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., is a project that is described fairly well by the title. Specifically, this project endeavors to design a team of four quadrotors, along with all of the artificial intelligence required for autonomous behavior, that will be able to play competitively against human opponents. The quadrotors must be able to act and respond in real-time, as well as making 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 must be designed and brought together in an efficient manner. It is the intention of this report to lay down a specific design plan and time line in order to make possible the orderly 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 contributing motivations for this project are many, but the main influence is the personal fascination with creating something that can fly of its own accord. Simply put, the A.Q.U.A.L.U.N.G. project is 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, requires 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 is matched to one of four subsystems. These subsystems are:

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

Each subsystem is interconnected with the others in a manner that is straightforward, making their integration a more manageable task than trying to connect of of the dozens of systems to each other individually. The general organization is that every subsystem feeds information to the AI system, which uses this information to make real-time tactical decisions. Then the AI subsystem sends 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 assists 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 are many factors in the motivation for this project, ranging from personal to practical. The practical motivations for this project revolve around how it can be used and applied once complete. The personal motivations are responsible for choosing this specific project, as opposed to one of many others that may also have 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 differeing “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 would be in the 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 would also lay the groundwork for active use of quadrotors in a police or military environment. 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 is not quite as direct as the previous options above, but it is tied closely to the most recent one. This project could make significant 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 are 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 ties in very closely with the practical applications. Each member in the group feels that this project is the most significant undertaking in their academic career, and wishes for this undertaking to have genuine, substantial purpose, as opposed to merely being another assignment completed for grade alone. This desire stems from the fact that each

member is 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 is also traceable back to the definition of a profession, that is, “a calling requiring specialized knowledge”. Not only is it desired for the project to be useful, but also to be challenging. Each member desires to work on a project that is novel, and demands the use of the “specialized knowledge” that has been accrued during their undergraduate academic career, not simply a project that “reinvents 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 is to gain experience in working on a problem that has not been solved, to make footsteps for others to follow.

In addition to being motivated to work on a project that requires specialized knowledge that has already been learned, it is desired to work on a project that allows for growth into areas that each group member has not worked in before. Each member is 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 is the shear 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 wishes 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 working on this project.

2.2 Objectives and Goals

The objective of the A.Q.U.A.L.U.N.G. is to have four quadrotors play a game of laser tag against four human opponents. The four quadrotors will work 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 is 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 have to happen. The quads have to be agile, fire with good accuracy and they have to be able to make decisions quickly and dynamically, making them unpredictably elusive.

The quadrotors will not have to map their own environment when searching and destroying, they will be navigating in a know environment that is pre mapped and programmed into the AI. This will give the quadrotors a leg up on the competition, if and only if the opponent has never navigated the game course before, making them an even more threatening force against the humans. If the human opponents have navigated the course before then they start the game with the same knowledge as the quadrotors, a mental map of the course. It will be 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 will 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 are going to have to be able to see if this is going to be a fair fight, and a fair fight is what this project is all about. So the objective of the video camera on each quadrotor will be to of course capture images, the images the quadrotors are seeing, and send them back to the central computer. Although, like most other things on the quadrotors, the camera will have to be a balance of resolution, weight and cost. If the camera does not have the resolution to be able to send back images that the Artificial Intelligence can decipher, than there is not even a point to having a camera in the first place. So the camera has to be able

to show one object from another, without being too bulky or too expensive, as weight on a fly machine is 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. [2.2.1] 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. At this point in time the objective will be to cater to a lesser trained more relaxed combat environment than advanced military training, so the image processing needs to be somewhere in the vicinity of keeping up with normal human capabilities if it is to be on a competitive level.

For stabilization, the quadrotor will be using 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 will need to be able to self stabilize whilst navigating, shooting and evading. The objective would be 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 is possible to program the quadrotors to do advanced combinations, such as shooting a fix or moving target whilst the quadrotor itself is moving, If the artificial intelligence evaluates to big of a threat from the enemy for a particular quadrotor, it would be possible to not only move and shoot at the same time, but another advantage it could have would be to retreat out of harm's way as well. Advanced movement and stabilization will be a key in the effectiveness of the quadrotors tactical ability.

