectrum, to place and connect nodes in such a fashion that the quadrotors would have been able to move dynamically through the environment, but reduce the search space from that of a complete grid layout. Additionally, the complexity could have been adjusted based on predicted hot spots, adding more nodes and paths in areas that are predicted to experience frequent use and require flexible movement, and creating fewer nodes and choices in areas that are not anticipated to be used very much. An example of a path through a way point graph is shown in Figure 4.2.2.6.

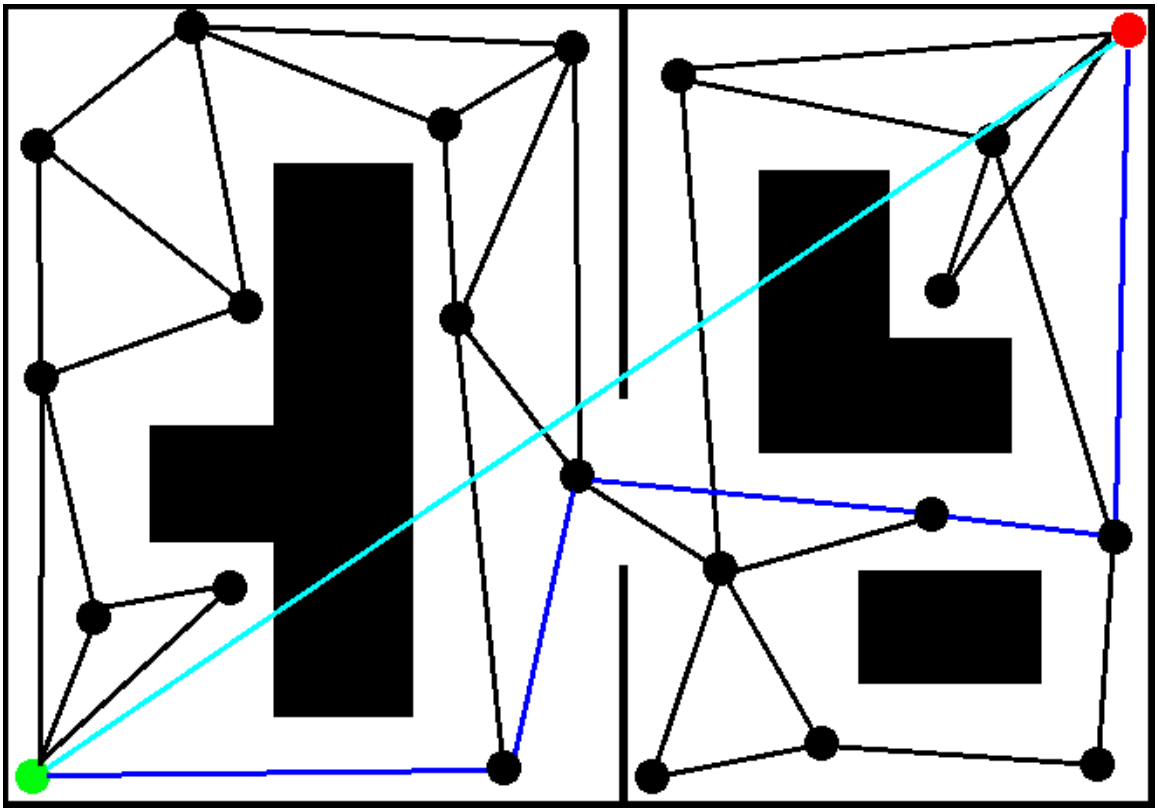

Figure 4.2.2.6: Example Path Using a Way Point Graph Representation

The green node (bottom left) is the starting location. The red node (top right) is the target location. The blue line segments represent the path found by the A* algorithm. The teal line is the heuristic for the starting node (although the heuristic would not be calculated for the starting location).

Having saved the best for last, it is time to discuss navigation meshes. A navigation mesh allowed for a description of the parts of the environment that can be moved in freely by populating it with convex polygons. (AI-Blog.net, 2008) Each polygon stored information about its vertices and the neighboring polygons that can be traveled to. The center of each polygon was a node that the A* algorithm will use as a starting point for creating the path to follow, but since information was available describing all moveable areas, the path could be improved, creating the most direct possible path through the environment. An example of a navigation mesh representation is shown in Figure 4.2.2.7.

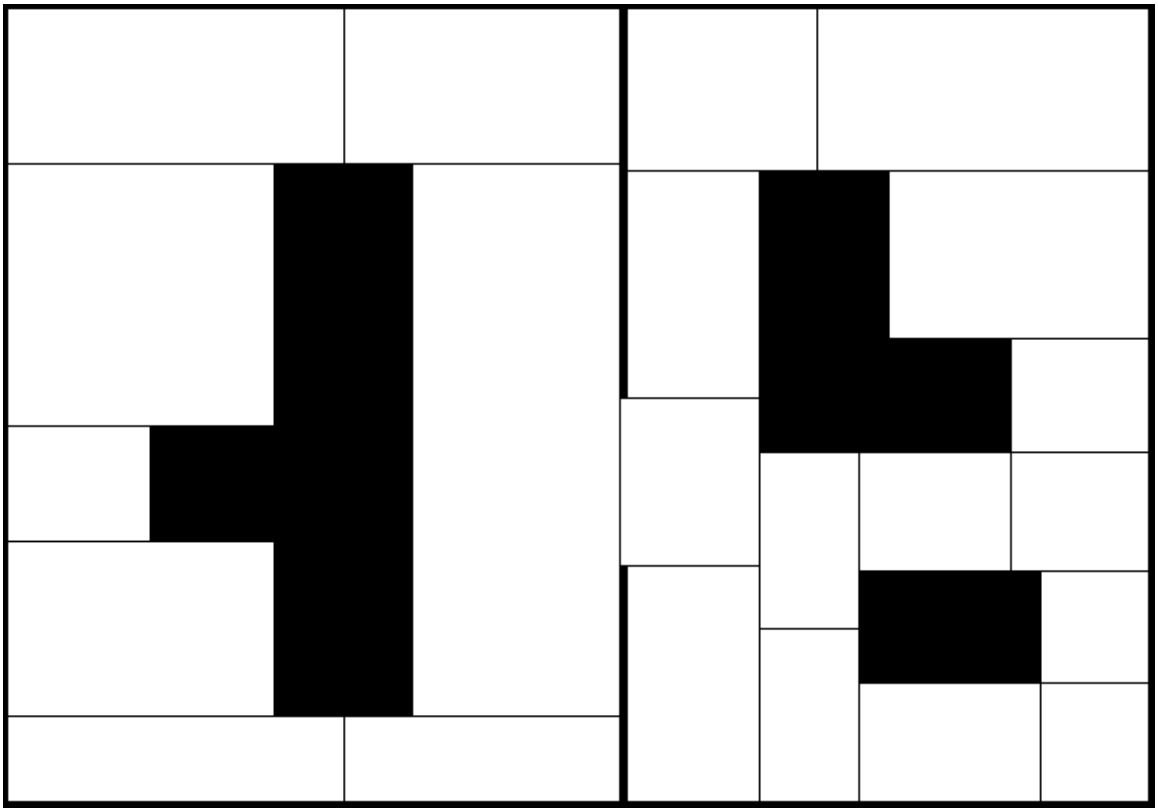

Figure 4.2.2.7: Navigation Mesh Representation of Example Environment

As with the way point graph example, this is not an optimized example. The purpose of this demonstration is to show the general idea behind a navigation mesh

As with a way point graph, the heuristic used was the Euclidean distance from the evaluated node to the target node if no obstacles were in the way. As well as allowing for improved efficiency in the paths, the use of polygons to represent large chunks of the environment meant that the search space was reasonably sized, allowing for rapid determination of the path to use. The positive aspects of using a navigation mesh to represent the environment made this method very attractive. The downside lied within the complexity added with creating more efficient, direct paths. The processing time is not really a concern, as altering the path only involved a small number of nodes once the A* algorithm had finished running. Rather, it required the developer to design a final step that wrapped the path around any obstacles and draw straight lines as often as possible. The method for this step that has been used in the A.Q.U.A.L.U.N.G. project is to draw a straight line from the starting point to the ending point, and then shrink the path the A* algorithm returned to this line. The result was the best possible path that could have been found using line segments. A question that could be asked at this stage is, “Why not take the path finding one step further by smoothing it with a spline function?” The answer to this question was that the flight control algorithms on board each quadrotor handled creating a smooth path once given the set of coordinates defined by the path

generated. An example of a path found by the A* algorithm using a navigation mesh is shown in Figure 4.2.2.8.

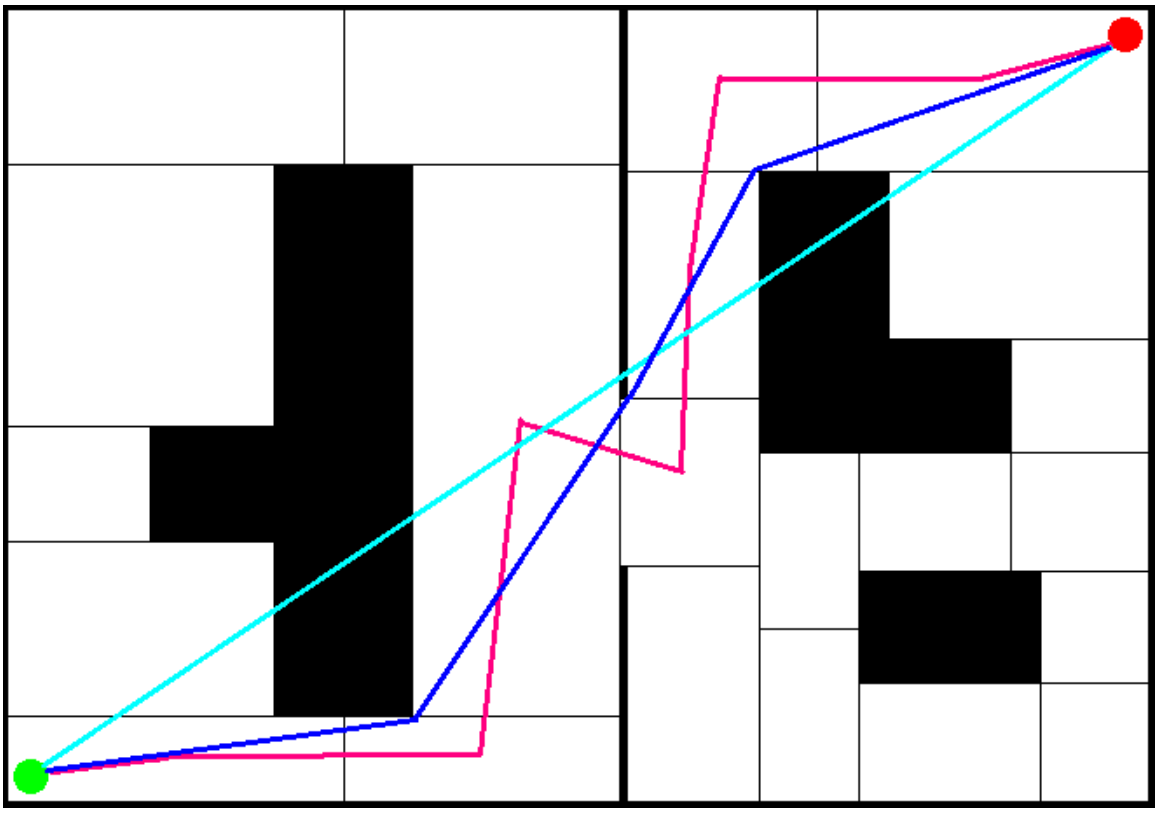

Figure 4.2.2.8: Example Path Using Navigation Mesh Representation

The green circle (bottom left) represents the starting location. The red circle (top right) represents the target location. The magenta line segments represent the raw path returned by the A* algorithm. The blue line segments represent the optimized path. The teal line represents the heuristic for the starting location, which is also the line used for optimizing the original path.

Once all of the positive and negative aspects had been examined, the most desirable representation of the environment was the navigation mesh. To further illustrate this point, Figure 4.2.2.9 compares the final paths returned for each representation.

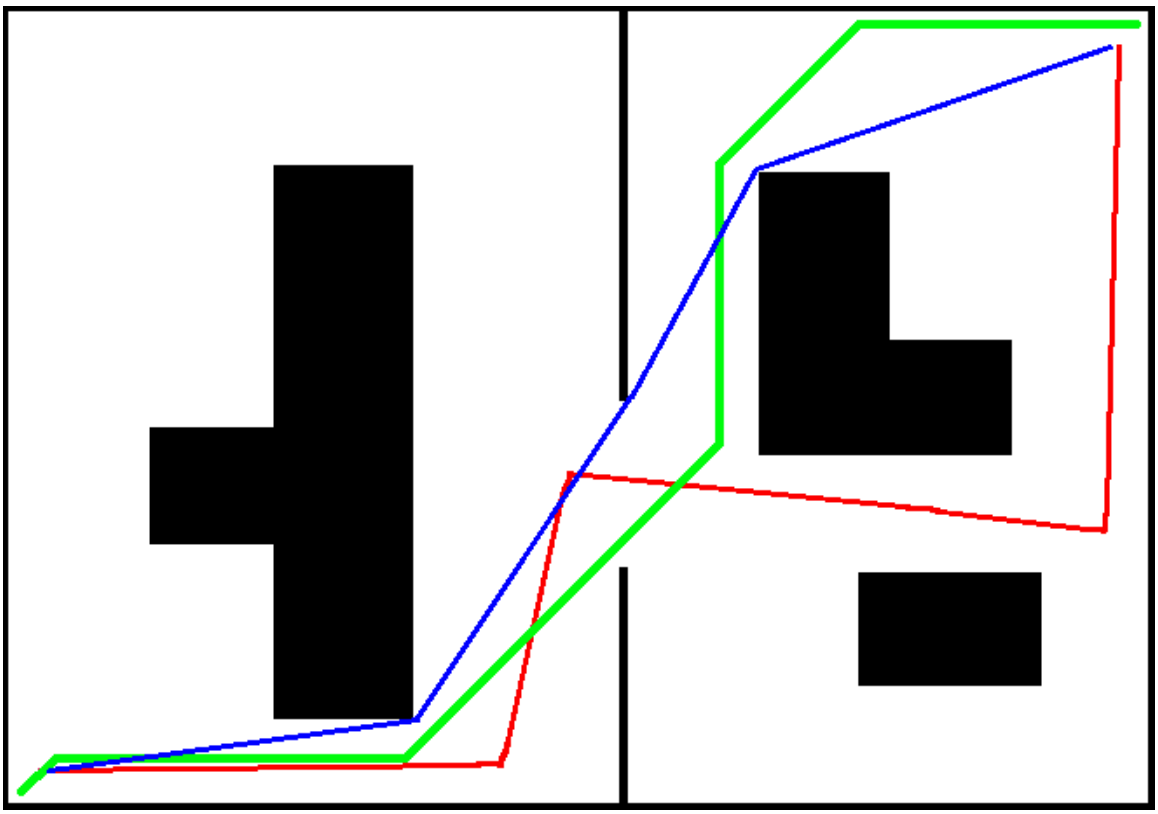

Figure 4.2.2.9: Comparison of Example Paths for Presented Representations

The green line is the path found using a grid representation of the environment. The red line is the resultant path from the way point graph. The blue line is the final path obtained from the navigation mesh.

As can be seen, the navigation mesh path was the most direct of the three options. Not only did this representation allow for the most direct path, it also worked extremely efficiently, having a small number of nodes for the A* algorithm to search. While it can often be difficult to predict with any certainty as to what method will yield the best results when dealing with complex real-time applications, the decision to use the A* path finding algorithm with a navigation mesh representing the environment is made was a high degree of optimism, and yielded efficient, desirable results.

4.2.3 Tactical Planning

The key ingredient to the Artificial Intelligence system in the A.Q.U.A.L.U.N.G. project in being able to compete against human opponents was the ability to think tactically and plan ahead. Any system that is incapable of this task and merely reacts to the observed actions of the human opponents can be easily beaten, once the pattern of reaction is discovered. A purely reactive system can be exploited in a variety of ways, and human beings tend to discover means of exploiting artificially intelligent system rather quickly.

One example could be the use of one player acting as a decoy to lead the team of quadrotors into a trap. A purely reactive system would have no apprehension about blindly following one opponent without any knowledge of the whereabouts of the others. Of course there are many other possible exploitations that human players could discover. In a virtual environment, developers can side-step these exploitations to some degree by giving the artificial combats an unfair advantage in the form of omniscience of the virtual environment and all players within, mathematically precise aiming abilities, reflexes faster than any human being, and the list goes on! The A.Q.U.A.L.U.N.G. project could not take these liberties. The quadrotors were operating in the real world, interacting with real physics, and had limited knowledge about the environment, just like the human opponents. The playing field was even, and thus the quadrotors needed to make intelligent decisions and plan ahead, attempting to anticipate the future actions of the human combatants, not merely react to the present and past actions. Unfortunately, as mentioned earlier, the use of machine learning techniques was unfeasible within the scope of the A.Q.U.A.L.U.N.G. project due to time and resource constraints.

One method to increase the tactical capabilities of the team was to give it an agenda of its own to follow. This would allow the team to be able to take action in the absence of knowledge of the actions of the human opponents. One method for coordinating the team's goals was to designate a specific control point in the environment. Taking a lesson from chess, wherein having control of the center of the board is a significant advantage, these control points represented regions of the environment that were desirable to keep as much control of as possible. An example of this can be show in the same environment used to discuss path finding, shown in Figure 4.2.3.1.

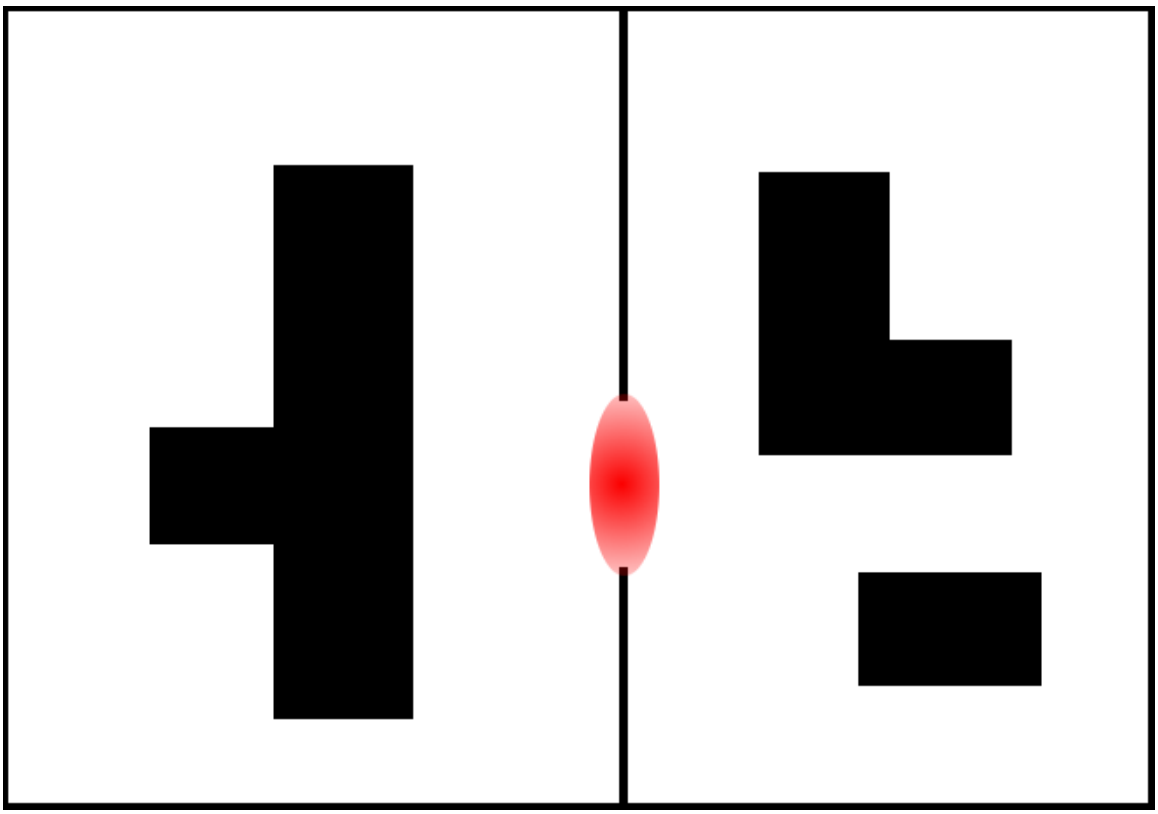

Figure 4.2.3.1: Possible Control Point in Example Environment

The red region is a possible control point in the environment. Control of this region can give a significant advantage to either team.

To maintain control did not mean the quadrotors needed to reside within the control point, but rather, as a Bishop provides control over the center of a chess board from the far corner, maintaining control implied having a superior ability to attack any opponent found within the far or near region of the control point. Consider the example shown in the figure above. The control point represents the only path to cross from one side of the environment to the other. Having control would then restrict the opponents' movement within the environment, as in order for them to cross from one side to another they would have to expose themselves, leaving them vulnerable to attack. Of course, this particular example is more likely to end in a stalemate as the geometry of the environment is such that both sides would be able to thoroughly cover the entryway, stacking the odds heavily against any attack action. Even though the geometry of the environment in the example is highly conducive to a stalemate scenario, and as such maintaining control of the control point is not likely to produce a win, failure to maintain a strong presence would provide an opening for the human combatants to pass through the control point safely and thus corner the quadrotors.

A more likely scenario for combat would have a more complex environment, producing multiple control points. Additionally, in a complex environment, one that allows for a flow of combat through the arena, as opposed to a standoff on either side of a wall, certain control points may vary in significance, becoming pivotal to the outcome of the combat, or conversely becoming altogether unimportant. In this kind of environment one of the many competing goals for the AI was to maximize the number and value of control points the quadrotor team can maintain.

Another goal of the AI for the A.Q.U.A.L.U.N.G. project was to predict the actions of the enemy players. This primarily took the form of predicting an opponent's location, a vital piece of information in combat, simulated or otherwise. Unfortunately this task was nearly impossible to do with any accuracy until an opponent was spotted. This means that until an opponent was spotted, the most desirable course of action was to control as many control points as possible simultaneously. Once an enemy is spotted their future positions can be guessed based on a starting position and last known direction of movement. A probability field was then distributed across the environment denoting the likelihood of the opponent being found in a given region. Figure 4.2.3.2 gives a rough idea of how this worked.

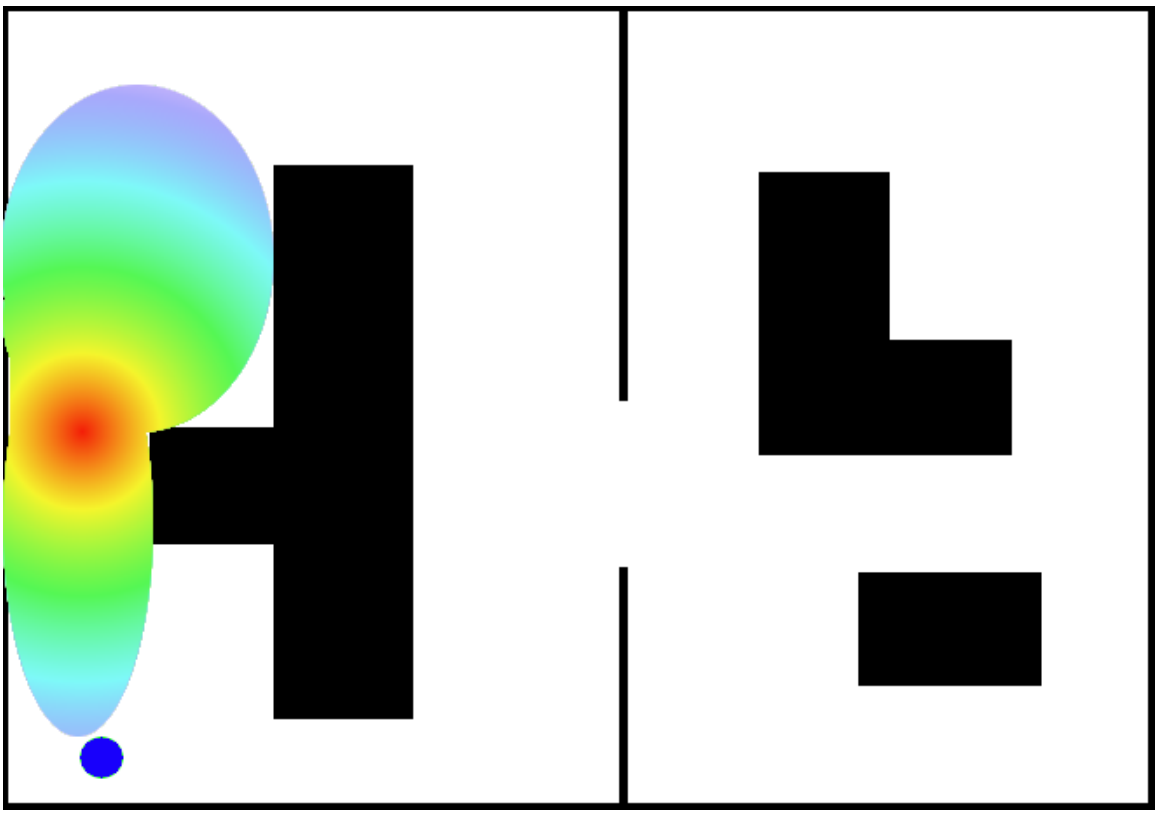

Figure 4.2.3.2: Visual Approximation of Probability Field

A human opponent is seen at the solid blue circle heading toward the top of the example environment. From this information the opponent is expected to be within the colored region, the probability of being in a specific location varies with color, red being the most likely. It is important to note that this is purely a rough graphical representation to give a general idea of how the system functions. No actual measurements or calculations were performed in this example.