For keeping track of each of their positions, each quadrotor will use 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. [2.2.2] So by using the Inertial Measurement Unit the quadrotors will be able to calculate their positions as they move 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 to report back the position of the quadrotors with a certain tolerance in mind. For example,

if the sensors on the IMU could report back an accurate position within plus or minus X number of inches to its actual position, it would be considered acceptable and could be used with confidence. However if the error becomes too great, and the microprocessor is saying that the aircraft is in one position, when it is really a foot away this is no good and cannot be trusted to accurately guide our aircraft through the course safely. So the objective would be to find this tolerance point and not exceed it, but 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 build up causing inaccurate readings.

In order for all of these tactical movements, retreating movements and images to be processed, there needs to be an intelligent system to make quick and important decisions. This needs 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 programmed into the central computer will take care of all of these decisions. The most important aspect of the artificial intelligence for the quadrotors will be their ability to act unpredictably even though they will roam 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. [2.2.3] So in order to maximize the ability of the quadrotors to play competitively against humans, dynamic decision making is 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 will need an energy source to power them; this energy will be 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 makes them perfect for the application of powering everything needed on the quadrotors during a game of laser tag. The objective will be to figure out how to maximize the energy output of the batteries while keeping their weight down. For example: it is 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, will 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 is 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 will have to be in place. Each

quadrotor will have to have a transceiver on it that can send data back to the computer. The computer will also have to have a transceiver on it, but it has to be able to receive data from four transmitters at once, since of course there will be four quadrotors talking to it at all times throughout the game. The wireless communication system will 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 will be sending back to the computer their position in the field, which was determined by dead-reckoning, they will also be sending back the data from the video camera that is on board the quadrotors. All this information needs to be able to be sent back while the computer is sending information to the quadrotors. It computer will 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 is one of the most important aspects of the whole operation.

The brains of the operation as a whole will be the central computer. This unit will be in control of piecing all the components together, and making them all interact with each other as one. The objective of the central computer is to have the computing power to make this happen, but since the quadrotors will be in combat as the computer gets information, it not only has to be able to compute the information and send out commands, it needs to do so in real time with zero lag. Without a fast processor the quadrotors could be getting commands to shoot way to 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 can be split into two categories hardware and software. For the hardware the requirements can range from physical characteristics of the parts, which cannot be changed, to performance of the parts, which can depend on a number of factors. The software side of the requirements deals 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

needs to be strong enough to be able to hold all of the components that will be placed onto each quadrotor, yet light enough to be able to be lifted off the ground. The more weight the quadrotor has on it, the bigger the frame needs to be, since more powerful motors will 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 will have a diameter of somewhere in the range of 10 – 18 inches. This is due to the fact that it has to successfully carry more than the normal load for a quadrotor, and in consequence the propellers will need to be a little larger. The propellers can range anywhere between 5 – 15 inches in diameter and the pitch can vary.

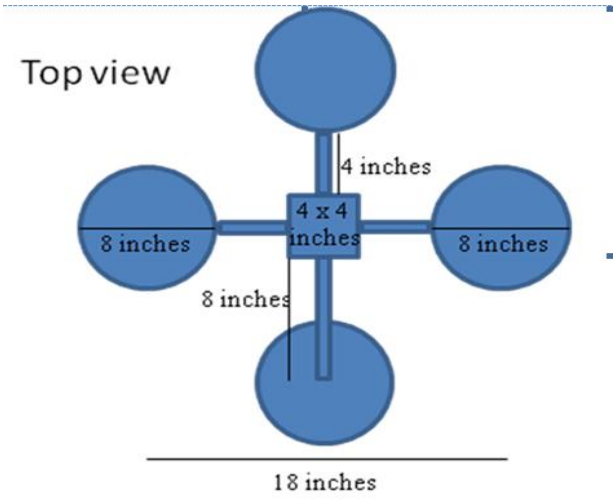


Figure 2.3.1 Possible Quadrotor Dimensions

Figure 2.3.1 shows one possible dimension set up for the quadrotors, it shows some of the dimensions that need to be taken into consideration. It all starts with the size of the interface plate; here it is a 4 x 4 square in the middle of the quadrotor. This dimension will depend 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 4 x 4, then a bigger interface plate will need to be considered. Once the sizes of the interface plates are established it is possible to choose the length of the arms for the frame. The arm length is important in balancing the aircraft as well as keeping the propellers far enough away from the interface plates. The arms have to be at least as long as the propellers radius to insure that the propellers are not hitting anything on the quadrotor, but it is even better to have the arm length be twice the radius of the propellers. This provides better balance and steering capabilities when the quadrotors are in dynamic flight. Now the propellers can be chosen, there are two dimensions that need to be taken into consideration, the length of the propellers and the pitch of the propellers. This will all depend 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 are important are the amps that it will draw, and the Kv rating. The amps matter because the batteries need to be able

to support whatever it draws, and the Kv rating tells how many RPMs/volt, finding the perfect balance for these two ratings will be the key to optimizing the quadrotors performance.

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

Laser gun and sensors also need 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 for use 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.

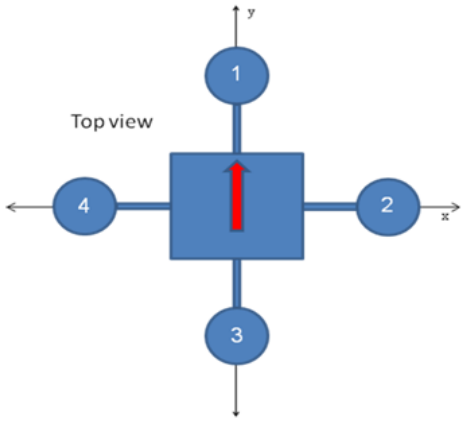
Another project that has been researched is a previous senior design project at UCF. The Reconnaissance and Demolition Super Attack Tank project described in the Spring 2012 Senior Deisgn 1 class has been researched for the merits of computer vision solutions described therein. The motivation for researching 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



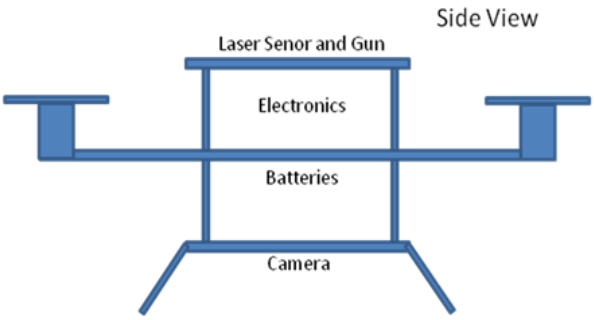


Figure 4.1.1 *Figure 4.1.2*

The architecture of the quadrotors will look like figures 4.1.1 and 4.1.2. Figure 4.1.1 is a simplistic view of the type of flight configuration that will be implemented in A.Q.U.A.L.U.N.G., in non-technical words it is the '+' configuration instead of the 'x' configuration. This type of configuration was chosen for simplicities sake, it only requires two motors to change their speeds in order to move forward, backward, left or right. For example, if the quadrotor wanted to move forward motor 1 would slow down while motor 3 would speed up causing a tilt forward, while motors 2 and 4 have no change at all. Once movement takes the quadrotors off the axes all four motors will of course be needed, however the '+' oriented flight configuration is the simpler, yet still effective, choice.

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 will, of course, supply power to the entire system, varying with whatever each component needs. The quadrotors will have the flight control (navigation) system and image processor on them, and will wirelessly send two pieces of information to the AI control system. It will send its current position deduced by dead-reckoning and it will send a video stream. The AI control now has this information and will decide what the quadrotor should do from there, if the video sees a threat it will 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!