Some care must be taken when employing the use of a probability field as, if unconstrained, it can provide information that is virtually useless and thus just waste processing power. A probability field expands over time to determine the probability of the opponent's location. After a relatively short period of time, the probability field becomes so large, and the probability of finding the opponent in a specific region so low, that it is no longer of any benefit. Additionally a probability field has no means for determining whether or not a human opponent would take a branching path through the environment, so each branch encountered on the path further halves the effectiveness of the probability field. How to respond to the information provided by a probability field was highly contextually dependent, as an intelligent course of action under one set of circumstance may have been akin to forfeiting under another. Three key factors in determining how to respond to a probability field were the cost of investigating, the level of awareness of the other human opponents, and the level of awareness of the area to be investigated. For example, if an opponent was expected to be in a region that was

known to be free of other opponents, attempting to intercept and eliminate the lone combatant was desirable. If, however, the region the opponent is expected to be in was not known to be safe, and the whereabouts of other human combatants was not known, intercepting the opponent was a greater risk, as it could have been a trap.

The final area of interest that was examined within the environment is a safe spot. Occasionally the situation would get out of hand and the most reasonable course of action was to fall back and regroup. In order to facilitate this tactic, various regions within the environment were tagged as cover zones, under various conditions. Some regions were a corner to back into for a moment to escape fire, providing cover from many angles. Other regions were a good point to fall back to because they provided a better firing position, such as right past a control point that funneled the opponents. These regions were not places of immunity, merely places that may have provided a brief respite when needed. Figure 4.2.3.3 shows a few safe zones of different types in the example environment.

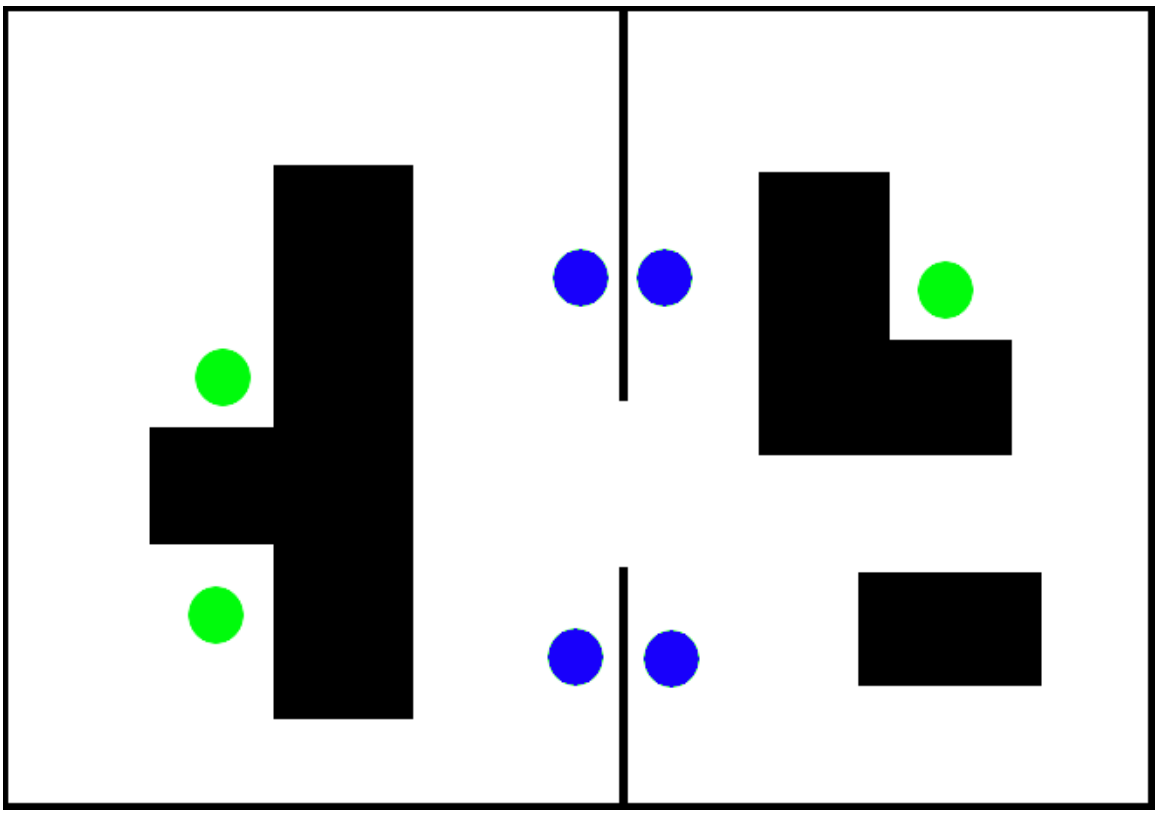

Figure 4.2.3.3: Potential Cover Zones in Example Environment

The green circles represent the first variety of cover zone described, providing protection from fire from a wide range of angles. The blue circles represent the second variety of cover zone discussed. These zones are fairly exposed to fire, but an opponent stepping through the pathway to chase would be even more so.

Prepared with information about the environment describing valuable regions to have vision of, and zones to escape to if the battle isn't going well, and a method to predict the positions of human opponents once vision of them was lost. The environment was then scanned and the local maxima returned for the valuable locations. This equipped the AI subsystem with the ability to behave tactically and choose target locations for each quadrotor when out of combat.

4.2.4 Decision Making

The decision-making aspect of the AI subsystem was the high level control that bound together and directed the actions of all of the other functions and algorithms contained within the subsystem. This part of the AI looked at all of the data presented from the other subsystems, as well as data returned from within

the AI subsystem, and then sorted through this information for the most relevant bits and used these to decide on the next course of action for each quadrotor. The high level decision-making technique for the AI in the A.Q.U.A.L.U.N.G. project that has been employed is a customized variant of a state machine.

Before an explanation of the unique nature of this state machine will have any significant meaning, the typical operation of a state machine must be understood. A state machine typically is composed of a handful of differing states, each one defined by the actions performed during the state, and the switching conditions to other states. When in a state, the actions will loop endlessly until the conditions occur to switch to another state, which will then loop its actions. This allows for a multitude of behaviors to be designed and run with explicit conditions when one behavior switched into another.

The primary change that has been employed with the use of a state machine in the A.Q.U.A.L.U.N.G. project is that every state will perform the same actions. The difference from one state to the next lies in the priorities. Figure 4.2.4.1 shows a representation of the A.Q.U.A.L.U.N.G. decision-making state machine.

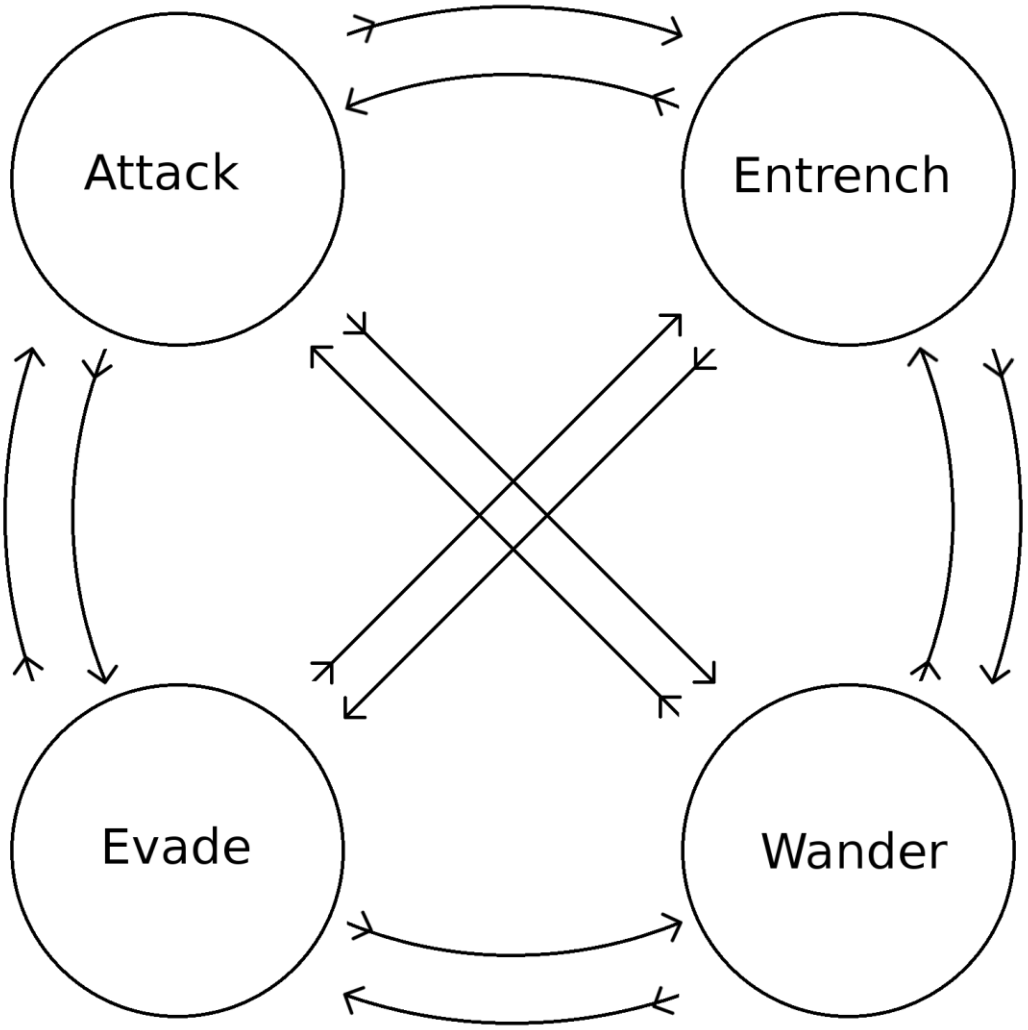

Figure 4.2.4.1

Each of the arrows denotes a flow of control from one state to another if certain conditions are met. These conditions are covered is the descriptions that follow.

The first state that will be described is the “Attack” state. In this state every decision was weighed toward pursuing the human opponents directly and attacking them. The switch-in condition for this state, regardless of the previous state, was a numbers advantage. If an enemy player was eliminated, causing there to be more quadrotor combatants remaining than human combatants, the system would switch out of the previous state and begin attacking head-on. The state machine would tell the tactical planning function to weigh pursuing enemies more heavily when evaluating potentially viable target locations.

The “Entrench” state performed as may be expected given the name. This state prioritized creating a strong presence in a defined region. This state was the one the system started in. The switch-in condition for this state was when the team numbers even out. When in this state the tactical planning function was instructed to prioritize maintaining control points that are clustered relatively close together and near cover zones.

The “Wander” state allowed for the A.Q.U.A.L.U.N.G. system to make choices based on random exploration. This state was only switched into if there had been no sign of the human opponents for a significant period of time. The system would then wander around the environment searching for opponents in a random manner. The benefit of this state was that it allowed the system to react to opponents more focused on hiding than hunting by looking in areas that would not be searched otherwise, as they had no tactical merit. When in the “Wander” state, the tactical planning function was instructed to randomly select regions that little was known about at the time, ignoring the tactical aspects such as maintaining cover of control points. Having a stochastic element as opposed to an optimized seeking function meant that human opponents could not predict when the quadrotors would explore next, and thus would have a more difficult time avoiding detection.

The final state was the “Evade” state. This state was switched into when the humans outnumbered the quadrotors. Under these conditions chasing after the enemy was foolish, and maintaining control was less desirable, as the advantage of numbers was often more influential to the outcome of a battle than the advantage of position. When in this state, the tactical planning function was told to prioritize constant movement from one cover zone to another, forcing the enemy to expose themselves while chasing.

During each state the tactical planning system was constantly evaluating the environment and providing a list of suggested target locations to the state machine based on the prioritization of environmental features. Each set of target locations would be weighted based on the features of the set, such as maximized

control over critical control points, or proximity to cover zones, and so forth. The state machine would then make a determination of which set to use. Once a set of target locations was chosen, these would be provided to the A* path finding algorithm. The A* algorithm would return a sequence of locations, representing the vertices of the path discovered. These points were then communicated to each associated quadrotor. The quadrotor was also constantly broadcasting its current position back to the AI subsystem, so that progress could be tracked. Additionally the state machine was always listening for a collision warning interruption. If this interruption occured the state machine would tell the quadrotor in question to immediately stop moving. The state machine would then provide information on the nature of the interruption to the A* algorithm, so a new path could be determined. The final input the state machine must have responded to was data from the computer the computer vision subsystem. If an enemy was spotted this information was not only given to the probability field algorithm, but also directly to the state machine. When an opponent was spotted the quadrotor was told to pause its current path. The quadrotor was then given the position of the enemy, and told to change its orientation to face the enemy directly. The quadrotor was instructed to fire once it was aiming at the opponent's center of mass. After the quadrotor had fired, it continued along its path. A diagram of the interactions of each of these functions, algorithms, and objects in the AI subsystem is given in Figure 4.2.4.2.

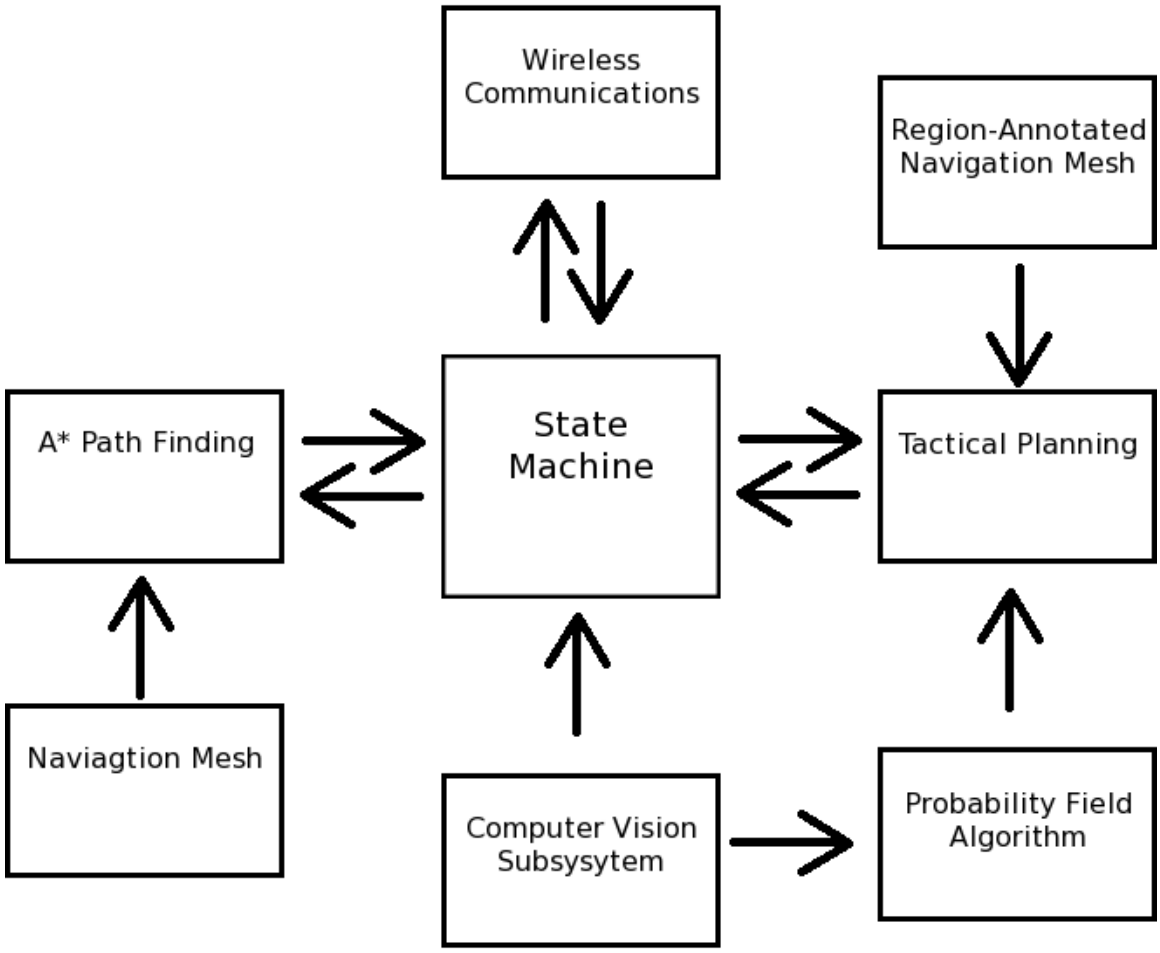

Figure 4.2.4.2: Organizational Layout and Data Flow of AI Subsystem

Each box represents a data structure, function, algorithm, or object with the A.Q.U.A.L.U.N.G. project. The arrows represent the flow of data and control from one box to another.

4.2.5 Hardware for Centralized Control

The hardware for the AI Control System needed to be able to handle processing large amounts of data rapidly and in parallel. As such the following components were purchased to ensure that the system would be able to work as desired under any load.

For the A.Q.U.A.L.U.N.G. project the Intel core i7-3770K Ivy Bridge processor was used. This is a quad-core hyper-threaded processor that operates at 3.5GHz, with a 3.9GHz Turbo. The i7-3770K also has its clock multiplier unlocked, which allowed for overclocking, which was desired as insurance should the default speed have been insufficient. Given the high processing demand on the system, it was therefore intuitive that the primary choice for an Intel processor as opposed to an AMD one is the fact that Intel has a niche in higher-performance CPUs, as well as the fact that it is often more straightforward to overclock an Intel CPU. One of the main advantages of this particular processor was the fact that it is hyper-threaded, allowing for 8 threads to run in parallel, a feat that the Intel's lower tier processors cannot accomplish. This capacity for a higher number of active threads allowed for 4 threads to be dedicated to video processing, one per stream. Then, one thread was be dedicated to communication with each quadrotor unit. One thread was reserved for path finding. One thread was set aside for high-level decision making. The final thread handled all other miscellaneous tasks.

The next component to be discussed is the motherboard. Obviously, the motherboard must support the processor, so it must have had an LGA1155 socket. In order to facilitate the possibility for overclocking either a Z77 or Z75 north bridge was also required of the motherboard. A required aspect that is found on almost all available motherboards was having at least one SATA 6Gb/s port. The final requirement was that the motherboard has at least one PCI Express 3.0 x16 port. All other necessities such as display ports and Gb/s LAN adapters are standard to all motherboards. Meeting these requirements at an affordable cost was the ASUS P8 Z77-V LK Motherboard.

Another component was the Random Access Memory, RAM. In order to allow for some overhead in the video processing aspect, as well as for a complete virtual representation of the environment, 16GB was used. Additionally, RAM timings of 9-9-9-27 or better was required. To accommodate these specifications the G.SKILL F3-1600C9D-16GSR was used.

Continuing the discourse on components for the A.Q.U.A.L.U.N.G. project's central control computer is the topic of storage. Due to the fact that any bottle-necking in the flow of information could have had adverse effect on the performance of the A.Q.U.A.L.U.N.G. project, the faster the hard drive, the better it was. As such a Solid State Drive was required. The hard drive was using the aforementioned SATA 6Gb/s

port on the motherboard. While generally the idea that “bigger is better” is employed in the selection of hard drives for general use, in this case, after a certain point, size was unnecessary. The hard drive selected was the G.SKILL Pheonix III FM-25S3-120GBP3.

One component that was included in the PC for the project was not needed to be purchased, as a group member already had a spare. This component was the graphics card. The graphics card worked with the CPU for the video processing tasks involved in the project. The specific card that was be used is an NVIDIA GeForce 8900 GTX.

Now to get every component the electric power it needed to run, bringing the discussion to the power supply. When selecting a power supply it was important not to get the bare minimum required by the sum of each component. The reason for this is that occasionally each component may require additional power during a performance spike, and over time the capabilities of the power supply may diminish slightly. As a result, having some overhead is always desired to ensure system reliability and durability. To meet this goal a 750W or better power supply was required. In addition to the power rating, many power supplies come with an efficiency certification. This certification, called the 80 PLUS certification, ensures that the power supply performs at or above a specified efficiency, based on the specific certification, under various power loads. Increased efficiency means less power is consumed, which results in a lower cost to run the system as well as being environmentally conscious. Increased efficiency also results in less heat produced, which improves system reliability and prolongs the life of system components. The A.Q.U.A.L.U.N.G. project insisted on using a power supply with an 80 PLUS Bronze certification or better. The power supply used for the A.Q.U.A.L.U.N.G. project was the Thermaltake SP-850M.

The final necessary component for the A.Q.U.A.L.U.N.G. project was a case to house the other components. The most obvious requirement for the case was that it was large enough to fit all of the other components inside, which was easy to verify by the dimensions of the components. The most likely component to cause size conflicts is the graphics card, as graphics cards do not have a standardized width. Thus it may protrude too far from the motherboard if the case is too narrow. The only other requirement for the case was that it supports good airflow in the system. This meant having at least two 120mm fans, one in the front of the case and the other in the rear, and additional vents for increased air flow. Finding a sale for a case that goes above and beyond these requirements for no cost over the bare minimum, the Rosewill THOR V2 was used.

4.3 Power Subsystem

4.3.1 Summary

For what this project is trying to accomplish it could be seen from the beginning of the project that a battery pack would be the only option for powering the quad rotors as they needed to be able to fly and move around in a large game field. When researching battery types there were a lot of different things to consider including: weight of the battery, battery life, cost of the battery and each battery types' pros/cons. Also the energy density of the battery needs to be looked at as it is a measure of how much energy a battery stores for a given volume.

The term cell and battery have become almost interchangeable but a battery is the device that produces electrical energy because of an electrochemical reaction. A cell is the basic electrochemical element. A battery is made up of two or more cells in series to form a unit, but most people use the two terms synonymously. Batteries come in different types: wet cell, dry cell, molten salt and reserve.

Wet cells have a liquid electrolyte and are sometimes called flooded cells because of the
liquid present. This liquid covers all the internal parts of the cell. Wet cells were the first kinds of batteries and can be either primary or secondary cells. Originally these wet cells were built as open-topped glass jars until dry cells and gel cells came into play. Wet cells are now often used in laboratory work, like in beakers to demonstrate how electrochemical cells work. Though they are effective batteries they are not as practical as the more well known dry cells and gel cells. Some examples of wet cell batteries are the Bunsen cell, chromic acid cell, Clark cell, Daniell cell, Grove cell, Leclanche cell, Weston cell and some automotive batteries. As these wet cells are mostly used for stationary purposes, any variation of wet cell battery was deemed unusable for the goals of this project and was not considered when choosing a battery type.

In the case of the dry cell battery, the electrolyte is no longer a liquid but a paste. There is only enough moisture in the cell to allow current to flow. Dry cells can work in any orientation because there is no fear of spilling liquid as in the case of the wet cell. In dry cells, there is a positive and negative pole and in between are different pastes of electrochemicals that are the reason for the chemical reaction which causes the electric charge to flow.

Molten salt batteries can also be primary or secondary batteries. They have as an electrolyte a molten salt, and have a high energy density and power density. Molten salt batteries operate at very high temperatures however and must be well insulated. Molten salt batteries are most commonly found in power plants because they

operate between 470^{o} and 670^{o} Fahrenheit, as this project will reach temperatures nowhere near that range, molten salt batteries were not considered for this project.

Reserve batteries are stored unassembled and are activated when the internal parts are put together, like by adding the electrolyte. They can be stored unactivated for long periods of time. They are primarily short life batteries as in seconds or minutes though they can have a long storage life, as in years. An example of reserve batteries are those found in oceanographic tools that become activated once they are placed in the water. These types of batteries were also deemed unsuitable for this project and were not considered when researching different power sources.