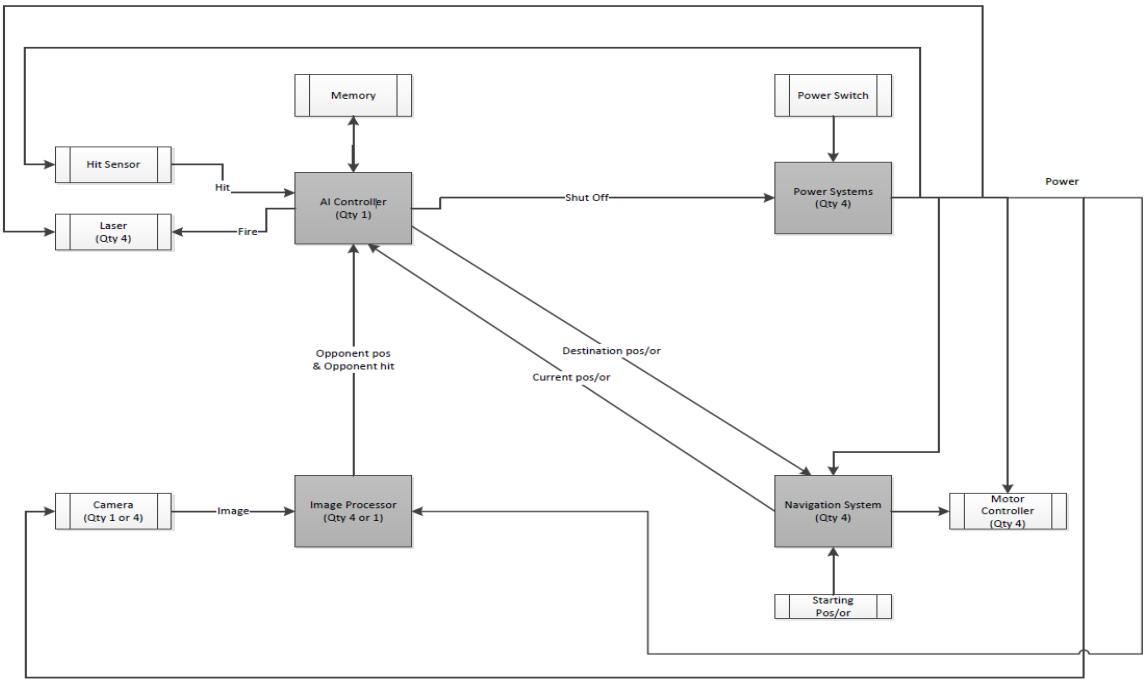


Figure 4.1.3 Main Subsystems Flow Chart

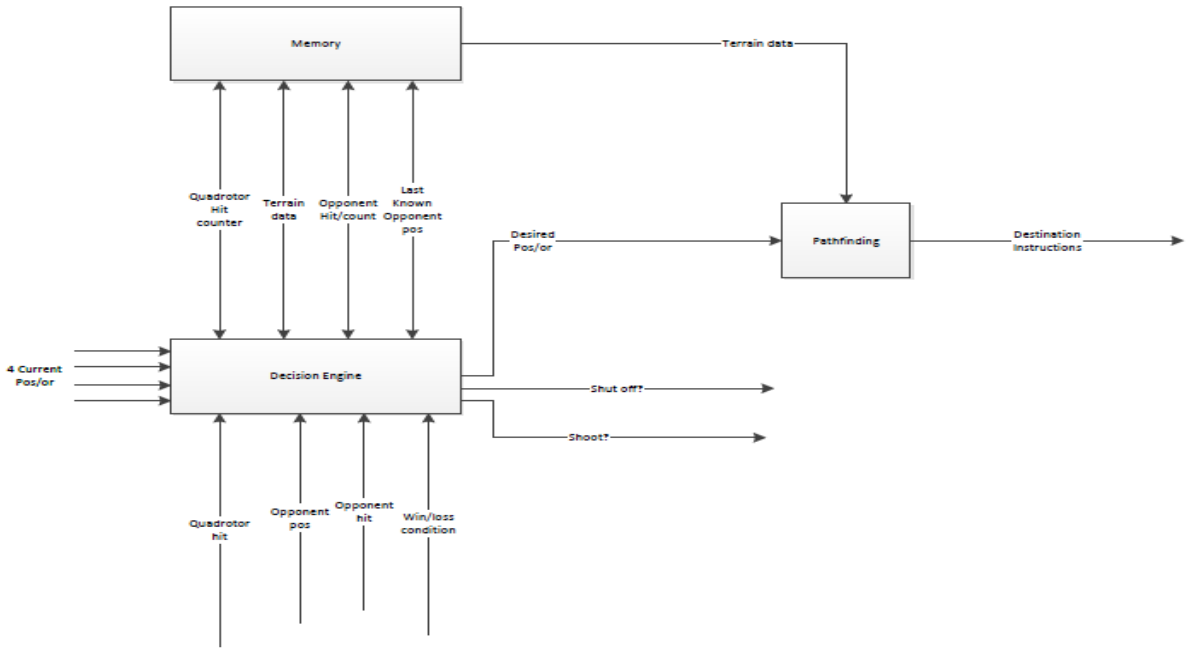


Figure 4.1.4 The AI Controller subsystem, covered extensively in section 4.2.