Primary batteries are known as single-use batteries because they are made to be used to their full extent and then discarded. They are not meant to be recharged because the chemical reaction that occurs within the cell is non-reversible. Listed below are different types of primary batteries that were researched when looking for a viable power source.
Alkaline batteries have a long shelf life and high energy density, and are the most common types of batteries. They come in different sizes for example AA, AAA or 9 V, and can be found in almost every household. The amount of current an alkaline battery can deliver is almost proportional to its physical size, so larger alkaline batteries, like C and D cells, can deliver more than smaller ones like AA or AAA. Alkaline batteries generally don't provide more than 1.5 V and don't deliver much current, so for an application that requires more, one would have to use several D-cells to increase the load. Increases the number of batteries will increase the amount of power available but it also increases the weight of the battery pack necessary and for a project that needs to be light and agile alkaline batteries didn't fit the bill.

Figure: Alkaline battery types

Aluminum batteries are also known as aluminum-air batteries and have the highest energy density of all batteries, which would make them very effective. They are not widely used however because of their cost and the lengthy process of the byproduct removal so their use is restricted for mainly military purposes. As this is not a military project, it would have made it very difficult to get aluminum batteries for this project and as such were crossed off the list as well.

Lithium batteries are dry cells that have a high charge density and use a lithium metal or lithium compound for its anode. Because of their long life lithium batteries can be found in think like artificial pacemakers, other implantable medical devices, watches and cameras. They are costly however so they aren't used in less critical applications because the battery may outlive the device it is in. Lithium batteries can

hold up to 3 times the capacity of alkaline packs and are lighter in weight but are also more expensive. They could possibly be an option for the project.

Figure: Lithium batteries, also known as button cell batteries

Mercury batteries are also known as mercuric oxide batteries, or mercurcy cells. They are a type of button cell battery that were used in watches, hearing aids, and calculators. Later research found that these mercury batteries were not exactly hazard free. In 1991 a European commission directive prohibited certain types of batteries that had more than 25 mg of mercury. In 1996 the United States Congress passed an act that banned the sale of any battery that had mercury in it. Since mercury batteries are illegal in the United States that made them not an option for powering the quad rotors.

Nickel oxyhydroxide batteries, are known as NiOx batteries for short, and work well in things that are high-drain applications. They can give up to twice the life of an alkaline battery in a high-drain application, like a digital camera, but in a low drain use they provide about the same life as an alkaline battery. They give off a higher voltage, so it should be noted that that can cause some problems in products that do not have a voltage regulator. Panasonic was able to power a car with 192 AA NiOx batteries, which is in itself an amazing feat but for this project needing to have more than just a few of these batteries is out of the question as it would weigh down the quadrotor and there isn't a lot of space to attach more than a handful.

Silicon-air batteries are made from oxygen and silicon, and are environmentally friendly. The technology for these batteries was developed by scientists at the Technion-Israel Institute of Technology in 2009. Silicon-air batteries are capable of supplying non-stop power for thousands of hours, have an unlimited shelf life, and are lightweight. These batteries have a high tolerance for a wide range of weather conditions and would provide a savings in cost and weight because of their physical make-up. Unfortunately silicon-air batteries have not hit the production lines yet as they are still in the process of being researched. Though these silicon-air batteries sounded amazing they couldn't be used for this project because the public doesn't have access to them.

Silver-oxide batteries are available in different sizes, either small like button cells or larger custom designs. Their cost is directly proportional to their size because they are made of silver. Larger cells are used for military applications like in Mark 7 torpedoes and Alfa class submarines. They can provide up to 40% more run time than lithium-ion batteries which is a plus, but they become hazardous when they leak, about 5 years after the first use. For the goals of this project, the group would need something larger than a button cell and time constraints didn't allow for getting

a larger custom design when there where other things that needed to be accomplished.

Figure: Silver oxide batteries

Zinc air batteries work when the oxygen in the air oxidizes with the zinc in the battery. Zinc-air batteries have high energy densities and come in different sizes, small enough to power hearing aids, or large enough to power film cameras, and even larger for electric vehicle propulsion. They can be used to replaced the banned mercury batteries and it is envisioned that in the future as a battery for electric vehicles. Secondary zinc air batteries are hard to come by because the chemical reaction necessary to get them to work has to be closely controlled.

Zinc-carbon batteries are actually composed of a battery that is in zinc can so these batteries can be a container and also a negative terminal. Zinc-carbon batteries are the least expensive primary batteries and generally the types of batteries that are found when a product comes with “batteries included.” Zinc-carbon batteries were the first dry batteries to become commercial, and can be used in low drain or intermittent devices. The problem with these batteries is that they have a low energy density, meaning that the voltage isn’t constant during use and therefore wouldn’t be advisable to use for this type of undertaking.
Zinc-chloride batteries are better version of the zinc-carbon batteries because it uses purer chemicals. These batteries have a longer life than their predecessors and steadier voltage potential as well. They are marked heavy-duty to distinguish themselves from the regular zinc-carbon but they still don’t last as long as alkaline cells. As alkaline cells weren’t going to do the trick for this project, zinc-chloride cells wouldn’t work either

As the quad rotors are something that are meant for continuous use for multiple uses it was apparent that other types of batteries should be considered, since primary batteries are meant to be discarded after they have been used to capacity. Secondary batteries are also known as rechargeable batteries, or accumulators, or storage batteries. The chemical reactions that occur within these types of batteries are able to be reversed and allow these battery types to be used again. Found below are some of the secondary batteries that were researched during the course of this project.

Lithium- air battery, also known as Li-air, have an extremely high energy density because they use oxygen in the air to oxidize instead of another chemical necessary in the battery. The only problem with this battery type is, like the silicon-air battery the research hasn't been completed and so neither has the manufacture of the batteries themselves.
Lithium-ion batteries use different cathodes and electrolytes depending on what type of Li-ion battery one is looking at. Advantages of using these types of batteries include, their weight (generally lightweight), their size, and relatively high energy density. They can operate in a wide range of temperatures and are more effective than nickel-based batteries. Lithium is a little more pricey than other materials used to make rechargeable batteries however, and can make Li-ion batteries a little more costly. These could have been an option for this application.

Figure: An example of a lithium ion battery

Lithium iron phosphate batteries are also known as LFP batteries and offer a longer life cycle than other lithium-ion batteries. They experience a slower rate of capacity loss and use safer cathode material than some other lithium ion batteries. LFP batteries could also have been a good power source.

Figure: An example of a lithium iron phosphate battery

Lithium-sulfur batteries, also known as Li-S batteries have a very high energy density and are relatively lightweight products. Because sulfur is pretty low cost and because they have a higher energy density it is said that lithium-sulfur batteries will supersede lithium-ion cells. However because of their high potential it is advisable to use a microcontroller, a voltage regulator and other safety precautions to control the battery's operation and to prevent quick discharge. Lithium-sulfur batteries while they

seemed like a viable option required a lot to monitor and the project didn't need another thing to focus when other options were available.

Lithium-titanate batteries, known to some as LTO batteries, are good for certain applications that require high rate capability and long life. They can be charged a lot faster than other rechargeable batteries, in less than 10 minutes, but that is because of their low voltage. Their low voltage is an indication of their low energy density however and was deemed to be best suited for a different application then the one proposed by this project.

The nanowire battery is a type of lithium-ion battery that was invented in 2007 at Stanford University. It is created by using a stainless steel anode that is covered in silicon nanowires, and because of silicon's elemental properties it can store more than lithium. This allows nanowire batteries to have a greater energy density, and their large surface area means they are also capable of fast charging and discharging. Because these batteries have not been around for very long testing is still being done on life cycle testing, they were expected to be commercialized by 2012. Prototypes for certain applications like for cell phone batteries, have been devised but as this project doesn't fall under that category so other batteries had to be looked at.

Figure: An example of a silicon nanowire battery

Nickel cadmium batteries, also known as NiCd or NiCad batteries, are made in a variety of sizes. They work pretty well in all temperatures and have a good capacity and cycle life but have higher self-discharge rates than other batteries. They have a low internal resistance and so can supply high surge current which makes them good for things like cordless power tools and remote-controlled electric model vehicles. These batteries were considered a good option for powering the quad rotors.

The nickel-iron battery, also known the NiFe battery, is very resistant to abuse in that it can be overcharged, overdischarged, short-circuited and still have a relatively good life capacity. It does however have a low specific energy, does retain charge

particularly well and is expensive to make. Because of these disadvantages of the nickel-iron battery, other battery types have replaced it for most applications and so it was not considered for this project.

Nickel lithium batteries, also known as Ni-Li batteries, are a prototype battery using a lithium anode and nickel hydroxide cathode. Generally, these two different materials can't be used together because there isn't a known electrolyte that is compatible with both metals. Instead two electrolytes will be used and separated so each touches its respective metal. Ni-Li batteries are proposed to hold more than three times as much energy as lithium-ion batteries and to be safer as well but this battery technology is still being researched. Since this battery is not commercialized yet, it was obvious we could not use it for the project.

Nickel-metal hydride batteries are similar to nickel-cadmium batteries, but it can hold two or three times the capacity of one. Its energy density is similar to that of a lithium-ion battery. NiMH, nickel-metal hydride batteries for short, have a poor charge retention though. They also have a slight memory effect, which means that after a while the battery will hold less charge. For a project that requires a lot of recharging, it was thought to be a better idea to pick a battery that didn't suffer from charging issues.

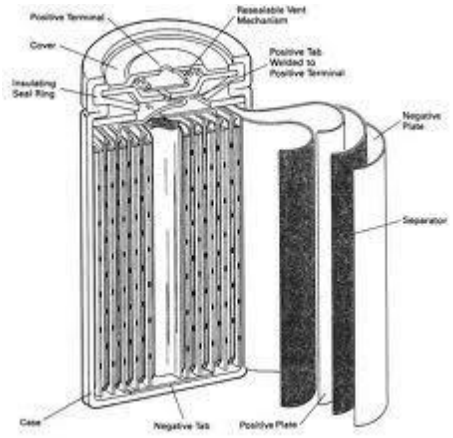

Figure: Cutout of a nickel-metal hydride battery

Nickel-zinc batteries come in the same sizes that more commonly known alkaline batteries come in, but these are rechargeable. They perform similar to nickel-cadmium or nickel-metal hydride batteries but with a higher nominal voltage. It is predicted that they may replace lead-acid batteries because of their better energy-to-mass ratio but after about 30 charges they seem to self-discharge quicker. If this project were not designed for longevity maybe nickel-zinc batteries could have been a choice.

Figure: Example of nickel zinc battery

Polysulfide bromide batteries are actually a type of fuel cell, and fuel cells are not viable for this type of application because fuel cells require a constant input of fuel and oxygen to produce the chemical reaction that makes the electricity.

Potassium-ion batteries were first invented in 2004 as an intended replacement for lithium-ion batteries. Unfortunately only a prototype has been built and as such this battery type could not be considered for this project.

Sodium-ion batteries are predicted to be a cheaper alternative to lithium-ion batteries. Sodium-ions are more abundant than lithium ones and can store more energy as well. Unfortunately for the time frame of this project sodium-ion batteries will not be sufficient as they have not been commercialized yet because research is still being conducted.

Vanadium redox batteries are good for large energy needs devices, but they don't work well in a wide range of temperatures and they are costly to make. They are also larger than what this project would require, and therefore were not considered for powering the quad rotor.

Zinc bromine batteries are environmentally friendly, are relatively cheap to manufacture and have a high energy efficiency. They are also one-sixth the weight of lead acid batteries and have minimal battery degradation and performance loss. The problem with zinc bromine batteries in terms of this project, is that zinc bromine batteries are used mainly for industrial purposes, and can be found in things like golf carts, lawn mowers and wheelchairs. All of these are ground vehicles and can support a larger power source and this project is focusing on an aerial vehicle, which requires something a bit smaller.

Lithium-ion polymer batteries, Li-Po batteries, are a type of lithium-ion battery that are lightweight, are more resistant than other batteries to overcharging and has a recharge life of 300-40 cycles. Li-Po batteries are very commonly used among hobbyists that work with hobby aircrafts, and come in sizes that would be perfect for what this project is trying to accomplish. They come in varying voltages and varying mAh and a perfect fit can be chosen for the user's needs. It was deemed that this should be the power source for the project. Because the frame of the quad rotor could support it, and because the group bought two Li-Po batteries they decided that instead of interchanging the batteries when one needed to be recharged, they would just use both batteries to charge the quad rotor at the same time. Though it was possible to have all the elements on the quad rotor powered by the Li-Po batteries, for ease and convenience the camera that was mounted on the quad rotor was powered separately by a single 9V battery.

Figure: Li-Po batteries as used for this project

4.3.2 Power distribution to system

It is one thing to have power to the system but next comes the issue of distributing it to the elements in the system so all necessary components have power. The diagram below is a high level description of the different aspects of the project that need power.

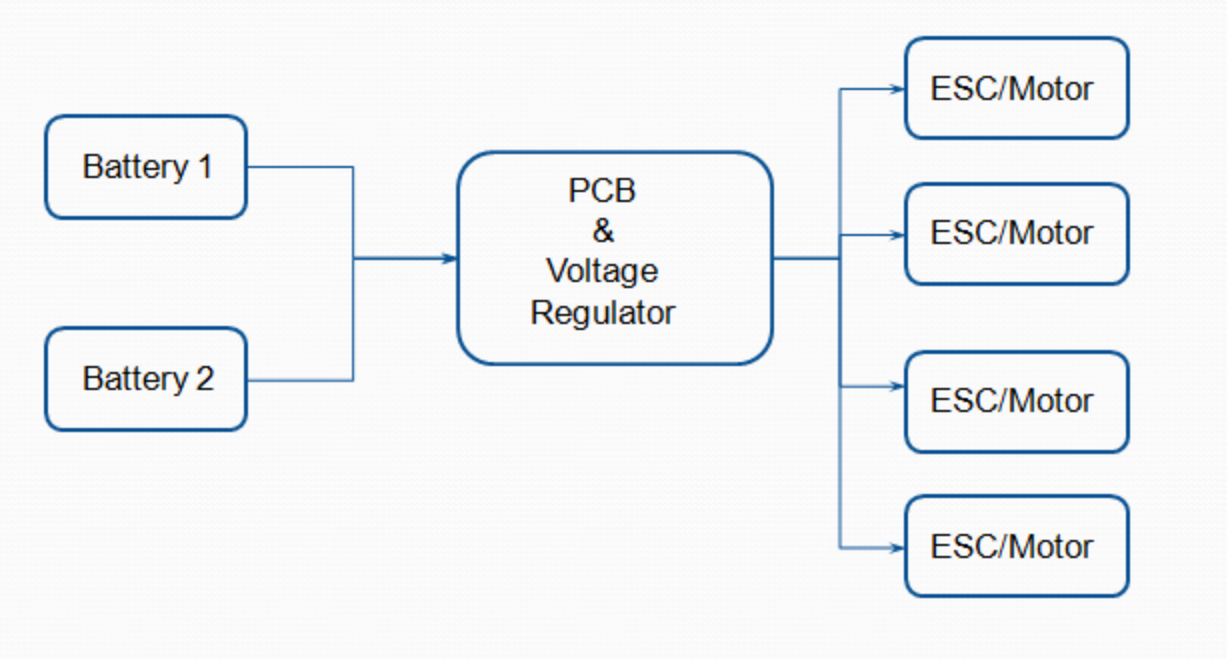

Figure: Block diagram of how the power will be distributed

To get power to all the elements in the system, the group deiced to make a printed circuit board (PCB) that would serve as a power distribution center for the whole system. The two connectors on the side of the PCB would connect to the two Li-Po batteries and bring power to the system.

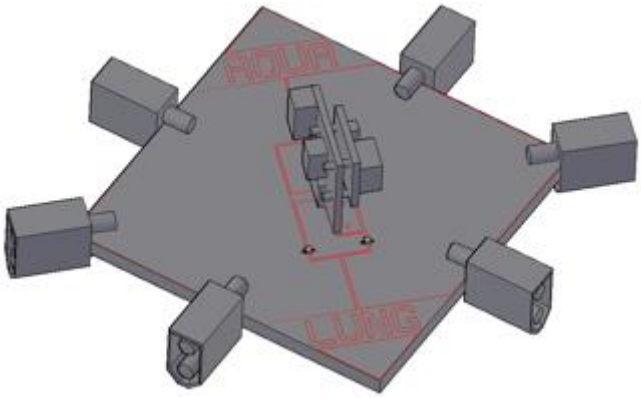

Figure: AutoCAD image of the PCB the group designed

In the middle of the PCB, the group mounted a 5V voltage regulator that came with the microprocessor. Two small diodes can be found near the voltage regulator that ensured that the batteries did not feed each other but were feeding the necessary elements. The four other “arms” that are connected to the PCB branch off to the four separate electronic speed controllers and motors. The voltage regulator connects

directly to the microprocessor and receives its power in this fashion. The group name A.Q.U.A.L.U.N.G. was etched on the board for more aesthetic reasons.

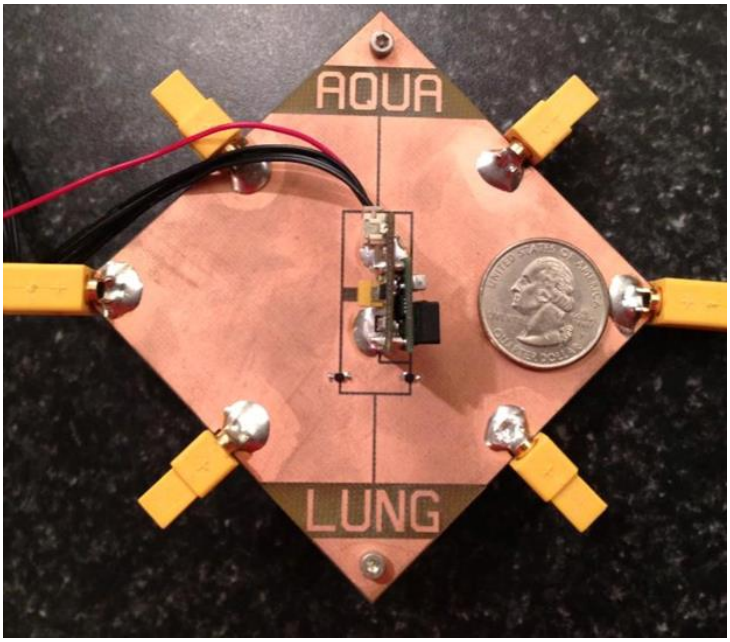

Figure: Actual PCB as used for this project

4.4 Flight Control Subsystem

4.4.1 Summary

The flight control sub system on the A.Q.U.A.L.U.N.G. was the main component of stabilization for the air craft, as well as, a partial component of navigation. For the stabilization, an Inertial Measurement Unit (IMU) was used. IMUs are electronic devices used on moving vehicles that measure its velocity and orientation, as well as gravitational forces. Two components are quintessential on an IMU, an accelerometer and a gyroscope, a third component is sometimes added for farther accuracy when tracking position, this component is a magnetometer. To give the A.Q.U.A.L.U.N.G. the biggest advantage when stabilizing and trying to track and navigate its position, all three devices were used.

At the heart of the flight control system, doing all the calculations was the microcontroller. Within the microcontroller there had to be enough memory and computing power to handle the three sensors that will be on the IMU. This is a constraint that is at the base of the design for the flight control subsystem, as it was most important to flying a stable aircraft.

Other important factors for choosing which microcontroller and IMU to use are operating voltage, input voltage, DC current I/O, clock speed. The operating voltage is the voltage at which the board's electronics turn on, so if we didn't get enough voltage (the recommended input voltage) the board might not turn on, rendering it useless. The input voltage, as previously mentioned, was important because if not enough voltage is supplied to the electronics the board might not turn on or just run unstable. But if too much voltage was supplied to the board the voltage regulators might overheat and damage the board, or a number of other heat related problems could occur. The digital pins used for input/output have max current specifications that needed to be taken into consideration. Clock speed affects how many instructions will be executed in a set amount of time, and also affects how much power will be consumed from the batteries. A faster clock speed draws more power and of course a slower clock speed draws less power.

The physical dimensions of the microcontroller matter as well since all the guts of the machine needed to fit in a certain amount of finite space. So the area becomes important for building our frame, and when the IMU was placed onto the microcontroller, the height of the two become important as some type of protective shield would most likely incase the electronics for protection. The weight of the flight control system also played a role in the overall design of the aircraft. It may not weight much by itself but with everything else on the craft, the weight could total up, with this added up weight in mind picking the motors and propellers then become the topic, since the craft had to be able to lift itself off the ground and fly without consuming too much power too quickly.

Last, but certainly not least, was the cost of the fight control system. With all the money in the world, all the factors could be optimized and a perfect microcontroller could have been purchased. But on a budget, picking and choosing become essential.

4.4.2 IMU Stabilization

Before any specs can be looked at and examined a rudimentary explanation for how to stabilize the quadrotors will be looked at from the point of view of each sensor on the Inertial Measurement System.

The first sensor on the IMU was the accelerometer, for an aircraft such as the quadrotors that will be implemented for this project a 3-axis accelerometer was used. The accelerometer delivers information from itself to the microcontroller in one of two ways, depending on the brand bought, digitally or analog. If the accelerometer outputs from itself with an analog voltage, this voltage will be within a preset range, knowing this range one can easily convert to a digital value using an Analog to Digital Converter. Depending on what ADC is use the range will be different, for example there are ADCs that are 8-bits, 10-bits, 12-bits or whatever. If it is 8 bits, the range will be from 0 – 255, 10-bits 0 – 1023, similarly 12-bits is from 0 – 4095 and so on following the 2^ (bits – 1) formula. The ADC will then output a number within the range for each axis, for example (ADCx, ADCy and ADCz being the outputted values for each axis from a 10 - bit Analog to Digital Converter):
ADCx = 855 or in 10-bit binary 11 0101 0111
ADCy = 944 or in 10-bit binary 11 1011 0000
ADCz = 311 or in 10-bit binary 01 0011 0111
In the end a g value (g = 9.8 m/s/s) will be obtained for each axis, but some more calculations will need to be done first before this value is reached.
Now that a decimal value has been given from the ADC, it needs to be turned into a voltage. Each ADC should have a reference voltage in its data sheet that corresponds to the number of bits. As an example let's say the reference voltage is 1.8 volts then:

Volts = ADC * Refv / 2^(bits – 1)

So, Voltsx = ADCx * Refv / 2^(bits – 1) → 855 * 1.8 v / 1023 = 1.504 V
Voltsy = ADCy * Refv / 2^(bits – 1) → 944 * 1.8 v / 1023 = 1.661 V
Voltsz = ADCz * Refv / 2^(bits – 1) → 311 * 1.8 v / 1023 = 0.547 V

After the calculations a voltage level is reached, now it is essential to calculate how far off the zero-g voltage level the accelerometer is. The zero-g voltage level is the voltage level you expect as an output if the accelerometer were under 0gs of acceleration. This value is specified in the spec sheet that comes with an analog output accelerometer. For this example the specified zero-g voltage will be 1.50 v. So in order to figure out how far off from zero-g the accelerometer is, the difference between the two levels needs to be taken:

Deltavolts = Volts – VoltsZerog

So, Deltavoltsx = Voltsx – VoltsZerog → 1.504 V – 1.500 V = 0.004 V
Deltavoltsy = Voltsy – VoltsZerog → 1.661 V – 1.500 V = 0.161 V
Deltavoltsz = Voltsz – VoltsZerog → 0.547 V – 1.500 V = - 0.953 V

Now Deltavolts is the accelerometer reading in volts and the next step is to turn this voltage into a value in g. Again to do this a value from the accelerometer spec sheet is needed. The sensitivity is the relationship between the changes in acceleration to change in output signal. [6] For an analog accelerometer this value is expressed in mV/g, so all one would need to do to change the Deltavolts value to a g value is to divide by the sensitivity. For a simple example, the sensitivity is 400.0 mV/g or .4000 V/g. The equation for this conversion would look like this (with R being the vector direction of acceleration):

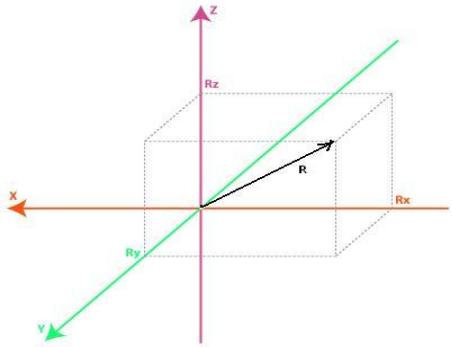

Figure 4.4.1.1.

R = Deltavolts/Sensitivity

So, Rx = Deltavoltsx/Sensitivity → 0.004 V/.4000 V/g = 0.0100g

Ry = Deltavoltsy/Sensitivity → 0.161 V/.4000 V/g = 0.4025g

Rz = Deltavoltsz/Sensitivity → -0.953 V/.4000 V/g = -2.3825g

An inertial force vector has now been established as R, now would be a good time to talk about what exactly the R vector value means. As the name suggests the accelerometer measures acceleration, that's great if the quadrotor is moving, "accelerating", because it makes it possible to tell which direction the quadrotor is moving because the R vector will be pointing in that direction and how fast by calculating the integral of acceleration. However what if the quadrotor isn't moving? If the quadrotor isn't moving there is still a force acting on the accelerometer, that force is gravity. So if the quadrotor is not being subjected to any other forces other than gravity, the R vector becomes our gravitational vector.

To level out the quadrotor, aka stabilize it; it will be necessary to calculate the angle between the R vector and the x, y and z axis of the ground plane. Zeroing out the angle between the z axis would mean that the R vector of the quadrotor and z axis of the ground plane are in line with one another. Similarly, having the R vector of the quadrotor 90 degrees from the x and y axis would stabilize the

aircraft, as it would not be able to roll or pitch. Roll being the rotation around the x axis, and pitch being the rotation around the y axis. [4.4.2.1]

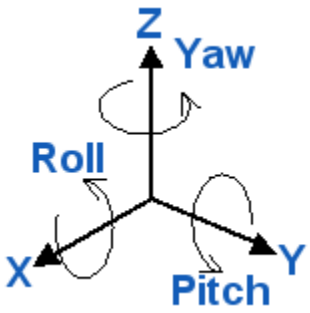

Figure 4.4.2.2

So what about the Yaw, or rotation about the z axis? This motion is stabilized by the motors, on a traditional helicopter the tail propeller cancels out the yaw motion, but on a quadrotor, rotors 1 and 3 rotate in one direction while rotors 2 and 4 rotate in the opposite direction, thus cancelling out each other's torques achieving yaw angle stabilization.

To calculate the angles between the R vector and the ground plane x, y and z axis some rather simple trigonometry is needed.

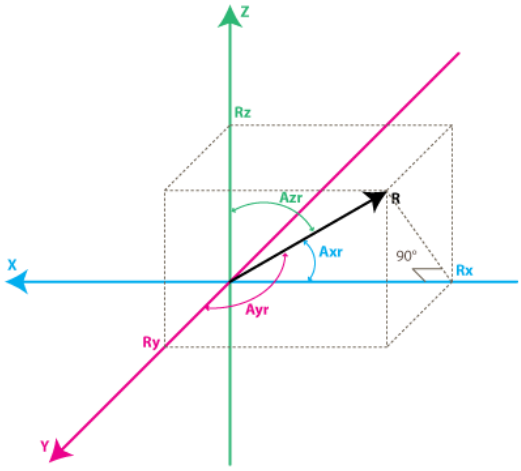

Figure 4.4.2.3

The angles are defined as Axr, Ayr and Azr. With Axr being the angle between the vector R and the x axis, Ayr being the angle between the y axis and R vector and similarly Azr being the angle between the R vector and z axis:
(R being the square root of (Rx^2 + Ry^2 + Rz^2)

So, cos(Axr) = Rx/R
cos(Ayr) = Ry/R

cos(Azr) = Rz/R

To get the values Axr, Ayr and Azr the inverse of cosine needs to be taken,

So, Axr = arccos(Rx/R)
Ayr = arccos(Ry/R)
Azr = arccos(Rz/R)

Now, an acceptable form for the inclination of the quadrotors has been reached. From here it will be possible to read the data from these angles and properly stabilize the aircraft in theory. But in real world applications the accelerometer is not always 100 percent accurate so if you would like a more accurate reading for stabilization of the quadrotors adding a gyroscope to the mix would benefit the quadrotors tremendously. [4.4.2.1]

A gyroscope measures the rotation around the axes as a rate of change, for this project a three axis gyroscope will be used.

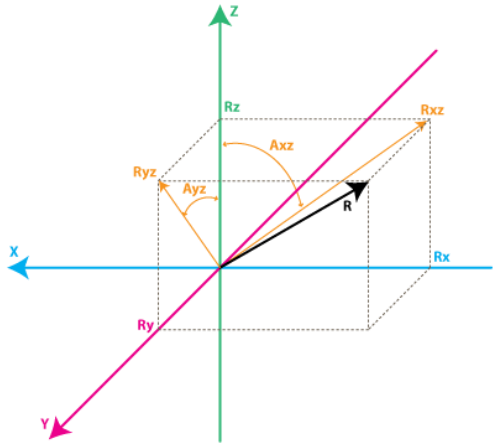

Figure 4.4.2.4

For a three axis gyroscope we will define the vector R in terms of Rxy, Rxz, and Ryz. With, Rxy being the portion of R on the xy axis
Rxz being the portion of R on the xz axis
Ryz being the protion of R on the yz axis
Now that the projection of the inertial force vector if defined, the angles between them and the axes can be defined as well. These angles will be Axz, Ayz and Axy.
Axz is the angle between the Rxz vector and the z axis
Ayz is the angle between the Ryz vector and the z axis
Axy will always be 90 degrees, since the xy axis is orthogonal to the z axis

In other, more familiar, words these angles can be expressed as:

Axz = angle of rotation around the y axis or pitch

Ayz = angle of rotation around the x axis or roll

Axy = angle of rotation around the z axis or yaw

Instead of walking through step by step of how to get from the R vector to a value in degrees per second, the variables and constants will be explained and then a complete formula will be given, as well as an example. A gyroscope will output a value from its Analog to Digital Converter for each axis of rotation:

ADCxy being the output value from the gyroscope of the rotation around the z axis

ADCxz being the output value from the gyroscope of the rotation around the y axis

ADCyz being the output value from the gyroscope of the rotation around the x axis

Refv will be the refrence voltage from the ADC, and as with the accelerometer a 10 bit ADC will be used in this example.

VoltsZeroRate is the voltage that is output from the gyroscope when there is no rotation at all. The Sensitivity of the gyroscope tells you how many volts the output from the gyroscope will increase when the rotation of the gyroscope is increased by one degree per second. And of course the result that is needed which will be in degrees per second is the rate of change in angle: RateAxy, RateAxz and RateAyz

So, RateAxy = (ADCxy*Refv/1023 – VoltsZeroRate)/Sensitivity

RateAxz = (ADCxz*Refv/1023 – VoltsZeroRate)/Sensitivity

RateAyz = (ADCyz*Refv/1023 – VoltsZeroRate)/Sensitivity

For an example, Refv = 3.0 V, VoltsZeroRate = 1.5 V, and the Sensitivity = 0.002 V/deg/s. If the ADC outputs are as follows:

ADCxy = 645 or in 10-bit binary 1010000101

ADCxz = 543 or in 10-bit binary 1000011111

ADCyz = 311 or in 10-bit binary 0100110111

Plugging all the values into the formula yields:

RateAxy = (645*3.0V/1023 – 1.5V)/0.002V/deg/s = 195.75 deg/s

RateAxz = (543*3.0V/1023 – 1.5V)/0.002V/deg/s = 46.188 deg/s

RateAyz = (311*3.0V/1023 – 1.5V)/0.002V/deg/s = -293.99 deg/s

Now that a useful accelerometer value and a useful gyroscope value have been reached it is time to put them both together into an algorithm. To avoid confusion later on in the calculations a variable name change will be needed to specify what value is coming from where. Rx, Ry and Rz from the accelerometer will now be called Rxacc, Ryacc and Rzacc. To recall:

Rxacc = (ADCx*Refv/1023 – VoltsZerog)/Sensitivity
Ryacc = (ADCy*Refv/1023 – VoltsZerog)/Sensitivity
Rzacc = (ADCz*Refv/1023 – VoltsZerog)/Sensitivity

To make things simpler as well, Racc will be normalized:

| Racc | = SQRT(Rxacc^2 + Ryacc^2 + Rzacc^2)
Racc (normalized) = <Rxacc/| Racc |, Ryacc/| Racc |, Rzacc/| Racc |>

The output from the algorithm will be a variable call Rest which is the accelerometers corrected values based upon gyroscope data and past estimated data. Of course:

Rest = <Rxest, Ryest, Rzest>

An overview of the algorithm would be helpful now as to avoid some confusion before some calculations start. The first thing is that the accelerometer is going output a value Racc, then the algorithm will take that value and check it against the gyroscope data and past Rest data, it will correct whatever need be corrected from these values and output a new estimated vector Rest. [4.4.2.1]

So starting at time zero, Rest(0) = Racc(0), then the algoritm will do regular measurements at equal time intervals of T seconds, this will yield new measurements Racc(1), Racc(2), Racc(3) and so on, as well as yield new estimates of Rest(1), Rest(2), Rest(3) and so on. With Rest(n) being calculated from Racc(n), Rest(n-1) and Rgyro which will be introduced next.

Rgyro will be a new measured value that uses data from the gyroscope and the previous estimate, it is a vector as well:

Rgyro = < Rxgryo, Rygryo, Rzgryo >

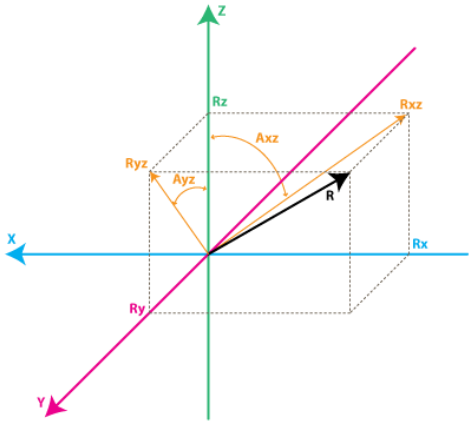

Figure 4.4.2.4

Looking at the triangle formed by the z axis and the Rxz vector, it can be deduced that

Tan(Axz) = Rx/Rz → Axz = atan2(Rx,Rz)

With this and knowing Rxest(n-1) and Rzest(n-1), Axz(n-1) can be found:

Axz(n-1) = atan2(Rxest(n-1), Rzest(n-1))

The gyroscope measures the rate of change of the angle in question so an estimation of the new angle Axz(n) is as follows:

Axz(n) = Axz(n-1) + RateAxz(n)*T

RateAxz is the reading obtained from the gyroscopes ADC for the Axz angle, but to get a more precise formula the average ratoation rate can be used, called RateAxzAvg:

RateAxzAvg = (Rateaxz(n) + RateAxz(n-1))/2

Then replacing RateAxz with RateAxzAvg would yield:

Axz(n) = Axz(n-1) + RateAxzAvg*T

The same process can be done for the Ayz angle which would yield:

Ayz(n) = Ayz(n-1) + RateAyz(n)*T or Ayz(n-1) + RateAyzAvg*T

Now to relate this angle calculated by the past estimation to Rgyro, first establish that the magnitude of Rgyro = 1 since Racc is normalized. Using this, this relationship can be established: Rxgyro = Rxgyro/1 = Rxgyro/SQRT(Rxgyro^2 + Rygyro^2 + Rzgyro^2)

From that base equation, some substitutions can be made where:

sin(Axz) = Rxgyro/SQRT(Rxgyro^2 + Rzgyro^2)

cos(Axz) = Rzgyro/SQRT(Rxgyro^2 + Rzgyro^2)

tan(Ayz) = Rygyro/Rzgyro

And with other algebraic and triganomic substitutions,

Rxgyro = sin(Axz(n))/SQRT(1 + cos(Axz(n))^2 * tan(Ayz(n))^2)

& Rygryo = sin(Ayz(n))/SQRT(1 + cos(Ayz(n))^2 * tan(Axz(n))^2)

For the sake of simplification on the microprocessor further simplification to the formula will be done:

Rxgyro = 1/SQRT(1 + cot(Axz(n)^2 * sec(Ayz(n)^2)

Rygyro = 1/SQRT(1 + cot(Ayz(n)^2 * sec(Axz(n)^2)

Rzgyro = SQRT(1 – Rxgyro^2 – Rygryo^2)

Now the two values Racc and Rgyro have been calculated, Racc from the current readings on the accelerometer, and Rgyro from Rest(n-1) and current readings on the gyroscope. Now comes the final step in the whole process, getting the final Rest(n) value.

Rest(n) is calculated by using both Racc and Rgyro, and the weighted average is taken between them:

Rest(n) = (Racc*w1 + Rgyro*w2)/(w1 + w2)

Dividing by w1 in the numerator and denominator leaves

Rest(n) = (Racc + Rgyro*w2/w1)/(1 + w2/w1)

Where w2/w1 is the deciding factor of how much more the gyroscope should be

trusted compared to the accelerometer. If the gyroscope should not be trusted than w2/w1 goes to zero and all that is taken into account is Racc/1 or Racc. If the gyroscope should be trusted considerably more than the accelerometer than a higher value for w2/w1 will be put into place, this causes the gyroscope readings to dominate. The value w2/w1 should be experimentally calculated. So now:

Rxest(n) = (Rxacc + Rxgyro*w2/w1)/(1 + w2/w1)
Ryest(n) = (Ryacc + Rygyro*w2/w1)/(1 + w2/w1)
Rzest(n) = (Rzacc + Rzgyro*w2/w1)/(1 + w2/w1)

To normalize: R = SQRT(Rxest(n)^2 + Ryest(n)^2 + Rzest(n)^2)

Rxest(n) = Rxest(n)/R
Ryest(n) = Ryest(n)/R
Rzest(n) = Rzest(n)/R

These are the values that will be used for correction of inclination for the quadrotors, along with the values outputted from the magnetometer, a very similar if not the same procedure for converting output applies. [4.4.2.1]

Everything up to this point has made the assumption or been based off the fact that the sensors on the IMU output analog values. For IMU sensors that output digital values most of the procedure is the same depending on the model and make of course. The BMA150 accelerometer from Bosch Sensortec outputs a 10-bit binary number that relates directly to a g value, thus skipping a lot of calculations needed for analog output accelerometers. For example (as listed in the spec sheet):

- 2.000g	10 0000 0000
- 1.996g	10 0000 0001
...	…
-0.004g	11 1111 1111
+0.000g	00 0000 0000
+0.004g	00 0000 0001
…	…
+1.992g	01 1111 1110
+1.996g	01 1111 1111

So this accelerometer will output an R vector, Rx, Ry and Rz automatically, leaving just angle calculations left. No matter what output from an analog or digital accelerometer, gyroscope or magnetometer, it is possible to convert said output into a usable form for the software to correct inclinations in the quadrotors flying pattern and stabilize the aircraft accordingly.

4.4.3 Navigation (Dead-Reckoning)

In order for the AI controller to send out instructions on where the quadrotors would go next, the quadrotor had to relay its position in space back to the central computer. This feat would have been completed with dead reckoning, a method for finding ones position using previous position, and velocity over a time period. Using all the conversion in the stabilization section above, the accelerometer outputs acceleration, so to get position out of it gravity would have to be removed and the integral would need to be taken. Once this was integrated the velocity was now found, one more integral will yield a position. The gyroscope outputs a rate of change, which is velocity, so this output would only need to be integrated once to find position.

Errors in the integrations would have occurred; the trick would be to filter out these errors. Right off the bat accelerometers usually have more noise than gyroscopes, which leads to faster drift. Drift being how far off of the actual position the measurement is reading. Since a double integration would be needed, any residual bias causes errors that grow with the square of time, so it would not take long for dead-reckoning that is uncompensated to become unstable with accelerometers. Gyroscopes are not Saints either, they have problems as well.

The trick to all of this would have been, finding out approximately how long it would have taken for the position that was being relayed back to become unreasonably large and off the actual position. Whether it was a few seconds or a few minutes it would have to be corrected before that limit was reached, or else bad things can happen to the quadrotors, consequences discussed in section 8.4 Risk Assessment.

So what would have been some good ways to correct these errors? It depended on the application the IMU was being used for. If dead-reckoning was being used by ground vehicles than it might have been easier to correct these errors, it also would take the z-axis out of the equation since the vehicle is going to be on the ground. For example a sensor could have been place upon the wheel of the vehicle and every time it turned one whole revolution the vehicle had moved the distance of the circumference of the wheel. All that would be left to do would be figure out direction which could be done with the sensors aboard the IMU. However, there is no way to check the distance traveled, like the previous example, while flying. The A.Q.U.A.L.U.N.G. system did have one advantage that could have been used for checking its position periodically, and that was the

camera. Let's compare this advantage to what humans use in everyday life, sight. If a human knows its environment, let's say a bedroom, they know exactly how many steps it takes to reach the dresser from the bed. For this example its two steps in the x direction, a 90 degree turn, then two steps in the y direction. This task can be accomplished without fail 100 percent of the time with sight, and also very accurately without sight, just like the vehicle mentioned above. But now, if you take a human, blind fold it and put it into a swimming pool, where it will be "weightless" then give it a route to take to get to the stairs, this task becomes extremely hard. Let's say this human now has to move two units in the x-direction make a 90 degree turn and then move two units in the y-direction. The term unit is used here because there is no solid rate or distance of movement to rely on. The term stroke could be used, however not every humans stroke equals out to the same distance, just as not every quadrotors burst of motor equals the same distance. So now not only does this human have to manage to move a set distance in the x direction then turn exactly 90 degrees and move a set distance in the y direction, it has to do this with its eyes closed and a with a possibility of moving upward or downward in the z direction. This is a comparable metaphor for what these quadrotors had to do with dead-reckoning.

What is the simple answer for the human to be able to reach the stairs in the pool? Take the blind fold off and look where you're going! This was one possible solution for the quadrotors in the A.Q.U.A.L.U.N.G. system because it used vision to spot opposing players during the game of laser tag. The camera being used for enemy detection could also have been used for visual "check-ups" on the dead-reckoning position. Through-out the course, there could have been strategically placed icons, or symbols. These symbols would be mapped into the AI controller's memory just as the rest of the course was, and would be indicators of where the quadrotors were in the course. For example, all four quadrotors take off from the starting base, where all the sensors are zeroed out, they fly around while the dead-reckoning algorithm does its calculations on where the quadrotors are, once the camera sees the symbol, the AI controller would recognize it and relate it to a location on the map, then send this position information back to the quadrotor. The quadrotor now would have a true position reference to compare its dead-reckoning position too; it could either do one of two things now. It could "zero" out its readings and take the position of the symbol to be its position in space, "zero" is in quotations because this would not be the true zero it would be just a refreshing of the system to give the dead-reckoning a absolute for sure reference to its position in space at that time. The second option it had was just reference this symbols position in space and use its past data to try and find out exactly where it was in space.

The second option would have probably been the best bet for the goal that was trying to be accomplished. If the first option was used, than dead-reckoning would not even needed in the first place, a bunch of symbols could be everywhere and the vision could just fly the aircrafts around using sight. However using both methods combined would yield the greatest accuracy when determining position. The trick would be to check the position of the dead-reckoning before massive error sets in, because if the dead-reckoning error was way off and the camera sees a symbol, the average of those two things would still be off. Another thing to consider was not using the average of the two positions, Figure 4.4.1 shows that the average of the two positions would be right in the middle of them which would not be wanted.

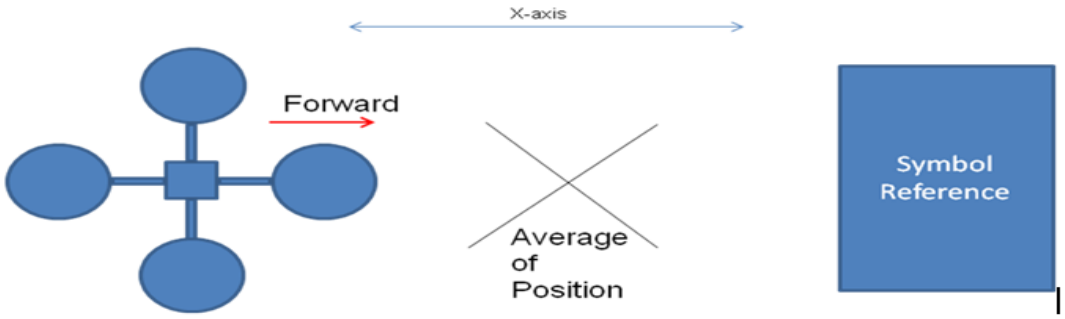

Figure 4.4.3.1

Using this symbol as a reference could have, however, yielded an accurate "zeroing" target. If the quadrotor was facing forward when it saw the symbol, it would then know that it was headed straight toward the symbol and could use the symbols x-axis to check its dead-reckoning x-axis, and correct it as needed. With multiple symbols around the course to add a system of checks and balances to the dead-reckoning of the IMU, a reliable navigation system could have been accomplished. And with an accurate position being reported to the AI, everything could run swimmingly for the AQUALIUNG system.

Stabilization was achieved on the A.Q.U.A.L.U.N.G. system; however a true dead-reckoning was never fully achieved. Drift errors were too significant in the system to overcome. A GPS module was used on A.Q.U.A.L.U.N.G.s quadrotors; however this did not give the project the accuracy it was truly aiming for..

4.5 Computer Vision

4.5.1 Summary

A.Q.U.A.L.U.N.G. is an extremely intensive software prototype. Starting from the actual navigation artificial intelligence all the way to the motion detection and processing algorithms. Not only will A.Q.U.A.L.U.N.G. be used for a fun way to challenge people to laser tags, but it can also be expanded into military and law enforcement uses. Some of which could be search-and-rescue missions to handling deadly situations without putting innocent lives at risk. At the basics of all “smart” things is hardware. Every single hardware piece in these so called “smart” things is handled by a program, or software. Without software, hardware is considered a useless object. Software gives directions, controls, best pathing decisions and even motion detection. The goal of any software design is to try to closely replicate the decision making of that of a human brain. Humans think mathematically, physically, emotionally and even drastically at times. Emotions play a major role in a human’s life, but not that of a robot or piece of software. Software only deals with the best possible decisions and outcomes given certain circumstances. Software thinks mathematically and physically with lots of IF, ELSE-IF, AND, and even OR statements. Combining all of these with certain thresholds/limitations, a powerful virtual “brain” is developed that will be able to make decisions on its own. With software, there will always be bugging issues and lagging issues, so the main goal to any software engineer is to make coding efficient/simple and quick using the smartest algorithms and best designs.

Computer vision/robotics is a field that has been around for over 50 years. One of the first robots ever created was in 1961 by general motors. Even though the job of these robot was very simple (lifting hot molten steel and dropping into cooling liquid) it still made the lives of humans easier. Which is what software and computer vision is supposed to do. Computer vision should be able to replicate what us and humans can see and think. A.Q.U.A.L.U.N.G. will be able to scan a known environment, detect any and all possible unknown objects in this environment, and aim and fire a laser at the center of mass of that object using multiple detection/edge detection and processing algorithms. To make this possible, A.Q.U.A.L.U.N.G. must be extremely efficient and quick with all of its commands and code.

4.5.2 Human/Motion Detection Research

The main motion detection research came from reading many up-to-date other research projects along with reading multiple inputs from CVPR 2012 papers on the web, one of which was “Pedestrian detection at 100 frames per second” which was created by Rodrigo Benenson, Markus Mathias, Radu Timofte and Luc Van Gool in the University of Leuven, in Belgium. Another very important source was a snippet that was used was on www.axis.com under video motion detection (VMD) It gave an overall basic understanding of how exactly detecting motion in videos actually works and was an excellent place to start researching.

CVPR 2012 papers on the web which were read

1) “Pedestrian detection at 100 frames per second”
   Rodrigo Benenson, Markus Mathias, Radu Timofte and Luc Van Gool
   ESAT-PSI-VISICS/IBBT, Katholieke Universiteit Leuven, Belgium

2) Efficient Object Detection Using Cascades of Nearest Convex Model Classifiers.
   Hakan Cevikalp
   Eskisehir Osmangazi University
   Meselik Kampusu
   Bill Triggs
   Laboratoire Jean Kuntzmann

3) Real-time Facial Feature Detection using Conditional Regression Forests
   Matthias Dantone
   Juergen Gall
   Gabriele Fanelli
   Luc Van Gool1

4.5.3 Motion Detection

The information about motion detection is far from complete. Motion detection is simply the processes of finding or detecting a change in movement or x-y-z position of an object or target relative to its background. The three ways to detect motion are; electronically, mechanically and motion perception, which is only used by natural living organisms. There are exactly 6 ways of detecting motion.

- Infrared
- Optics
- Radio Frequency Energy
- Sound
- Vibration
- Magnetism

A.Q.U.A.L.U.N.G. will only deal with electronically detecting motion and processing motion using optics.

4.5.3.1 Motion Detection Algorithms

Motion detection algorithms are directly aimed at only detecting motion in constant video streams/frames. Of course what comes to mind right away is simplicity, accuracy, high frames per second, and ability to distinguish useful motion over non-useful motion. As far as the latter, A.Q.U.A.L.U.N.G. will be acting and performing in a known environment where there should be no, non-useful motion. So taking that into account, attention is turned only to accuracy, simplicity and high frames per second to meet the goal of the detection Algorithms for A.Q.U.A.L.U.N.G..

ISPY PTZ control: Ispy's PTZ control was something that looked like the last piece of the puzzle to A.Q.U.A.L.U.N.G.. They developed a genius PTZ (pan tilt and zoom) algorithm that works amazingly well with MJPEG video streams. Obviously, this requires the correct hardware, a video camera that has the capabilities to pan, tilt and lastly zoom. This allows very accurate detection and will be able to pin-point exactly where the "motion" is. As in figure 4.5.3.1-1, ISPY's PTZ detection streams directly to the server at that IP address and from there, is extracted and able to use processing algorithms upon. ISPY has implemented the circle in the direct center of the location in which the camera is facing and adds a line in the direction that the camera is moving. So with this, it will allow the user/operator to get a "heads-up" on where the camera will orientate next.

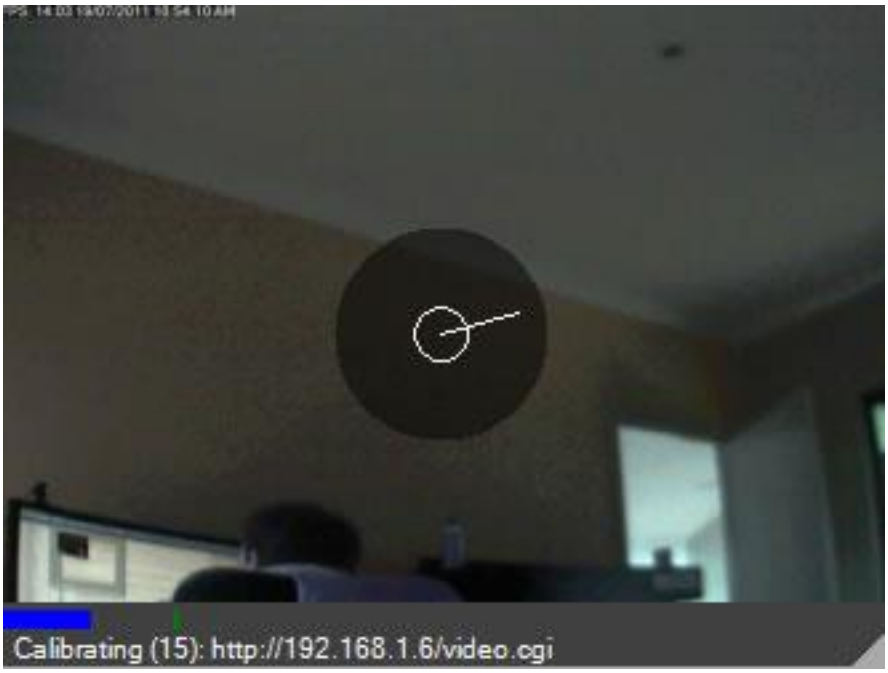

Figure 4.5.3.1-1: ISPY PTZ control. Permission Granted

Keeping into account that we will be having a wireless camera mounted upon the quadcopter, the detection algorithm must be very precise and accurate. The reason why choosing this ISPY algorithm is out of the question is because there are too many useless features that will come along with it, which will just make things overcomplicated. Again, A.Q.U.A.L.U.N.G. has to use simple, accurate detection techniques and algorithms to meet our goals and criteria. Tilts, and zoom, are completely useless for A.Q.U.A.L.U.N.G. and will only take up extra battery power that could be used for something else instead. The main goal is to simply detect any unknown objects in a known environment and process it using one or more of the many processing algorithms. Since this project is heavily relied upon the accuracy and simplicity of the detection, ISPY will not be used for our detection Algorithm.

AForge Source Libraries: Coming across the AForge source video libraries seemed to be perfect for A.Q.U.A.L.U.N.G.. AForge libraries provide lots of classes which can help attain many motion detection and even motion processing algorithms. Some of the many AForge motion detection algorithms are the following;

Two frames difference motion detector: This motion detection algorithm deals with looking at two consecutive frames and computing the amount of difference between them. This algorithm is the simplest and the quickest.

Figure 4.5.3.1-2: AForge two frames difference motion detector algorithm

Permission Pending

The only problem with this algorithm is its lack of accuracy. It is extremely quick, and simple to implement and use, but since its sole purpose is to simply detect a motion object, its accuracy fails short of what is required to calculate the center of mass of the human (mid to lower chest) and fire the laser at that point. For A.Q.U.A.L.U.N.G. we need a more accurate approach

Simple background modeling motion detector: AForge developed a more accurate way of detecting motion. Simple background modeling motion detector, which will from now on be mentioned as “SBMMD.” SBMMD may only be used in a known environment. One of the requirements and specifications of A.Q.U.A.L.U.N.G. is that it would be acting in a known environment, so right away; this detection algorithm seemed to be the one.

Figure 4.5.3.1-3: AForge simple background modeling motion detector algorithm

Permission Pending

Aforge's SBMMD algorithm focuses on finding the difference between a frame representing the background, "known-environment," and the current video stream. The coding and algorithm is very simple and quick. It uses techniques as training the algorithm to detect any changes in the frame with respect to the original, non-interfered basic background frame. The main reason for deciding to most likely use this detection algorithm is because as seen in figure 4.5.3.1-3 the highlighting is extremely accurate since the algorithm can instantly and easily detect the un-known object "human" in the current frame. This accuracy is much needed for A.Q.U.A.L.U.N.G.. Looking back up at figure 4.5.3.1-2 the highlighting is not as accurate as A.Q.U.A.L.U.N.G. would need it to be in order to do all necessary calculations and conversions.

Custom motion detector: It is also possible to use a combination of the previous two detection algorithms. Doing so the best of both algorithms are combined into one, very powerful, accurate and quick detection. First, the program and algorithm will be fed the original non-interfered known environment snapshot frame and the algorithm will always use this as a reference. As soon as motion is detected, SBMMD will be the first to accurately detect it. Once the target, "human" begins to move, AForge's two frames difference motion detector algorithm will come to life and keep a very accurate update on the current position of the target detected. A.Q.U.A.L.U.N.G. may simply have to use a custom combination of the two motion detector algorithms to combine into one very strong and accurate one.

4.5.3.2 Motion Processing Algorithms

Motion processing algorithms are simply algorithms that are focused on interpreting the raw data received from the detection algorithms and applying some sort of work upon them, for example; counting total number of moving object, tracking the moving objects and lastly, highlighting the moving objects. Of course the latter will be most useful for A.Q.U.A.L.U.N.G. because the goals were stated to simply detect, process, aim at highlighted object and fire laser. As for A.Q.U.A.L.U.N.G. it has been decided to use MJPEG format for all of the video streams needed to extract the raw data and process it. MJPEG seems to be the best decision as far as the video format of the stream, mainly because of the vast majorities of algorithms that can view, edit and make decisions upon viewing and interpreting the MPJEG data. Of course when looking at specific processing algorithms, there are certainly many options to choose from. The goal of A.Q.U.A.L.U.N.G.'s motion detection portion would be to find the simplest, "light-weight" easiest to interpret and code way of extracting the raw data from the stream, evaluating it and making decisions upon it. The following are some of the few processing algorithms from AForge motion processing algorithms that may just be possible to complete this goal;

Motion area highlighting: This is the basics of processing algorithms. The sole purpose of this processing algorithm type is to highlight the areas that were found by the motion detection the figures above all have this processing algorithm enabled into them, just to show the moving object in the frames.

Motion border highlighting: AForge's motion border highlighting motion processing algorithm may only be used and processed using the most accurate motion detection algorithm that was explained above. The most accurate being SBMMD and the custom motion detection algorithm. The reason for this is because this algorithm is extremely sensitive and updates very frequently. The following is a screenshot to show the accuracy and detailed border highlighting of a frame from the motion detection algorithms;

Figure 4.5.3.2-1: AForge motion border highlighting processing algorithm

Permission Pending

Grid motion area processing: This Processing algorithm seemed to be most interesting. The reason for this is because grid motion area processing deals with breaking the entire frame screenshot into a grid. The grid is filled with cells. These cells may be customized to the exact needs of A.Q.U.A.L.U.N.G.. We may increase the size of grids both horizontally and vertically with ease and it's not only extremely customizable, but also reliable and accurate. The main use for this type of processing algorithm is to show and record which of the cells in the grid had the most activity. Practical use, it may be used as security cameras, or even to record data on types of environments, for example, at a fork in the road, do most cars go left, or right. That is just one example of thousands that may be possible with grid motion area processing. This is a quick screenshot that shows exactly how the grid and cells work;

Figure 4.5.3.2-2: AForge grid motion area processing algorithm

Permission Pending

With AForge's grid motion area processing algorithm the user may edit the amount of cells each frame will set. This works very closely to how pixels work. Each single pixel can be on or off, 1 or 0, with multiple ranges of color combinations of RGB (red, green, blue). Red, green and blue are the primary colors when it comes to video processing and applications (gaming for example), as opposed to red, yellow and blue which are the primary optical colors. Each and every single cell in the grid may be recorded and stored later for more processing to understand more about which parts of the grid had the most action (which cells changed from original snapshot background).

4.5.4 Video Processing UML Class Diagram

The motion detection portion of A.Q.U.A.L.U.N.G. is extremely important; it is the basis behind all the processing, navigation and AI responses. Referring to figure 4.5.4-1 below one can follow the functions and procedures that the main computer follows to process each and every frame and detect motion.

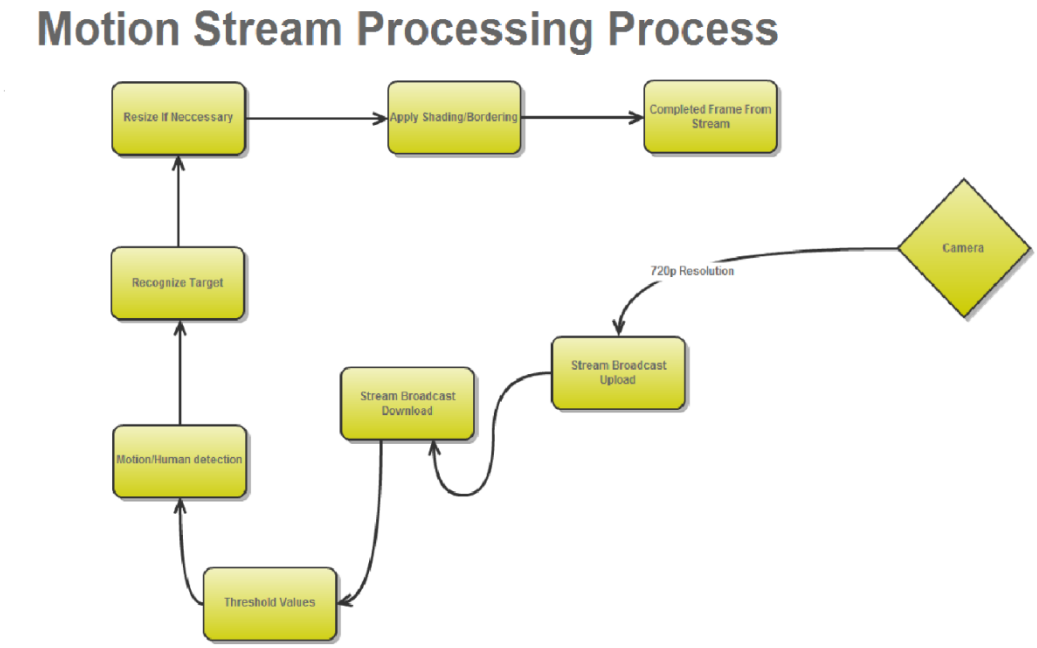

Figure 4.5.4-1: A.Q.U.A.L.U.N.G.'s Video Processing UML Class Diagram

Starting at the camera on the quadcopter, the first frame ever sent to the main computer will be that background of the main known environment. With that information, the computer may now know what do detect changes in. For example, if a person walks across the room, the detecting algorithms will understand that the person was not originally in the known environment and treat it as a target. The way this process happens is the following;

1) Camera uploads stream to offsite server, usually a website at a minimum resolution of 720p.

2) The main computer system will then access the stream that has been uploaded and will link to it in real-time downloading the video frames.

3) After the link is established and the stream is up and running on the computer then all necessary thresholds will be applied to the frame.

4) Once motion is detected within that frame/stream we move onto next step.

5) It is confirmed that this object is in-fact a human and mark it as a target.

6) Next, we may be able to resize the processed image to have it in the correct size that the processing algorithm may work with efficiently.

7) This is the most important step. Here, the motion processing algorithms go to work efficiently and accurately either shading or bordering in the target.

8) Lastly, the completed processed image/frame is ready to be sent on to the targeting algorithm to fire the laser at.

4.5.5 Wireless Communication

A.Q.U.A.L.U.N.G. is working entirely off wireless communication between the quadcopter prototype and the processing computer to make real-time updates and adjustments to each and every motor at an exact specific time. To do this, a very reliable, accurate and quick wireless device must be mounted onto the quadcopter to be able to send the data back to the main computer. This involves buying a wireless communication adapter.

4.5.5.1 Wireless Modules

Xbee 802.15.4 microchip:

XBEE provides, at www.digi.com which was an extremely helpful site which played a big part in the research phase of A.Q.U.A.L.U.N.G.. Having searched the many other sites and comparing and contrasting multiple wireless communication chips, XBEE's turned out to be the best for the job. It is extremely small, lightweight and accurate, operating at a 2.4 GHz frequency band. The range of the XBEE ZB is a maximum of 133 ft indoor, with an outdoor line-of-site range of 400 ft maximum. As far as the supply voltage, XBEE will be 2.1-3.6 VDC (volts direct current). The main reason for having a XBEE to control the quadcopter instead of the traditional R/C remote control is because an R/C remote control is very expensive and makes everything easy. The following figure shows the basic XBEE wireless microchip;

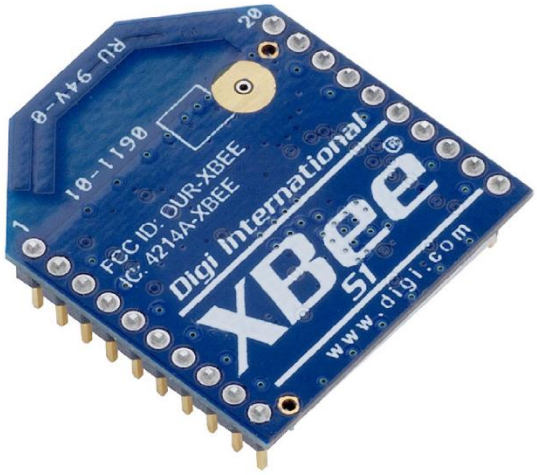

Figure 4.5.5.1-1: XBEE 1mW PCB Antenna (802.15.4) – Series 1

Permission Pending

For a total of $21.99 XBEE's 1-mW PCB antenna microchip is possible to have and mount onto A.Q.U.A.L.U.N.G. easily and simply. The reason it has been decided to go with the XBEE series 1 instead of another type of wireless communicator is because of the features and key specifications of this specific wireless chip. As far as features, some of the many that are included are the following; PCB antenna, cross-compatibility with many other XBEE and 802.15.4 XBEE modules and low efficient power consumption. A key-specification that A.Q.U.A.L.U.N.G. relies on is quick communication and data rate. XBEE completely takes care of this with their promised up to 115.2 kbps interface data rate.

Zigbee:

Another option is using Zigbee's model 802.11b/g 2.4 GHz wifi chip. Zigbee's main website helped compare and contrast many forms of wireless communications, whether it be Zigbee, Wi-fi, Bluetooth, UWB (Ultra Wide Band) or even IR Wireless. It so seemed that both UWB and IR wireless would out of the question for A.Q.U.A.L.U.N.G. because of their way over the top specs and costs. As far as the specified GHz range that A.Q.U.A.L.U.N.G. would need to work in, Zigbee, Wifi, and Bluetooth all operate in the 2.4-5 GHz frequency range.

Figure 4.5.5.1-2: Zigbee Wifi 802.11b/g chip

Permission Pending

After intensive research, it has been found that using a wifi chip has a high complexity, but even worse, a high power consumption. Power consumption is a very important aspect of A.Q.U.A.L.U.N.G. because many onboard devices and information must be processed very quickly while simultaneously keeping the motors running at at least hovering power capacity. Instead of buying multiple high top end batteries, it has been decided that A.Q.U.A.L.U.N.G. will use the resources that is achievable to work extremely efficient. Even though the wifi data rate can be anywhere between 11-54 Mbits/sec, which is very fast, it must be taken into account the amount of data the quadcopter's embedded chip will be sending is nowhere near that large. So using a Xbee 1MW PCB should be far more efficient, cost wise and power consumption wise for A.Q.U.A.L.U.N.G..

4.5.5.2 Camera

The use of a camera mounted upon the quadcopter will make everything possible as far as the motion/human detection. When looking at different types of cameras and whether to choose and onboard embedded camera, or an external camera we have to take into account a few vital important things first;

- Camera must be light-weight to not draw any unnecessary power from the batteries to keep the quadcopter in flight.

- Must have a high enough resolution so that the images may be extracted well enough and fed through the motion detection and processing algorithms.
- Camera must be able to transmit stream by up to at least 80 ft.
- Must not exceed a 5 volt power requirement.
- Must be strong enough to survive the impact from a drop of 5 ft., just in case testing goes wrong.

Onboard Embedded Camera:

Using an onboard embedded camera has its ups and down. First of which is its extreme lightweight. Very important for A.Q.U.A.L.U.N.G. because all extra weight will do is require the motors to use more power percentage to maintain a stable flight, and requiring more power will pull more power from the batteries. So to minimize cost and power consumption weight is a big issue. As far as the serial camera module, TTL, weight is not an issue. Referring to figure 4.5.5.2-1 one may see that everything is extremely simple and easy to use on the chip. A serial connection will help mount the chip onto the microcontroller simply and easily.

Figure 4.5.5.2-1: Serial JPEG Camera Module, TTL

Permission Pending

A few good features that come along with the serial jpeg camera module ttl would be low cost and low power. The cost being $59.00, ordered from microcontrollershop. The minimum power required to run this embedded camera chip will be 3.3 volts DC supply at 62 mA. The transfer rate of the JPEG pictures and raw data tops out at a speedy 1.2 Mbps. Built-in down sampling will help with the quick transmission of data along with low space requirements. Two different mode features come with this camera module, a low resolution, for quick viewing, and a high-resolution for storage/viewing/processing.

External Mounted Camera:

Coming across the X10 external outdoor camera widened the options of A.Q.U.A.L.U.N.G.. For here, low-cost/weight must be sacrificed in order to receive a clear, high definition workable stream. X10 seemed to be the perfect candidate. It leans towards the more expensive side by costing $129.99, ordered from www.x10.com. This camera will cost double the amount of that of the embedded onboard one. But with this extra price, comes more powerful features and lenses. The following is a picture and blueprint of the X10 mounted camera.

Figure 4.5.5.2-2: X10 Airsight Outdoor Wireless IP Camera

Permission Pending

As one may see, it is a fairly large camera with the dimensions being; height-3.25”, width-3.05”, length 7.5” with the antenna reaching 7.25”. This is the second option that A.Q.U.A.L.U.N.G. may have. This is an extremely powerful camera, which takes care of all the uploading of the stream in high definition and high resolution. The main frame computer can easily access the stream and begin its motion detection and motion processing algorithms on it.

5.0 Design summary of Hardware and Software

5.1 Hardware Design Summary

In the following section a theoretical combination will be discussed, these values and specs, although chosen with thought in mind, might not be the final product of A.Q.U.A.L.U.N.G. There might be unforeseen changes to the final product due to experimentation and its results, for example the pitch of the blade might theoretically sound good in this section, but when put together in real world applications it might not be the best choice for A.Q.U.A.L.U.N.G. Any changes that are found to be significant in the implementation of this project will be documented after final testing is complete, and a solid model has been established. The changes that were mentioned above will now be documented alongside the original document as to easily compare the final result with the original thought.

To start off with it is assumed that, the PCB together with the microcontroller used, will fit onto a 4 inch by 4 inch interface plate. The interface plate used was a 6 by 6 inch plate. So this set the dimensions for everything else that needed to sit on, in or below the interface plates. The microcontroller used will be an Atmel AT32UC3L064, with Inertial One which features a 3-Axis accelerometer from Bosch, a 3-axis gyroscope from Invensense, and a digital compass from AKM. This Atmel microcontroller was not used in the project instead an Arduopilot APM 2.5 microcontroller was used, which also had the same sensors. This package together with the PCB was less than 2 inches tall, however in order to give the electronics room the standoffs between the middle interface plate and top interface plate will be 2 inches.

The batteries for this project were situated underneath the middle interface plate and were LiPo batteries. Orginially thought to be, “…rated for 4500mAh – 65/100

c, with three of them put together they will be able to give the quadrotors enough power for an efficient flight." However, A.Q.U.A.L.U.N.G. ended up having two three cell batteries instead of just one three cell. Individually they were 2200 mAh.

What was originally thought to be motor specs, "The motors for the quadrotors will be four 1200 RPMs/volt kv rated motors, drawing approximately 11 amps. These motors are to be paired with propellers that are 8 inches in diameter and that have a 3.8 inch pitch." Figure 5.0.1 shows exactly what the pitch of a propeller is. However, the motors for the project ended up being four 1650 RPMs/V rating that could draw 17.3 amps maximum. The motors were paired with 10 inch diameter propellers with a 4.5 inch pitch.

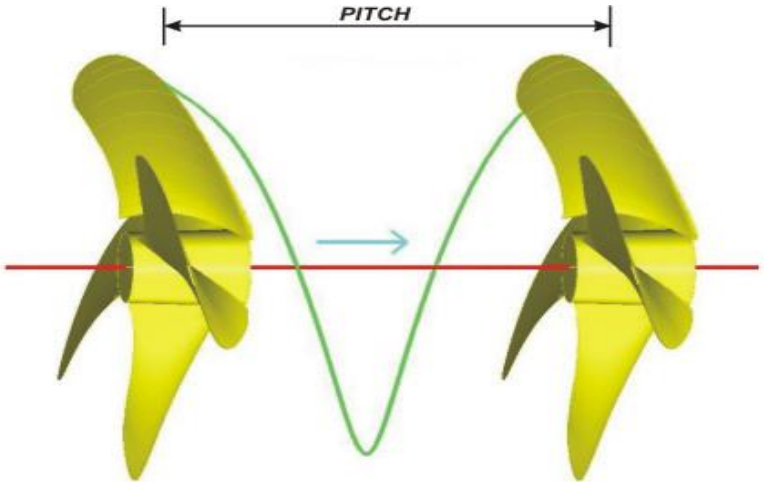

Figure 5.0.1

The pitch is the distance a propeller travels in one complete spin of 360 degrees. [1]

When vision was first pondered these specifications were thought to be exactly what A.Q.U.A.L.U.N.G. needed, "The camera for the quadrotors sight will be a webcam, the Logitech HD C310, which offers 720p video and 5 megapixel snapshots. It is relatively cheap and slender; however it will be stripped down to only its bare essentials." A webcam was not used in the A.Q.U.A.L.U.N.G. project; a wireless 2.4 GHz camera was used. It was powered by a 9 V battery that was mounted onto the quadrotor; this was not a problem as the quadrotor was powerful enough to lift this extra weight. This camera had a receiver that plugs into the computer which would be receiving the live stream from the camera.

For the laser gun and sensors, the same kind of idea that was used on the camera would have been implemented here. The LAZER TAG multiplayer battle

system would have been used; it consists of two guns and two sensors. The guns are enormously huge and bulky, since they are made to look like guns, and the sensors are embedded in cheap armor looking vests. The lasers would have been stripped from the guns and mounted onto the top interface plate, facing forward. While the sensors would have been stripped from the vests and placed on the front side and back side of the interface plates, as to simulate what the humans will be wearing. A senor on the front chest area of the quadrotor, and a sensor on the back area of the quadrotor. Lastly, the wires would have been stripped and connected to the appropriate component on the aircraft. Implementing of the laser gun feature and sensor were not achieved during the A.Q.U.A.L.U.N.G. project due to time constraints.

Legs for the quadrotors did not turn out to be such a challenge as they came attached to the frame purchased. Since the batteries were hanging from under the bottom interface plate, the legs needed to be able to shelter the batteries during landings. It was originally thought that, “The best bet for the legs will be two “sled” like components that will screw into the bottom plate.” However four legs were used to protect the batteries just fine. Figure 5.0.2 shows a very rough drawing of what the legs looked like.

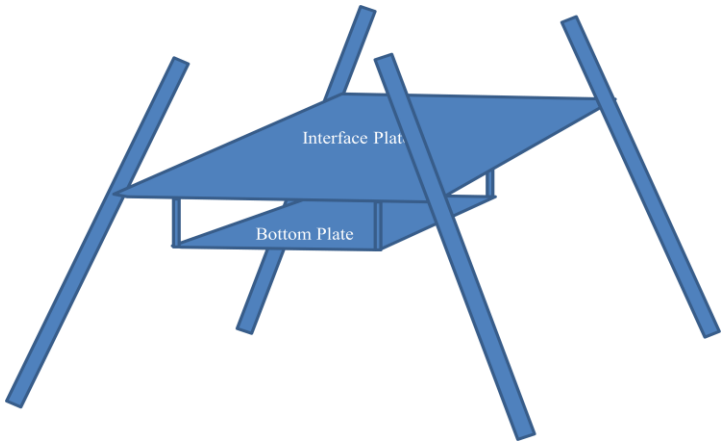

Figure 5.0.2

5.2 Overall Software Design and Class Diagram

The following is the overall top-level software class diagram that A.Q.U.A.L.U.N.G. will be implementing. Each device has multiple functions and variables which will be explained in detail.

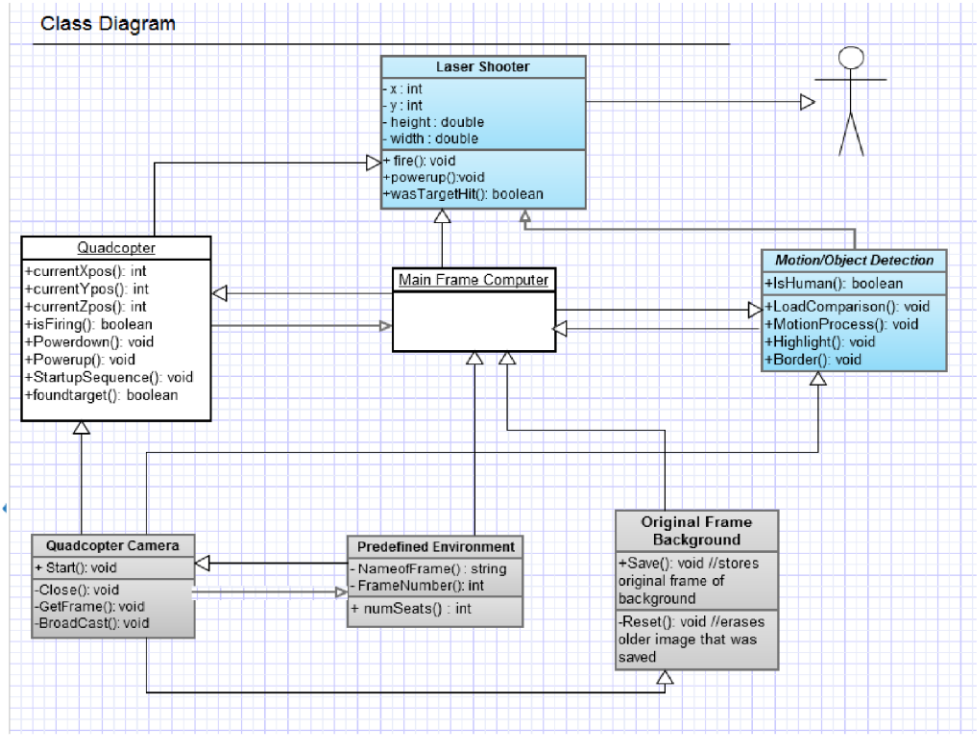

Figure 5.1-1: A.Q.U.A.L.U.N.G.'s Software Class Diagram

- Quadcopter Camera: The reason for this class function is for all the raw data streams and frames may be broadcasted and used in the predefined environment, original frame background and the motion/object detection algorithms.
    - Start(): void- This function will start up the camera from it's power off mode, no pictures or frames shall be recorded yet.
    - Close(): void – This function will shut down or power off the camera, at this time it has been assumed that all necessary requirements and needs have been successful or that the quadcopter is running low on battery power.
    - GetFrame(): void- The only and main reason of this function is to take one single snapshot for either the predefined environment or the original frame background information.
    - BroadCast(): void- The broadcast function will broadcast the stream to our upload server via the quadcopter's onboard wireless communicator or directly to upload site, which will allow for the

main frame to quickly receive a live camera feed off of the quadcopter.

- Predefined Environment: After the Quadcopter camera has completed the GetFrame function, it will then send that information into the predefined function in order to successfully store the known environment. Multiple images and frames may be sent to have a larger sample size, therefore increasing accuracy and successful rate of A.Q.U.A.L.U.N.G..
    - NameofFrame(): string- This function will allow the user to input a string to name the frame that has been sent to the predefined environment from the quadcoptercamera. Mainly used for organization and easy retrieval.
    - FrameNumber(): int- This variable will store the frame number in the order the quadcopter camera sends the data to the predefined environment class. Starting at “1.”
    - NumSeats(): int- This variable will be preassigned to show the maximum number of frames may be possible. May be implemented using an array of some sort or even a double array to show frame number as well.
- Original Frame Background: Once the quadcopter camera takes a screenshot and frame shot of the environment we know will be the background of the undisturbed area, it will then forward that data to this function which has two functions with it.
    - Save(): void- This save function simply saves the original frame or set of frames into memory on the main frame computer, this information will then be transferred multiple times between the motion/object detection function and main frame computer.
    - Reset(): void- This reset function will delete all prior raw data that has been sent to original frame background function. The main use of this function would be for software testing and making sure everything is lining up correctly and accurately and smoothly.
- Motion/Object Detection: The primary purpose of this function is to use the information given to it by the quadcopter camera and main frame computer, which is all the information needed to make calculations and decisions. Once this functions has the original frame background along with the predefined environment and all of the snapshots from the quadcopter and main frame computer, it may now apply any sort of detection algorithm that is decided to use, along with its respected processing algorithms.

- IsHuman(): Boolean- This Boolean simply stores whether or not the suspected motion is in-fact a human, and may be labled as a target to poass onto the MotionProcess() function. A simple TRUE/FALSE will be used here.
- LoadComparison(): void- In this function call the motion/object detection function will begin to load the predefined environment along with the original frame background onto a comparison sheet which will allow the software to then begin the motion processing algorithm, along with setting IsHuman as TRUE or FALSE.
- MotionProcess(): void- This is the meat of the whole code. Here, a specific, or multiple motion processing algorithm(s) may be applied to the frames in order to detect motion, after motion is detected, by using any of the algorithms, if in-fact the motion is a human, it will then trigger IsHuman() as TRUE, if not, then IsHuman() shall be labeled as FALSE. After this time, the frame will then either be shaded/highlighted in a certain color, or simply bordered in a certain color to apply for an easy comparison and visual of the motion. This motionprocess function will also help with the software specific testing later on.
- Highlight(): void- If this function is called, then the certain frame on which it was called upon will have the motion/target highlighted in. For this function the use of AForge's grid motion area processing algorithm will shine. To see what the frame will look light after the highlighting function is called upon, refer to Figure 4.5.3.2-2.
- Border(): void – Either the Border or Highlight functions will be called upon the frame after the motionprocess is completed. The reason for this is for easily detecting the motion and for error detection/correcting. As far as the border function, the use of AForge's motion border highlighting processing algorithm shall be used. Refer to figure 4.5.3.2-1 to see exactly how the border function will apply the borders to the specified frame snapshot.

- Quadcopter: This function will be the virtual quadcopter where everything from its current x-y-z positions to if it found a target to call the laser shooter functions. This function must work flawlessly because everything runs through this function. Little bugs and delays in the code will cause the quadcopter to run extremely inefficiently and wrongly. All of the code should use up-to-date techniques to lower run time and get the commands to the motors/cameras/lasers as quickly and swiftly as possible. The following functions will be in the quadcopter function routine.

- currentXpos(): int- This is simply the X coordinate with respect to the original X coordinate (0). Will be in the form of an integer, or possible a double if changed to use many decimals. Using a standard X-Y-Z 3-D plane as the point 0-0-0 will be in the exact center of the known environment room. For example, 1-0-0 will be 1 meter in on the positive X axis.
- currentYpos(): int- This is simply the Y coordinate with respect to the original Y coordinate (0). Will be in the form of an integer, or possible a double if changed to use many decimals. Using a standard X-Y-Z 3-D plane as the point 0-1-0 will be one meter on the positive Y axis and 0,0 on the X-Z axis.
- currentZpos(): int- This is simply the Z coordinate with respect to the original Z coordinate (0). Will be in the form of an integer, or possible a double if changed to use many decimals. Using a standard X-Y-Z 3-D plane as the point 0-0-1 will be in the exact center of the known environment room, except 1 meter off of the ground floor.
- isFiring(): Boolean- This function will be a simple TRUE/FALSE statement Boolean. True meaning that the quadcopter laser shooter is currently in the process of shooting and should not mess with any other functions until isFiring is turned back to FALSE (the quadcopter laser shooter has completed shooting and may go on to other tasks).This is important because the accuracy of where the target is must be spot on, and moving or ascending or descending the quadcopter may mess with the given command of the location to shoot at relative to the current quadcopter's currentXpos, currentYpos and currentZpos.
- Powerdown(): void- This function will power down the quadcopter, whether it be for power consumption or for recharging. While powered down, neither of the functions should be running and none of the onboard chips or cameras or shooter will be getting power.
- Powerup(): void- This function will power up the quadcopter once ready for flight or fully charged. This function and the powerdown function will be a big role in the software testing to make sure all the parts on the prototype are receiving power efficiently and exactly how long the quadcopter will be able to accurately run and do tasks from the power up to the power down.
- StartupSequence(): void- The StartupSequence function will first Powerup the quadcopter then it will begin all tests to make sure everything is up and running. First testing each of the motors at a

small power % so no lift it generated, but just to test to see if the motors are accurately getting power. Next we will send a signal to the laser shooter to shoot a test shot to make sure that is also running. Once the main processes of the quadcopter are tested then the quadcopter will be in its idle state and waiting for the commands to begin its mission.

- Foundtarget(): boolean- When the motion/object detection function finishes running and the motionprocess function accurately determines that there is in-fact motion and labels that motion as a target, then it will send a signal to the quadcopter function to set the Foundtarget Boolean to TRUE. With this, the rest of the sequences may run correctly.

- Laser Shooter: This function will be the virtual laser shooter and sensor. Before any signals are sent to this function, every prior Boolean must be correct, some of which must be, foundtarget, isHuman and isFiring. Once these three Booleans are all set to TRUE, then the laser shooter function can be accessed. The laser shooter function has multiple variables and functions, each having a specific task or representation. Here are the details of each of the functions and variables;
  - x: int- This is the x coordinate of the located motion in the known environment, this is important because the laser shooter must know where to orientate the laser in order to correctly shoot at and successfully hit the target.
  - y: int- This is the y coordinate of the located motion in the known environment, this is important because the laser shooter must know where to orientate the laser in order to correctly shoot at and successfully hit the target. Notice that there is only an x and y coordinate and no Z. The reason for this is because all motion and targets are assumed to be at ground level and there is no reason of implementing another dimension, because this will just make things more complicated.
  - Height: double- This variable is a set number that tells the laser shooter the average height of males and females (targets/samples) so there is no specific need to implement the z coordinate. Any x and y combination will always have the same set z coordinate.
  - Width: double- This width variable is a double number that is the average width of our samples and targets. The reason for this is to accurately locate the center of mass and sensor on the human target (mid to upper chest).

- Fire(): void- This function will take into account x, y, height and width and with all of this information and knowing the exact location of the quadcopter with information currentXpos, currentYpos, and currentZpos, the laser shooter may accurately aim at and fire the laser at the specified location. The fire function may be set to fire one or more multiple consecutive shots at the same location if need be.
- Powerup(): void- This void function will be the powerup function. The powerup function will start the laser shooter and fire a shot to make sure everything has power and is up and running correctly. Will be used in software specific testing later on.
- wasTargetHit(): Boolean- This function will return whether or not the target has been hit according to the sensor on the human target. If target was in fact hit, this Boolean will be set to TRUE. If the target was not hit, the Boolean will be set to FALSE. A.Q.U.A.L.U.N.G. will be aiming for a 50% hit rate, so there will be two additional overall integer counter that will keep track of the hit/miss ratio so we can do specific testing and able to increase the accuracy of the prototype if need be.

- Main Frame Computer: This will be the main() function of all of the code, therefore, naturally will be the most important. There will be lots of processing happening at once, so it is vital to success that this main frame computer is powerful enough to handle all checks/code/commands of the entire program/prototype. In this main() will be the main start() function and end() function, possibly a debug() function to help with software and hardware error checking and testing.

6.0 Project Prototype Construction

6.1 Parts Acquisition and BOM

Item	Where bought from	Quantity	Price per Quantity	Total Price
Quadcopter Frames Prebuilt		4	$30	$120
Computer for Processing	Newegg	1	$500	$800
Laser Gun and Sensor	Ebay	2	$23	$46
Brushless Motor		16	$10	$160
Propellors		4	$3	$12
Microprocessor		4	$50	$200
Webcams	X10	4	$15	$60
Accelerometer/Gryometer		4	$100	$400
Battery		4	$30	$120
Miscellaneous Electrical Components		Unknown	Unknown	Allocating $150
			Final Price of all parts	$2,068

6.2 Final Coding Plan

When the time comes for coding, it would be optimal to get the main components all working individually before ever thinking about merging them together. Of course once the software and coding is all completed, refer to the software specific testing to intensively test each individual component and software. Once the specific testing is completed, then the prototype may be fully assembled and then thoroughly tested with all the components attached. As far as the final coding plans, it would best to first make sure the navigation system and battle commands are up and running perfectly, that is highest priority. Next, the motion detection/processing algorithms must be working flawlessly before they can be tested on the in-flight quadcopter. The motion detection and motion processing algorithms will be coded using C, C++ and parts of C# to efficiently and effectively allow for the accomplishment of motion detection. For the object

detection and motion detection. Firstly, the motion detection algorithms will be coded and when they are tested thoroughly then the motion processing algorithms will be designed. Once both are completed and fully tested, the computer vision portion of the prototype can be considered completed, and more time may be focused on other portions of the quadcopter design and coding. For each main function, all the individual functions will be completed in the order of importance first. For example, for the quadcopter portion, the powerup() and startupSequence() functions will be coded in order to test all the electronics and if there are any problems they can quickly and easily be solved without wasting too much time on other things. The order in which the motion functions will be completed will be the following;

1) Quadcopter Camera

2) Original Frame Background

3) Predefined Environment

4) Motion/Object Detection

The reason for this specific order is because each one is a prerequisite for the other. There is no way that the original frame background snapshot will be saved and stored without the quadcopter functions and variables all working correctly. As far as the navigation/quadcopter/laser shooter, the order will again be;

1) Quadcopter

2) Navigations/AI battle commands

3) Laser Shooter

The reason for having the laser shooter being coded lastly is because, first all of the motion detection and processing algorithms must be working correctly, along with the navigation and artificial intelligence controls. Once those are working simultaneously, then the laser shooter may be coded using the processing algorithms and the current position of the quadcopter and many other factors.

7.0 Project Prototype Testing

7.1 Hardware Test Environment

Just as important as testing the quad rotors, is picking the environment in which to test them. All individual components went through their own form of testing indoors under the supervision of the group members. Testing for these components was conducted in a room with plenty of workspace, tools, and accessibility to power outlets.

The finished quad rotor was first tested indoors as well. It was assembled and tested at a dining room table at a low motor speed with a group member holding down the center in case of any accidents, to first check and see that everything powered up and worked together. After it was determined that all subsystems of quad rotor were fully functioning and working together, testing was still conducted at the table with the quad rotor tethered to the table.

Though the intent of the project was to have the quad rotors fly in an indoor capacity, such as a laser tag facility it was hard to find a space large enough and available as often as needed to test in. The laser tag facility that the group originally hoped to test in, did not lend out their facility for the group to conduct testing in. On one occasion, the group was able to book a large hall in a nearby church and hold a practice session in there. The hall was an open room with at least 50 foot high ceilings. The group laid down comforters and sheets on the ground, so that if the quad rotor made an unexpected landing nothing would be impacted by a crash. Though this was an ideal testing location considering the quad rotors were meant to fly in an indoor capacity, availability for the space and during the group members schedules was hard to arrange.

On most occasions, hardware testing of the finished quad rotor, was conducted outside of one of the group member's houses. Strings were tied to the legs of the project and then either tied to a tree, or held by group members, while the quad rotor was flown. These outdoor conditions were not conducive because of varying weather conditions and inability to hold testing at night for lack of visibility, but happened to be when group members were available to meet.

7.2 Hardware Specific Testing

To make sure that the quadrotors work, the group has to first make sure that each of the individual electronic pieces work. To ensure that each item works, the group has to first conduct some hardware specific testing. They have to test each part's ability to perform as expected and make sure that all the electronics weren't in any way damaged before the group received them. If any parts were

damaged it could cause a delay in the build schedule while the group waits for replacement parts, or could cause a below average performance for the entire project.

To test the PC the group plans on making sure that the BIOS recognizes all components present in the quadrotor system.

To test the webcams, the group is going to hook them up to the computer and test that what should be seen by the webcam is being seen. This test is also going to check that the webcams are continuously streaming video, as this project needs to be done in real-time and needs a constant video sent back to the processor/ main computer.

A battery test will be conducted on the four LiPo batteries that are purchased. The group will do a battery life test to check what is the actual life-cycle of the batteries.

The motors that are bought will also need to be tested. They will hooked up to a power source and then tested to see that they are running at the right speed or else new motors will need to be looked into.

7.3 Software Test Environment

Having an environment that A.Q.U.A.L.U.N.G. can constantly be tested in and improved upon will be vital to the success and accomplished outcome. An indoor space out of the elements with proper ventilation to keep all the components and computer parts cool and running efficiently would be optimal. In the very beginning test phases, a soft landing grid spot may be implemented just in case the quadcopter runs out of battery without properly letting the main computer know so the power off sequence can be initiated. The reason for this is to avoid the breaking of parts just in case something goes wrong. When testing the beginning phase of the computer vision portion, it will be ideal if only the camera and detection algorithms be tested alone and separately before mounting the camera onto the quadcopter. The reason for this is to pinpoint any and all bugs and be able to fix them individually before putting all of the parts together. This is a common strategy used among many debugging issues and problem solving, which is called “Divide and Conquer.” Much easier to test and debug devices and programs separately without them being all intertwined with other components and other software. For starting to test the motion detection, will only have one stable “unknown” object in a room at a time. Start small, then work up to having that target move and possibly multiple targets at once.

7.4 Software Specific Testing

The software specific testing section comes directly from the software requirements since A.Q.U.A.L.U.N.G. will be tested according to the requirements that have been stated. It shall be explained exactly how that specific requirement shall be tested.

- Computer
    - The raw captured stream from the camera on the quadcopter must be opened up and processed on the main computer system.
        - The way to test this is to first upload the stream and make sure it is able to be downloaded properly and viewed live.
    - Must have enough RAM and hard drive space to effectively store the stream and original background known environment frames for the comparison between the new detected object and original frame.
        - Store sample background pictures and frames to make sure there is no lag or no bugging issues and enough space to properly store everything required to do all the necessary calculations and processes.
    - The computer must be quick enough and have a fast enough processor to run the code on the detected image and send a response back to the quadcopter in less than 40ms.
        - Once the motion is detected, the process algorithm will be applied to the frame and the time will be recorded when the quadcopter gets the response back. Must be less than 40ms
    - The computer must be quick enough and have a fast enough processor to run the artificial intelligence, navigation and battle commands on it and send the commands to the quadcopter to have it react quickly and swiftly.
        - Test navigation points and battle order will be given to the quadcopter and we will record data on how the quadcopter reacted and responded. It must be quick enough to be successful for A.Q.U.A.L.U.N.G..

- Human Recognition Detection System
    - The system must accurately and quickly detect humans through detection algorithms.

        - First the frames will be sent of the original background, to test the detection software to see if a human is found or not. This is where the IsHuman() Boolean will be set to TRUE or FALSE accordingly. If the detection algorithm sets the IsHuman() Boolean as FALSE when in-fact there is an unknown object, then this piece of software will not be working correctly and will have to be updated and further coded.
    - Will find all unknown specified objects in a known environment by using all or some of the detection algorithms.
        - Multiple unknown objects will be put in the known environment to test whether or not the detection algorithm can notice and realize them. At most two targets should be able to be labeled as unknown objects.
    - Must implement one or more than one motion processing algorithm onto the detection algorithm.
        - To test this portion of the software, an unknown object frame will be fed into the motion processing algorithm and tested to see whether or not it properly shades or borders in the respected object correctly, accurately and efficiently enough.
    - The system will then target the highlighted/bordered target and fire a laser at the center of mass of it.
        - To test this certain software portion, the frame of the processed image will be fed into the main computer and then the command to locate and fire this image will be sent to the quadcopter and then the laser shooter. At this moment, the quadcopter will remain stable and the unknown object target will remain stable and the fire laser function shall be implemented. If the target is hit accurately, should be 100% hit rate, then this portion of the software is considered good to go.
    - The power system must be sufficient enough to properly supply power to the camera on the quadcopter and enough power to broadcast the stream efficiently and cleanly.
        - To test this portion, we will limit the amount of power going into the camera to the very minimum. If for some reason the camera won't broadcast the stream efficiently, due to lack of power, we know there must be changes made to the power settings and PCB.

- The software shall be able to recognize only a maximum of two unknown objects (human targets).
    - First this will be tested with a single target, then the software will be implementing two targets. Both of which should be processed and shaded or bordered in correctly and accurately.
- The coding for the recognition and detection system must be 99% bug and error free to get as close as possible to a 50% successful hit rate with the laser.
    - For the testing of this, the testing of the individual components talked about earlier must “pass” the tests. Once they pass, the individual software components will be merged and tested together. According to the divide and conquer technique, this should be the most efficient way in testing all of the components needed to have the success of A.Q.U.A.L.U.N.G..

- Navigation/Artificial Intelligence
    - Quadcopter’s response to navigation commands must not exceed 4 inches of leeway.
        - Test X,Y, Z coordinates will be given, then team members will physically measure how accurate the quadcopter battle order adjustments were with a standard ruler and measurement techniques.
    - Navigation commands and orders must be sent to the quadcopter within 50ms of being sent to prototype.
        - A test will be performed to measure the exact amount of time needed to send a command to the quadcopter from the navigation code and upon analyzing the data, adjustments may be made accordingly.
    - Artificial intelligence must be smart enough to avoid any obstacles that may be present in the 3-D space
        - 3-D obstacles will purposely be placed in different places around the known environment room to test how the quadcopter reacts and moves. Ideally, nothing should be hit at all, and should remain at least 6 inches away from all other objects or walls.
    - Navigation and artificial intelligence commands and procedures must be able to reach the quadcopter at a maximum range of at least 15-20 feet from main computer

- To test this software requirement, the quadcopter will purposely be sent 20 feet away from the main computer to test if the commands will still reach the quadcopter via wifi/zigbee connection.
-

7.5 Final Test Environment

After all was said and done it only made sense to retest the quad rotors. The group had to check and make sure that all their hard work was able to produce a working quad rotor. Though the original intent was to have an indoor test environment, that vision was unable to be seen. The group was left with no choice except to conduct their final testing under the same conditions their original testing had been conducted in. That test environment being the backyard of one of the group members where they kept the quad rotor tethered at all times and were at the mercy of night fall and any kind of weather conditions.

For their very last test, the senior design final presentation/demonstration their test environment was also outdoors. The group chose to conduct this demonstration outdoors so the quad rotor would have enough height to fly and in case of any mishaps, would not cause damage to itself if it came down without warning. They chose the arboretum behind the Harris Engineering Building on the UCF Main campus.

7.6 Final Specific Testing

The phase of final specific testing is for testing the project's effectiveness and making sure that all the subsystems cooperate in a way that is progressive toward fulfilling the proposed goals of the project.

During the final specific testing phase, the group made sure the quad rotor could pass a start up test, all motors were to be running at the same speed and all components were to turn on, including the camera which was powered via its own power source.

After the start up test was conducted, the group did a hover test. The quad rotor had to lift off from its stationary position and hover at a comfortable altitude. Following the hover test, the quad rotor was tested to see if it could tilt/rotate in specific directions.

Table: Flight test check sheet

	Yes/Pass	No/Fail	Notes
Start-up Test	Y		
Hover <8 sec	Y		
Tilt Left	Y		
Tilt Right	Y		
Fly forward	Y		
Fly backward	Y		

Also the camera was tested to see if it the image displayed was correct.

8.0 Design Considerations

8.1 Environmental Impact on Quadrotors

A true game of laser tag was never played with quadrotors from the A.Q.U.A.L.U.N.G. project, these following evaluations are all hypothetical in nature.

To ensure that the quadrotors were going to be able to perform too their highest potential, the environment in which they would roam had to be taken into account. For this particular project, the quadrotors would be playing a game of laser tag indoors, in a known area, with known obstacles. However since that really isn't the end of the line for the applications of this technology, other, more extreme and different environments will be examined for their potential impacts on the quadrotors.

To start with, an examination of the laser tag arena was done, starting from what it is made out of. The floors of the laser tag courses can be either one of two things, hard or less hard. As simplistic and uneducated as that sounds, it breaks down into hard things being tile, wood or cement as the flooring, and less hard things being carpet or rubber flooring. If it is necessary to word it more eloquently, surfaces such as tile and cement will not absorb most of the impact if the quadrotor were to fall or crash onto it. These surfaces would, according to Newton's third law if I remember correctly, push back onto the quadrotor with an equal force that the quadrotor hit it with. What about carpet and rubber? Seeing as it's not the 70s and shag carpet is no longer in style, a thinner carpet is most likely to be used and under most carpet is cement. So although a layer of carpet or rubber might absorb slightly more of the force than tile or wood, the quadrotor would still take a significant amount of the force when it hit the ground. So it was very important to equip the quadrotors with protection for the weaker parts, such as the electronics. Even with electronics covered up, if it was a big enough fall, structural damage to the frame and propellers could result from the crash. All these things in mind, there would have to be a sweet spot where budget and durability had to meet and there would of course be risk of breaking due to an accident.

Another element of the laser tag environment would be humans themselves. The human element was the most unpredictable and threatening element in any type of environment. It would be impossible for two quadrotors to run into each other,

unless something malfunctions, because the AI would not let them run into each other. The AI knew where each quadrotor was and therefore could make the appropriate decision on evasive maneuvers. However the quadrotors weren't the only thing on the playing field, if a quadrotor and a human are both at full speed and turn the same corner, a collision could happen with nobody intentionally at fault. This collision could have resulted in a couple of things happening, first the propellers would hit and could cause some major damage could ensue. If the propellers go, then the quadrotor would plummet out of the sky and hit the ground causing the damage specified above. Some possible ways to avoid such an accident from accruing would have been to fly the quadrotor around at above normal human height, so if a human and quadrotor crossed paths at a high speed without realizing it nothing would happen.

Human hazards were not all unintentional like an accidental around the corner collision. It was entirely possible that when put in the course with humans, especially kids who are excited about the game and with their adrenaline pumping, the human opponents could attack one of the quadrotors with the laser gun itself, a swat from the human hand, intentional collisions, and anything else that could be considered malicious intent. The damage to the quadrotor would depend on the type of action taken by the human, it could have ranged anywhere from just messing up the dead-reckoning, to completely damaging the electronic components or any other vital piece of hardware. The best way to avoid such an attack would have been to try and make the AI recognized such an attempt and know how to take evasive action.

Air-conditioning could have played a factor in the environment of the quadrotors. If a stiff enough wind were to come from a vent, it could have pushed the quadrotor to a certain degree without the quadrotor or central computer knowing it. This could have caused the dead-reckoning to be off, and navigation could have been effected. Worst case scenario for this would be the AI telling the quadrotor to run into a wall because it thought the quadrotor was somewhere else. On the lighter side, it could have just given the microcontroller a harder time when trying to stabilize the aircraft.

If this technology were to be taken and put to use in something other than the entertainment world, a whole wide range of environmental factors would come into play and make the job of creating an efficient aircraft that much harder.

Let's say the quadrotors were to be put into use in the military, doing the exact same thing but with real weapons, not laser tag guns, and built a lot bigger to be able to hold these guns. Well, first off, what's the climate like? Are the quadrotors being deployed in Iraq where temperatures can climb to 120 degrees

Fahrenheit? Or are they going to be in Siberia where the average temperature in January is negative 13 degrees Fahrenheit? The quadrotor will have to be able to withstand extreme hot or cold temperatures, and the biggest threat temperature imposes is on the electronic components. So when picking electronics components this would have to be taken into careful consideration.

If it were for military use, which branch of the military would use them? It would be safe to say that all branches would be using this technology outside, so waterproofing in case of rain is a must. Another threat for all branches of the military is enemy artillery, so some type of shielding would be a must. For the navy, the parts would all have to be rust proof as salt water left to do its thing will erode any metal on the quadrotors. For the air force, the quadrotors could be dropped from an aircraft that is moving at mach 3 and 30,000 feet in the air. This could require a parachute to deploy until the quadrotors reach a safe altitude where they can start their search and destroy. For the army they could have the ability to Medivac a wounded soldier off the front lines and to safety. The possibilities are endless for the quadrotor technology in the military, just as long as the right adaptations are made to fit the environment they will be working in.

A less lethal but just as important application for quadrotor technology would be emergency services. Firefighters could use them to search a burning building for survivors; this would eliminate the need for humans to do the searching, however, of course, the quadrotors would have to be redesigned for such an extreme environment. For an environment that could possibly reach 2000 degrees Fahrenheit, special electronics would have to be used, as well as very efficient cooling devices. A special shield would have to be used to protect all of these components, even the propellers. For the quadrotor to be able to see in the dark and through smoke, special cameras would need to be used, possibly not even a camera at all but some sort of radar mapping device since visibility is low and thermal cameras would not be able to decipher between a human and fire.

For search and rescue the AI would also need to be changed, from knowing its whole environment to being able to map an unknown environment as it goes. This is very important as a collapsed building could look nothing like the blueprints of the original building. Mapping the whole environment is also important for the human rescuers as it provides a detail description of what they will face once they enter the building, decreasing their time in the building and increasing the efficiency of the rescue.

8.2 Assembly of Quadrotor

Assembly of Quadrotor

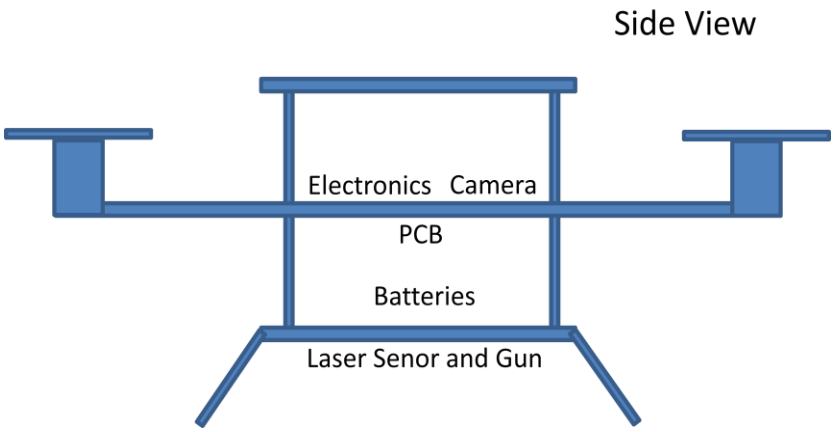

Figure 8.2.1

The Assembly and Disassembly of the quadrotors for A.Q.U.A.L.U.N.G. started with the parts list and having a visual concept of the overall layout as shown in Fig. 8.2.1. The frame, brushless motors, propellers, microprocessor, IMU, camera, battery, laser gun, laser sensor and printed circuit board. To start the assembly, a base was needed; the base from which everything else will be built upon is the frame. The frame can either be prebuilt or assembled out of pieces as follows. There will be two different parts, the interface plates and the arms. The interface plates are the platforms in the middle of the quadrotor that will hold and protect the microprocessor with IMU, the camera, batteries, laser gun, laser sensor and printed circuit board. The arms are the outward protruding lengths of frame that will hold the brushless motors and propellers.

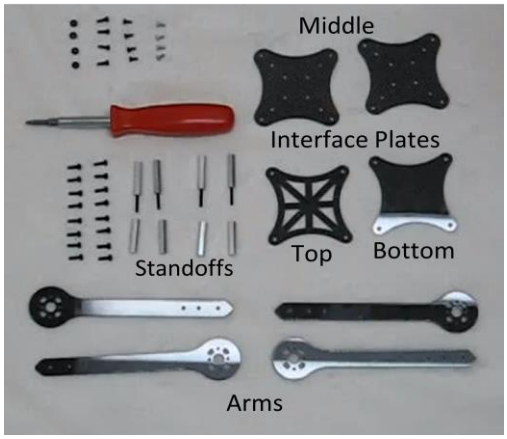

Figure 8.2.2


Figure 8.2.3

For this example a NOVA quadrotor frame will be used for visual reference. Starting with the middle interface plates and one arm, place the inner edge of the arm in-between the two middle interface plates and screw the arm into place. Continue with the other three arms, the arms should be sandwiched in-between the two middle interface plates. Now that a middle platform has been established the stand offs can be added to the frame. The standoffs provide support for the top and bottom interface plates, while creating enough space for electronics and batteries to be able to sit comfortably on the frame with protection. There are male and female standoffs, the males will go on the top side of the frame, the screw will penetrate through the middle interface plate and through to mate with the female standoff on the bottom of the middle interface plate. Once all four standoffs are in place, it should look like there are four pillars coming out of the top and bottom of the frame. The bottom interface plate now needs to be secured onto the bottom standoffs; it is as simple as matching all four corners of the bottom interface plate to the standoffs and screwing the plate down. Once this is completed the frame should be able to rest on the bottom interface plate without wobbling or moving, so place the frame down on the table and match up the top interface plate with the top standoffs, and secure the top plate down. Now a custom modification will be needed as a laser gun and sensor will hang from the underside of the bottom interface plate. Legs will be needed to protect these components when landing; four more standoffs could easily be used for this purpose. Four holes would have to be drilled in the bottom plate, the standoffs treads would be fed through the plate and a nut would secure them into place on the top side of the plate. There are many of different options one could use for landing legs; it all depends on the size of the camera being used and size of bottom interface plate. Once this is completed the frame is now ready for all the other components of the quadrotor.

This frame was just a reference to show where to start when building a frame. Any frame purchased can follow this guideline to put together the frame. However some words of wisdom for a project looking for a frame. Make sure to get a frame that attaches everything with bolts, for example the first frame used for A.Q.U.A.L.U.N.G. was a Turnigy Talon Carbon Fiber Quadcopter Frame, which broke because it used clamps to secure the motors to the legs. Over time the clamps wore down the leg rod and finally snapped it in half. Recovering was a matter of finding a better frame to go with for the project, which came with the purchase of a frame that bolted the motors to the legs. It is also a smart idea to get a frame that has four screw holes for the motors on the legs. The Talon only had two which caused problems when the quads crashed during testing as they would bend. So no two point motor mounts, as the motors come with four points to mount, so go with four.

The next step in assembly would be the electronics; there are three major pieces of electronics on the quadrotors, the microcontroller, the IMU and the PCB. These three pieces were assembled outside of the quadrotor then put in all together. On the bottom layer of these three was the PCB, this was doing the power dispersion from the batteries and was a stable platform that will be able to hold the microcontroller and IMU. The best way to connect the microcontroller to the PCB mechanically would be to have some sort of mini standoff that could offset the microcontroller from the PCB and then having the microcontroller secure to the standoffs, this would create air flow between the PCB and microcontroller for heat dispersion while still allowing a secure pairing of the two electronic components. This design was implemented on the first frame however the PCB was separated by standoffs on top of the microcontroller. For the second frame, the frame came with enough room to mount the PCB on top of the batteries then have the MCU on top of the interface plate.

Next was to connect the IMU to the microcontroller, this is done best by reading the instructions that came with both the microcontroller and the IMU. Since both parts have coordinate systems in place already, all that would need to be done is figure out which axis is which and make sure that the IMU's x axis is lined up and pointing in the same direction as the microcontroller's x axis. This approach was for the Atmel MCU, however Arduopilot came with the IMU already attached to the MCU.

The final step for installing the electronics onto the frame was to secure the PCB to the frame; this was made simpler by taking the top interface plate off of the frame for the installation, then putting it back on after. Once the top plate is off, screw down the PCB to the middle interface plate and reassembly the top interface plate, the main electronics are now assembled to the frame.

A good place to move on to now would be the battery installation. The problem, if one really wants to call it such a thing, was that the batteries will have to secure to the frame whilst in flight yet be easily accessible for removal when they need to be charged. This ruled out anything permanent for installation, and also ruled out screwing the batteries down, as it would have been too difficult and time consuming to take off every time the batteries need a charge. Solution? One easy way to get around this problem would be to install Velcro straps into the quadrotor. Bolting the Velcro straps to the underside of the middle interface plate, so that the batteries could be wrapped up close and tight to the underside of the plate, would secure them and make it easy to take off when charging was needed. Also there is no need to worry about bolting to the middle interface plate as there are two of them, so the screw would be in-between both of them and not penetrating the other side, thus causing no harm to the PCB. This option could have been more useful if only one battery was used; however two batteries were used on A.Q.U.A.L.U.N.G. making them too heavy for Velcro. The second frame came with a bottom interface plate, which the batteries sat on quite comfortably, with room on top for the PCB to be secured down.

Next up would have been the laser gun and sensor, mounted on top of the top interface plate it truly depends on the type of laser gun and sensor being installed. The important things to pay attention to when installing the laser gun and sensor are that the gun is aiming forward, so once the forward direction of the quadrotor was established via the IMU, make sure this was the direction you shoot at. The direction of the sensors is up to the team designing the quadrotors, whether there is one or two, if they were on the sides or front and back.

Camera mounting was the last step before connecting all the wires and installing the motors with propellers. The camera was on the middle interface plate of the quadrotor and again installation depends on the type of camera used for vision. As with the laser gun, it was very important to make sure the camera is installed facing the forward direction, so the quadrotor can move forward, see something in front of it, and shoot it.

The motors and propellers are installed last because they are in a much more fragile position, so it reduced the risk of damaging them while mounting and installing all the other central components. The most important part of mounting motor was making sure that the orientations were correct.

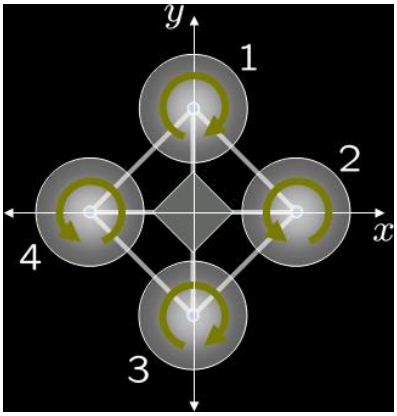

Figure 8.2.4

As Fig. 8.2.4 shows, the motors and propellers have to be installed in specific directions in order for the aircraft to be stable and be controlled by the software. Motors one and three had to be spinning in the same direction, and motors two and four had to be spinning in the same direction, yet two and four's direction had to be opposite the direction of motors one and three. This, as explained earlier in the paper, cancels out the torque in the yaw direction (rotation around the z axis). The motors should come with mounting brackets and screws with nuts to attach themselves onto the frame. Once the motors were mounted onto the frame it was time to wire them, now is when you want to test out which direction the motors are spinning, it might be helpful to put the propellers on if it is hard to tell which way the shaft is spinning. Once you have established which way the shaft is spinning if it is the correct way, than you are done, if it is not spinning in the correct direction, just switch the power leads and your motor should now be spinning in the opposite, correct, direction. Now check to make sure the propellers are on correctly, if the motor was on and pushing air downward, creating upward thrust, than the propeller is on correctly. If it was blowing air upward, simply take off the propeller and flip it, then reattach.

Now everything on the quadrotor should be assembled and ready for flight!

8.3 Maintenance of the System

Maintenance of the A.Q.U.A.L.U.N.G. system was a very important part in the success of the program. With such complex and sometimes expensive electronic

devices, fragile components in the motors and propellers, and the dynamic environment in which the quadrotors were performing, an organized system for pre-flight and post-flight maintenance checks becomes essential. Along with these flight checks, a routine disassembly to check hard to reach components could have also benefited the quality and lifetime of the quadrotors. The quadrotors were not the only parts to the system that needed to be checked and maintained, the central computer system needed to stay at the top of its game in order to execute calculations and send instructions to the quadrotors. The following section will discuss steps for completing thorough maintenance on the system.

Before every flight, even though it was time consuming and a pain, all screws, bolts, and connective devices should be checked to make sure they were secured. During flight, vibration occurs from the dynamic movements of the quadrotor and the relentless spinning of the motors. This vibration could cause connectors or screws to become loose, more often than not this loosening does not results in parts flying off the quadrotors, but it could have resulted in unwanted movement or shifting during flight which could cause damage to fragile electronic parts. If the propellers become loose, it could cause a wobble in the circular motion of rotation weakening the lift capabilities of the propeller. Again more often than not this will not cause the quadrotor to crash but it could have put an unwanted strain on the batteries as it would have taken more power to lift the aircraft.
Checking to make sure that the batteries are fully charged and ready for flight or a game of laser tag was important as well. Even though the quadrotors would have had battery power checking abilities, it would have been disastrous for them to fail and have the quadrotor run out of juice in the middle of the course. Not only could they have crashed, creating a possibility for extensive damage to its system, they could be stranded in the middle of a game of laser tag. And with a team of humans looking towards the sky for on coming opponents, not much attention would be placed on watching ones step, creating a possibility of a collision between the human and downed aircraft.

Once these two provisions have been taken care of, the last pre-flight check would have been to make sure all systems are go. Checking to make sure the quadrotor was talking to the computer and the computer was talking to the quadrotor, checking to make sure that the laser was working; the sensors were working and the camera was able to see. Then it was aggressive tactical flying from then on.

Post-flight checks would not have been as intensive as pre-flight checks since, well, everything pretty much gets checked before the flight. However, a post flight

check should have consisted of a quick observation of all components on the quadrotor. A head to toe inspection in order to look for any damage that could have occurred during flight, because if anything were to be damaged it was imperative to have it repaired before the next flight. A couple key areas to look at were the propellers, checking for any nicks, chunks or cracks from hitting something. The frame to make sure everything was straight, as running into something could have bent it making it harder to control. Both the camera and laser gun/sensors since they are not being protected by the interface plates. Again, it was not as thorough as the disassembly would have been, just a quick glance at everything to make sure nothing extreme happened during flight.

The disassembly was a way to make sure everything on the quadrotor was in working shape. The disassembly should follow the assembly of the quadrotor, just of course, in reverse order. So staring from the propellers, demount them, examine for cracks or nicks, and make sure they aren't bent or warping. Following the propellers would have been the motors, check the motor to make sure that the shaft wasn't bent or crooked at all, also the leads on the motors can wear down from connecting and disconnecting over time, so making sure that the connection from motor to power supply was as solid as possible is important. The camera was mounted on the quadrotor so making sure no damage had occurred from a possible rough landing or collision was important as well, of course checking the connection between the camera and microprocessor was just as important as the connection between an eyeball and the brain, so upkeep on these was essential. The laser guns and sensors were next, for the gun and sensors the connections needed to be inspected, but it's not just the connectors. The sensors can be affected by scuff marks from collisions, dirt, dust and anything else that could get in the way laser so the gun and sensors need to be cleaned and taken care of if they are going to perform well. Moving on up, batteries needed to be inspected for a couple things. Of course the common theme of every component, the connectors needed to be inspected, but after that the charge capability needs to be measured. As a battery gets later on into its life, it hold less and less of a charge until finally it will not be able to keep a charge at all, so maintaining a record of how your battery was operating is never a bad idea. Depending on the type of battery in use, different characteristics of the battery can cause failures, for example if the battery had acid in it, acid stratification could have occurred, knowing what kind of battery is operating the quadrotors and their weaknesses would help keep you ahead of the game in terms of maintaining the batteries. The electronics would have taken a little bit longer to inspect and it was not just visual inspection needed, measurements of output voltages and currents should have been taken. Applying the right voltage to the inputs of the PCB and microcontroller, then measuring the outputs and

comparing them to spec sheet values, or experimentally calculated values, will ensure that all the electronics are properly working, giving confidence in using them in flight mode. The frame was the last thing in the disassembly that needed to be inspected, this was more of a visual inspection, unless stress testing is available, to make sure all four arms are straight with each other, a level could have been used to aid in this process. Making sure there are no cracks, dings or damaged pieces in essential to having a strong base for the rest of the components to sit on. Once this was complete, following the assembly instructions will put the quadrotors right back into flying order.

The last component of the system that needed to be maintained was the central computer. The brains of the operation should be maintained by upkeep with the computer software, updating any programs being used to control the AI for the quadrotors. Not much to really say about the computer, other than keeping the system cool with fans so none of the electronics over heated, most of the central computer comes down to whether or not the processor is fast enough to keep up with the task at hand.

Keep up with all the maintenance needed for the A.Q.U.A.L.U.N.G. system and it will greatly increase the chances of having a viable product for years of entertainment purposes.

8.4 Risk assessment breaking parts

Within A.Q.U.A.L.U.N.G. there was a lot of risk associated with the project as a whole, risk being the level of complexity and chance of something going wrong. A.Q.U.A.L.U.N.G. was an attempt to combine several advanced topics into one fast moving dynamic system. Breaking this project down into subsystems was definitely a strong strategy for trying to make this project successful, but it could also be broken down into the same subsystems and analyzed for failure.

Starting with the AI controller, this part of A.Q.U.A.L.U.N.G. makes all the decisions regarding where and what the quadrotors do. With that said, it is then easy to see how this was the brains of the operation and if one thing goes wrong within the AI the whole system could fall apart. The most critical of the tasks it had was the pathing, pathing is basically telling the quadrotors where and when to go. If the pathing gets screwed up in any way the AI could tell the quadrotors to fly into a wall and not think twice about it. Assuming that the pathing for the system was working correctly, the quadrotors would be flying around just fine

navigating the course efficiently, however new data isn't being taken in from the image processor. This could be disastrous for one reason, the human element that cannot be predicted or calculated. With no vision coming into the AI controller, the quadrotors would not be in danger of running into each other or walls, since these are still being accounted for, the danger would be running into a human who is running through the course. There would be no tactical maneuvers being sent to the quadrotors from the AI because the AI wouldn't even know that there was a new "object" in the flight path.

The power system could be compared to the heart of the A.Q.U.A.L.U.N.G. system, if it doesn't pump out power than nothing in the system would run. Worst case scenario, the batteries just stop working, whether from running out of juice or a possible broken wire connection, it was very important to do the pre-flight inspections to protect against mishaps while in the air. Other possible failure derived from the power system would be incorrect power distribution, which would not be an easily fixed error. This error would stem from incorrect design of the PCB, whether wrong analysis were performed which would lead to incorrect values of the regulators or amplifiers. Or it could also possible for the theoretical values obtained from the equations to be off of the real world experimental values which could cause wrong voltages and currents to flow through components on the quadrotors. If the wrong voltages get put into components there's a chance that the parts could react normally, overheat or as Dr. Weeks likes to put it, produce the magic smoke. The parts could act normally due to a tolerance range specified in some parts spec sheets, for example the operating range could be from 1.6 volts to 3.3 volts, with a absolute maximum input voltage of 5.0 volts. So if the voltage regulation was targeted for 3.0 volts, but it was attenuating the voltage a little too much down to 2.5 volts, it would still be okay because its within the operating voltage limit. However, if the voltage regulator isn't properly regulating and it's putting out 5.0 volts, the part could overheat if prolonged use occurs. And if the regulator is putting out 6 volts, someone is going to see a magic show.

If flight control sounds pretty important to quadrotors, that's because it was. Two major things could have gone wrong with flight control that could have resulted in catastrophic damage to the quadrotors, stabilization failure and navigation failure. A failure in stabilization would have meant a certain crash for the quadrotors, without being able to stay in the air nothing about this project would work. There would be no laser tag, no AI control, no camera vision, because the quadrotor would be on the floor in pieces. Failures in stabilization could range anywhere from the quadrotors just spinning in circles till it crashed, to taking off, turning upside down and slamming into the earth. There is no way to guess how much

damage would have occurred, because just like the aircraft our guess would be unstable. The second major failure could have occurred with the navigation of the quadrotors, due to error built up within the dead-reckoning. If the quadrotor did not know where it truly was, it would be sending a false position to the AI, and without knowing it's doing wrong the AI could tell the quadrotors to go to the next position, which could be a wall. And the quadrotor itself, without knowing it is doing wrong would gladly fly into the wall, because it had total trust in the AI. So in order for this partnership to work out well for both parties, there needed to be accurate information coming from the quadrotors to the AI controller.

Within the image processing unit a couple things could have gone wrong, not recognizing humans or false identification of humans. While some of these errors would not have been as disastrous as the flight controls could have been, these errors would cause the project to be unsuccessful. If the quadrotors were not able to properly identify the enemy than they would be flying around either shooting at inanimate objects or not shooting at humans, which would turn the game of laser tag into flying target practice for the opponents and the game would lack any challenge what so ever. The potential for serious damage which could occur from the image processing not working would be collisions. Not only would the quadrotors not be shooting at the humans, they would not be aware of their presents making it impossible for the AI controller to tell the quadrotors to avoid a head on crash with the enemy. This of course can lead to a range of damage to the aircraft.

Risk assessment has outlined the importance of the cohesiveness within the A.Q.U.A.L.U.N.G. system, the only way the project would have been considered a success was if all the subsystems work together and work as they should. With this being said, a reality set in about how challenging this project was. However, what is life without a good challenge or struggle, somewhere within that battle between hopelessness and triumph, that's where engineers have the most fun.

9.0 Administrative Content

9.1 Milestone Discussion

Every idea has to have some beginning, some spark that catches fire and turns into a dream that needs to be executed, sometimes these dreams are for projects with a pathway of tasks that need to be completed so the project can be accomplished. This pathway of tasks can be thought of as milestones, checkpoints reached, check marks off the project's to-do list, all of them important and necessary for completion and ways to verify that progress is being made and things are on track. The idea for this project started one night when one of the group members watched a TED talk given by Dr. Vijay Kumar: "Robots that fly… and cooperate."

Dr. Kumar works at the University of Pennsylvania where he studies the coordination and control of multi-robot formations. His team has built quad rotors that are capable of sensing each other and forming teams and executing tasks such as overseeing construction and surveying disasters to name a few. Dr. Kumar's speech first talks about large scale unmanned aircrafts, which is where his idea for his small aerial vehicles first came into play. These unmanned aircrafts were still being controlled by someone remotely and he wanted to make something that could do things on its own. He wanted to create a vehicle that was truly autonomous and able to calculate motion and movement by itself. His team was able to program the quad rotors they built so they could calculate how to navigate obstacles, and how to sense each other spatially to form ad hoc groups to conduct small tasks as a group. It was while watching this TED talk that one group member was turning the idea over and over in his head and suddenly came up with the idea to use quad rotors to play laser tag against people.

Figure: Dr Kumar's quad rotor

On September 4, 2012 the team that would work on this idea came together during a senior design class with people that had a common interest in the idea of this project's mission. One computer engineering major and three electrical engineering majors, one of the electrical engineering majors with more than a basic knowledge of writing code, met and exchanged names and contact information to start the ball rolling and discuss what exactly they were trying to accomplish and at a high level how they planned to do it.

A week later the group met again to discuss more details about the project. They discussed how to divide up the responsibilities for the different project subsystems and drew up block diagrams to give a pictorial view of the schematics of the project. These block diagrams detailed the basics of how the 4 subsystems interacted with each other, the 4 subsystems being: power, artificial intelligence, flight control and computer vision. The group also discussed what parts would need to be bought and compiled a list with a rough estimate of the cost of the everything needed including a set aside amount for miscellaneous items to cover any incidentals, small consumable parts and tools needed. The group put together an initial report to document everything they discussed and included a budget, the block diagrams they drew up, and a summary to explain everything.

The next milestone the group reached was meeting their project manager, Dr. Samuel Richie. They scheduled a time to sit down and discuss the goals of their project with him. They gave him their report prior to their scheduled visit so he could review, formulate questions and make comments. When Dr. Richie met with the group he gave them a few words of wisdom and also suggested a few people to contact that could possibly be interested in helping. He also had a few samples of parts that he let the group take a look at to see if it was something they would be interested in incorporating into their design. As the project manager, Dr. Richie gave the okay to move forward with the plans as they had originally intended and told them should they have any concerns, issues, needs that they should contact him.

The next major milestone was to assess and realize with the budget that they had estimated that finding sponsors would be beneficial. The group sent out a proposal for the SoarTech sponsorship and starting reaching out to family, friends and acquaintances. The people at SoarTech however didn't feel that this project was one they would fund so the group continued to contact whatever resources they could. A few personal acquaintances of one of the group members offered to sponsor $500 and to pay for the motors as they were interested in the outcome of the project. Some contacts were also able to offer themselves as mentors that could help provide oversight, guidance, and helpful tips along the way. Some resources were also able to provide names of other companies and other individuals so the list of possible sponsors kept branching out and getting bigger and bigger.

Considering the mission of this project, to build quad rotors that would play laser tag with humans, it only made sense to reach out to a nearby laser tag facility.

The group contacted Hard Knocks, a laser tag venue located not too far from campus in Oviedo, FL. They sent out their request for possible access to their playing field during off hours to test and practice the quad rotors. The group also made their mission known and the value that the project could have to the Hard Knock family once the project was completed. Hard Knocks initial response was of a concern with the safety to their customers but the group is sure no harm would come to any person involved. The Hard Knocks family never replied back with a positive response for the group and so the project continued without that support or viable testing location.

The next major milestone was to iron out the details of the project, including the major milestones to be reached, the testing environment for both the hardware and software, a bill of materials so they could start ordering parts, etc. The group took all of this data and all the diagrams they had previously drawn up and combined it in a report for their project manager and also for another other party that would later on be interested in the background workings of this product. The report was submitted in early December 2012.

After turning in the report to Dr. Richie, the group awaited notice back from some individuals and companies they had contacted for funding and support. They had contacted some related companies and possibly interested individuals in the hopes they would pique some interest and possible receive some help, whether it be monetary donations or even just someone to donate some parts. They were hoping for a positive response to help cover the costs but they didn't have such luck. After realizing that the remaining budget was left to cover from their own pockets the group met to discuss the budget again and review their bill of materials.

The group met one day and discussed all the different parts they needed and decided that as their budget wasn't as extensive as originally anticipated, they would build just one quad rotor to achieve their mission instead of the fleet they envisioned. During this meeting they reviewed different websites and tried to find the best deals. After looking at different sites and their different shipping rates the group starting adding parts to their online shopping carts and buying parts.

During this time, a major step was conducting their CDR presentation in front of their senior design class and Dr. Richie. The group met to discuss their presentation slides and practice. They presented their progress and concerns to the class, and later on met with Dr. Richie to discuss their project progress.

This meeting with Dr. Richie was an interesting one as the group really hadn't received any parts in except the frame and battery charger. Some of the parts were being shipped internationally and seemed to be delayed in arriving. Dr. Richie was not very pleased as the time was ticking away and the group didn't have much to show. He informed the group they would need to meet with them in a week to make sure they were getting everything they needed done.

Within that week, the group received most of the parts necessary to have put together a working quad rotor. It was a major milestone they had reached because they were able to finally put together everything they had been planning. To make sure that everything was running effectively the group ran hardware tests of the parts to make sure everything was received in a working condition.

They worked tirelessly for a week to get everything mounted and up and running to be able to show Dr. Richie they were not just a bunch of slackers. They also had to build the code necessary to get the quad rotors up and flying of their own accord. They did some initial all system testing to make sure that everything was working together not just separately. A few simple motor movement tests were conducted to check that nothing would go haywire when the quad rotor was turned on. The group tested what different motor speeds would achieve what levels to determine what a good lift off and hover speed would be and also did some testing on start up and cool down procedures so the quad rotor didn't just plop down out of the "sky." A lot of troubleshooting was done during this time period as wacky things were happening when the power was turned on, and it was found that some of the motors were not as efficient as others. The trouble shooting led to minor changes and a better understanding of the path forward.

The night before their meeting, a fire started and incinerated one of the gator clips connecting the power to one of the 4 motors, so during their presentation they were only able to show 3 of the 4 motors running. They presented to him anyway which was still a powerful quad rotor tethered to the floor.

After meeting with Dr. Richie, the next major milestone was to finish the assembly of the project and achieve flight stability. The group conducted countless flight sessions and conducted final testing to make sure their quad rotor was able to be presented to their committee. During one such flight session, two days before the final presentation, the frame broke and the group scrambled to find a store nearby that could supply them with a new frame since they didn't have time to rebuild the old frame and recalibrate everything. With less than 48 hours the group was able to redesign the mounting of their quad rotor and get it up and flying and stable again. Milestone reached to say the least.

The last milestone the group had to reach was to conduct their final presentation. It was an interesting experience for all parties involved as something funny happened during the demonstration portion of their presentation. While they had been conducting almost all their final testing outside, testing was always done with the quad rotor tethered. During their final presentation, which was located at the UCF campus behind the Harris Engineering building, their quad rotor flew untethered and stable. It flew beautifully and everyone seemed impressed by the group's accomplishments but the wireless connection got lost during the first couple minutes of the flight. During the testing, the group had made the fail safe command to be to fly back "home," which was the only known GPS location. This location happened to be one of the members houses where they did all the testing, so upon losing connection the quad rotor began to fly home during the demonstration. In retrospect the story is quite comical, but the group was not

able to recover the copter, so during Senior Design Showcase, which occurred a week later the group was only able to show the broken frame and the remaining spare parts they had plus tell a funny story to any interested passerby.

9.2 Budget and Finance Discussion

When the group got together it was evident that this project would be a tad bit pricey. The hardware necessary to make this project worthwhile would cost a little more than any one person was willing or able to dish out. So initially, a proposal was sent out for the SoarTech scholarship. The project proposed fit all the criteria necessary to receive their $1000 sponsorship but unfortunately the group's idea for quad rotors to play laser tag was trumped by another group that wanted to make something else.

After reaching out to family, friends, coworkers, anyone that was willing to listen about the idea the group received $1000 and a promise from an interested friend to purchase the motors.

In the early stages of the project the discussion came up whether or not to buy prebuilt frames for the quad rotors. While it would be cheaper to buy parts and assemble the quad rotor frames themselves it would also create even more work for the group and would lead to additional testing being done to assure that the frames were stable enough and built correctly. It just made sense that the group should spend a little more to buy prebuilt frames and not deal with the hassle of trying to add onto their list of assigned tasks to be completed. The group had envisioned building a fleet of quad rotors but as their financial resources and time line were limited, a single quad rotor was built instead.

As a result of the quad rotors not being remote controlled, it was necessary to consider buying a computer for processing. This computer would be the main controller and handle and process the code necessary to achieve all the tasks at hand. The computer needed to be of a substantial processing power and was decided that it should be an intel core i7-3770k Ivy Bridge. The motherboard is a reliable brand with an LAG1155 socket, a Z77 north bridge, SATA 6Gb/s, USB 3.0, PCIe 3.0 x16, and has an ATX form factor. It has at least 16GB of RAM with 9-9-9-27 timing or better and at least 60GB SSD HD. It also has an 850W power supply with 80 PLUS Bronze or better certification. For Wi-Fi, the most cost effective of the following options was be chosen: included on motherboard, PCI slot, and separate router. Once all these specifications had been chosen the computer alone came out to $1000.

Since the objective of the project to was to play laser tag, it would follow that the quad rotors need to be able to detect their environment and the possible targets that should come into their path. This is where the webcam and wireless microprocessor come in. The webcam takes video of what is in the view of the quad rotors, and relays this message back to the main controller to detect whether or not there is a target in front and if so to fire at it.

Of course there is no way the project would work without power, so a battery would be necessary to power all the electronics present in the system. After plenty of research on different battery types, as can be seen in the power system section of this report, the best possible battery type was that of a lithium-ion polymer battery also known as a LiPo battery. Along with a battery being essential, motors are as well. With these being quad rotors, four motors per assembly were required and were provided for by one of the sponsors along with a few spares for just in case instances.

The group though they originally intended to make a fleet of quad rotors, had decided when they didn't receive as much funding as anticipated, that making a single quad rotor was sufficient. A single pre-built frame was ordered and would have been the only frame they were to purchase except that two days prior to their final presentation, the frame broke. As a result the group had to go out and find a hobby store that sold just a frame alone, and this is why the budget calls for differently priced frames.

Table: Budget for Project

Item	Quantity	Price per Quantity	Total Price
Quadcopter Frames	2	$30, $100	$130
Computer	1	$1,000	($1,000)
Motors	12	$10	($120)
Slowfly electric propeller 1045R 4-piece set	4	$3	$12
Wireless Microprocessor	1	$150	$150
Wireless 2.4 GHz camera	1	$20	$20
Wireless 2.4 GHz USB receiver	1	$40	$40
Zigbee	1	$80	$80

ESCs	4	$7	$28
11.1V 2200mAh 25C LiPo Battery	2	$15	$30
PCB	1	-	-
Miscellaneous	Unknown	Unknown	Allocating $150
		Total	$640

While the group was able to receive $1000 in monetary donations and had the PCB and motors donated, their budget was still higher than what was donated. All additional costs incurred by the project, including miscellaneous items needed to assemble the quad rotor were paid for out of the group members own pockets.

9.3 Sponsors

When considering the cost of this project it was easy to see that finding some monetary assistance would be ideal. In an effort to raise funds to help ease the financial burden on the group, a number of individuals and companies were contacted to see who was willing and able to lend a hand, a donation, or even just some words of wisdom.

To fund the project, a proposal was written and sent out to SoarTech. SoarTech was offering a $1000 Sponsorship for a project that had a significant software component and they promised to give preference to groups using a form of artificial intelligence in their system. This project fell perfectly into SoarTech's categories for acceptance so all of the group members resumes were attached and a letter was submitted stating why they should sponsor this project and what the project aimed to accomplish. Unfortunately this project was not accepted for this scholarship.

Hard Knocks, a laser tag facility located in Oviedo, was also contacted because of the potential this project has for their business. In the initial contact it was also asked if their facility could be used to conduct some testing. Their initial response was that they like to help students but they thought our idea would be a little cumbersome for them as a business, both because it would be difficult to allocate

time to use their facility and also because the only way they could take something on is if they could ultimately leverage the project. They didn't believe that this aerial design had a safe practical application in their customer environment. If their safety concerns could be met then they wanted to be contacted again, so a petition is in the works to get signatures from individuals that say they wouldn't mind playing and would waive any liability to Hard Knock.

John Enander sponsored $500 to the project. He is a long time personal acquaintance of one of the group members and has always had an interest in the future of that individual. When he heard of the group's project he was happy to assist and be of any support that he could offer.

Tim Parker, who happens to be the father to one of the group members, also was able to sponsor $500. Considering the nature of the relationship, it should go without saying that he had an interest in the outcome of the project and was happy to help the funding of parts for completion.

Jeff Moler is another personal acquaintance of one of the group members. He happens to be an engineer for Viking AT, a company that has been creating innovative mechatronics for years. Viking AT made the first piezoelectric valve and associated controls, and works with customers to create and implement technologies for locks, valves, switches, relays, pumps and compressors to name a few items. They also harvest energy from mechanical vibrations to usable electric charges. Moler is also interested in the future of the group member and also the outcome of this project. When told about this undertaking he volunteered to purchase the motors.

Sponsorship does not always come in the form of monetary donations. Sometimes sponsorship comes in the form of a mentor that can offer advice, guidance and a push in the right direction.

Dr. Avelino Gonzalez has lent himself out to be a mentor for the project. He has a research lab and is a professor at UCF with some interests in artificial intelligence, automated diagnostics and knowledge-based systems.

10.0 Operations Manual

Due to the fact that much of the software will not initialize without having a quadrotor connected, and the quadrotor built for the project having been lost during the final presentation, this manual will only demonstrate the use of the components that will still partially function.

The first step is to connect the override controller to the system. The A.Q.U.A.L.U.N.G. project used the MotioninJoy driver to connect a Playstation 3 controller to the system. Any joystick or driver combination may be used so long as it provides at least four axes of analog input. For the A.Q.U.L.U.N.G. project the left stick provided control over the pitch and roll axes of the quadrotor, the right stick provided control for the thrust, and the analog shoulder triggers provided control over the yaw axis. To ensure that the controller is working properly once connected via USB to the central control computer, open the MotioninJoy application and click “Enable”.

Next, open the Mission Planner software. Once it has loaded press the “Connect” button and wait for connection. Then select the “Actions” tab.

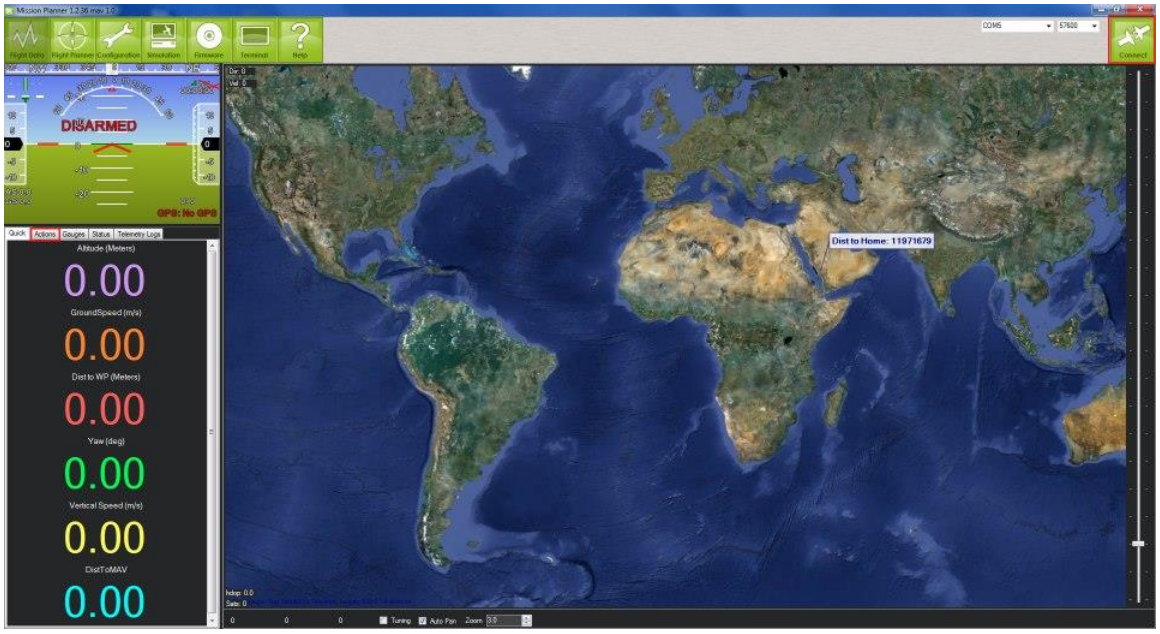

Figure 10.0.1 Opening Screen of Mission Planner

Once under the “Actions” tab, click the “Joystick” button.

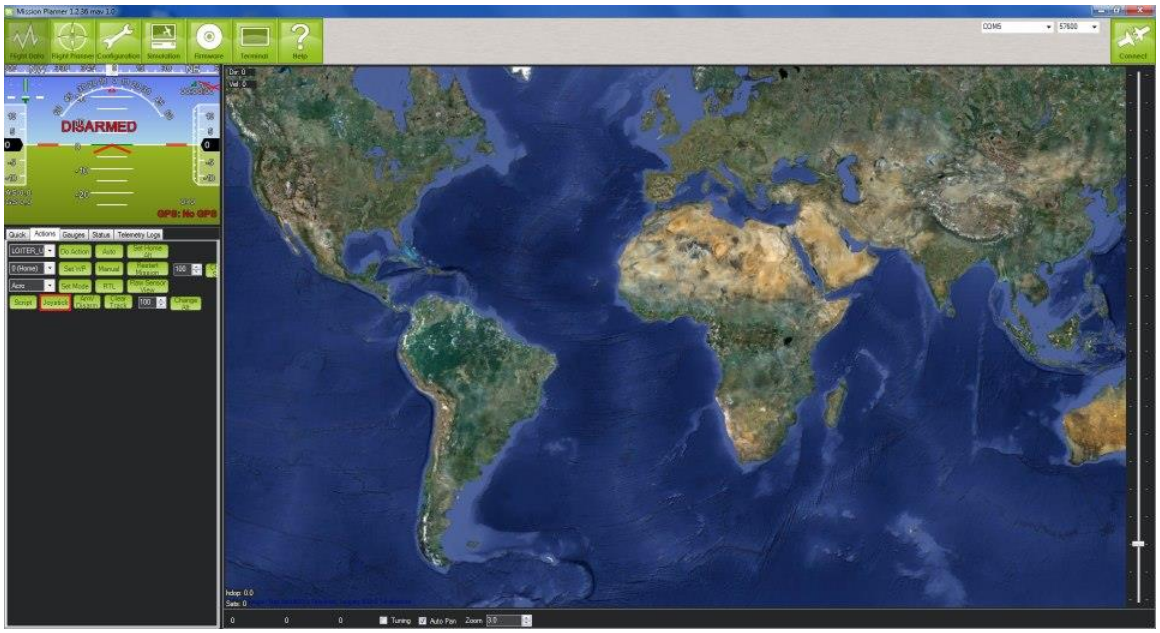

Figure 10.0.2: Screen with “Joystick” button highlighted.

This will open a new window. Ensure that each axis and button is responsive and verify that all axes and buttons are set correctly. Once this is done click on the “Enable” button and minimize the window. (Keeping the window minimized allows for rapid changes if an axis is inverted or some other minor problem arises.)

Figure 10.0.3: Joystick Window in Mission Planner

Next hold the throttle all the way in the “off” position and make sure that the quadrotor is in “Stabilize” mode. While continuing to hold the throttle “off”, click the “Arm/Disarm” button. Slowly ease up on the throttle until it is sitting in the neutral position, which is just shy of enough power to take off. Switch the quadrotor into the “Alt Hold” mode by pressing the “Square” button on the controller and increase the throttle slightly.

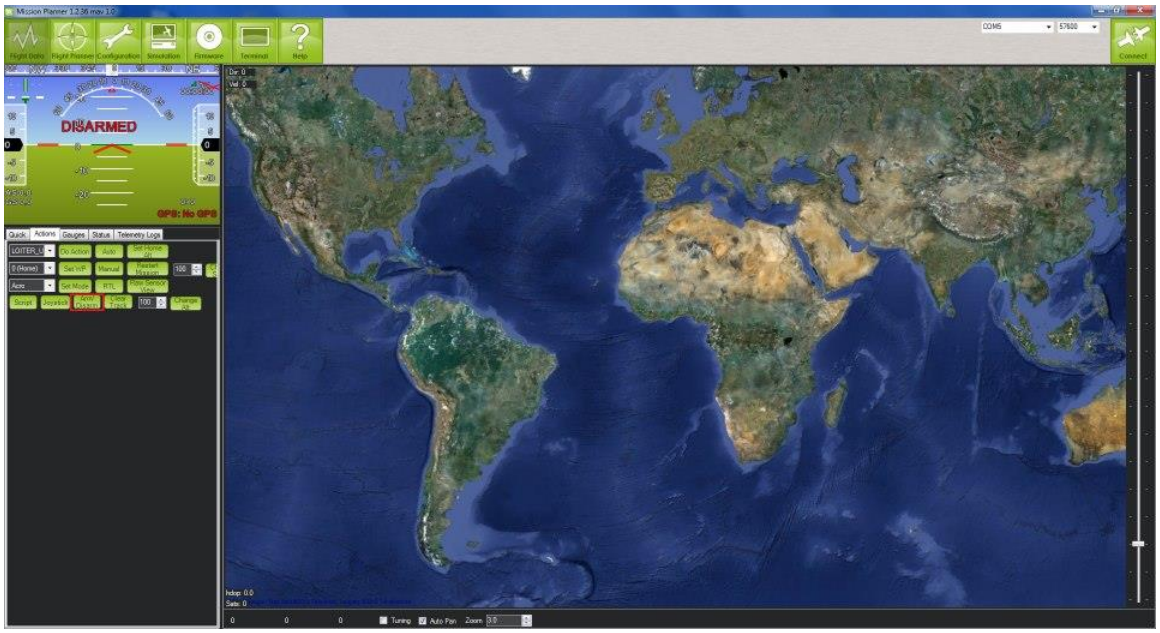

Figure 10.0.4: Screen with “Arm/Disarm” button highlighted

Once the quadrotor is hovering a safe distance above the ground, run the AI software and allow it to verify connection. Once it has finished verification, launch the mission.

References

[2.2.1] http://www.livescience.com/4950-key-optical-illusions-discovered.html

[2.2.2] http://www.thefreedictionary.com/dead+reckoning

[2.2.3] http://emergentbydesign.com/2010/04/05/essential-skills-for-21st-century-survival-part-i-pattern-recognition/

[4.2.1.1] http://clarkvision.com/imagedetail/eye-resolution.html

[4.2.2.1] http://www.policyalmanac.org/games/aStarTutorial.htm

[4.2.2.2] http://www.ai-blog.net/archives/000152.html

[4.4.2.1] http://www.starlino.com/imu_guide.html

[4.4.2.2]http://www.analog.com/en/content/td_accelerometer_specifications_definitions/fca.html

[5.0.1] http://www.propellerpages.com/?c=articles&f=2006-03-08_what_is_propeller_pitch