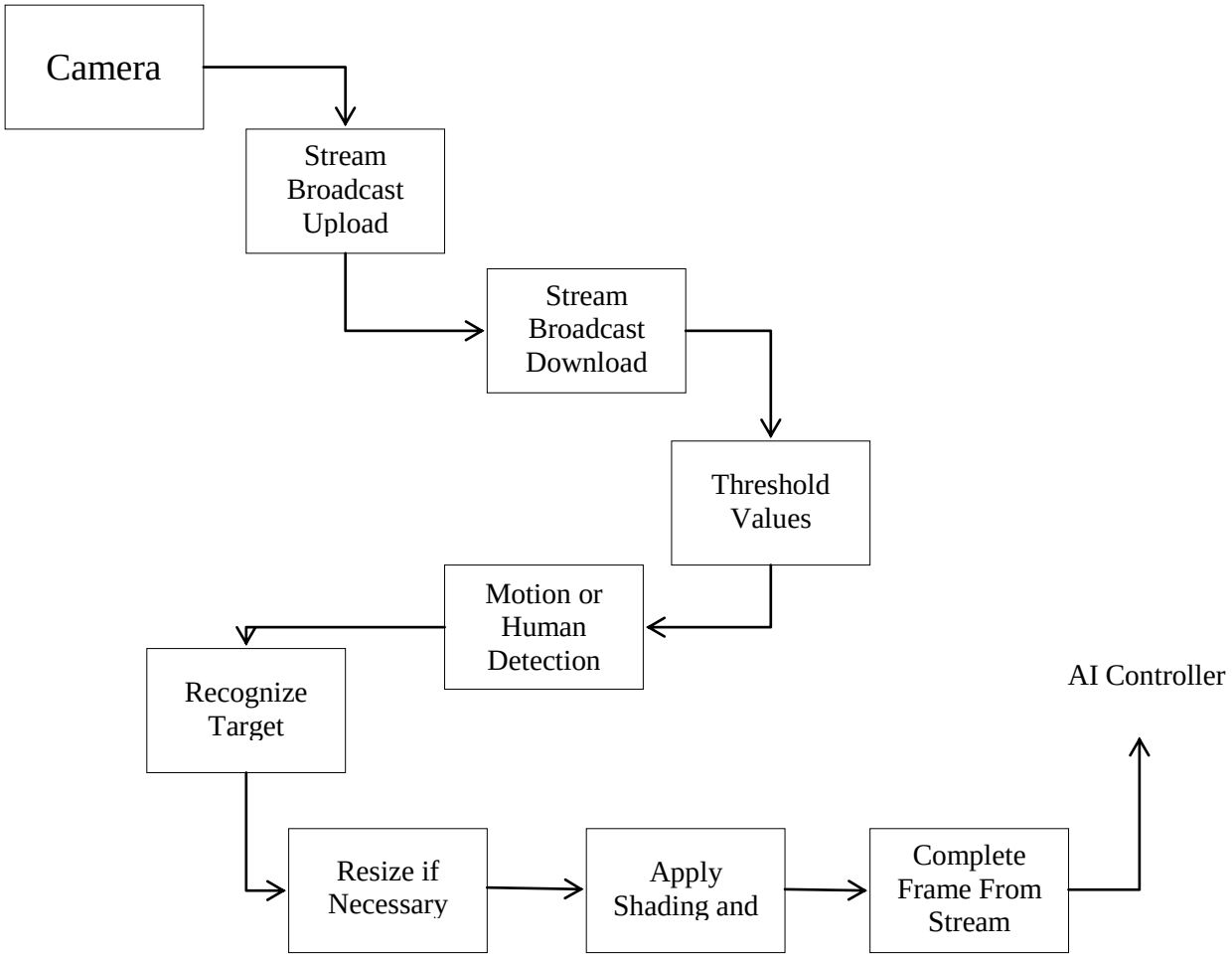


Figure 4.1.5 Image Processing Subsystem Flowchart, covered extensively in 4.5

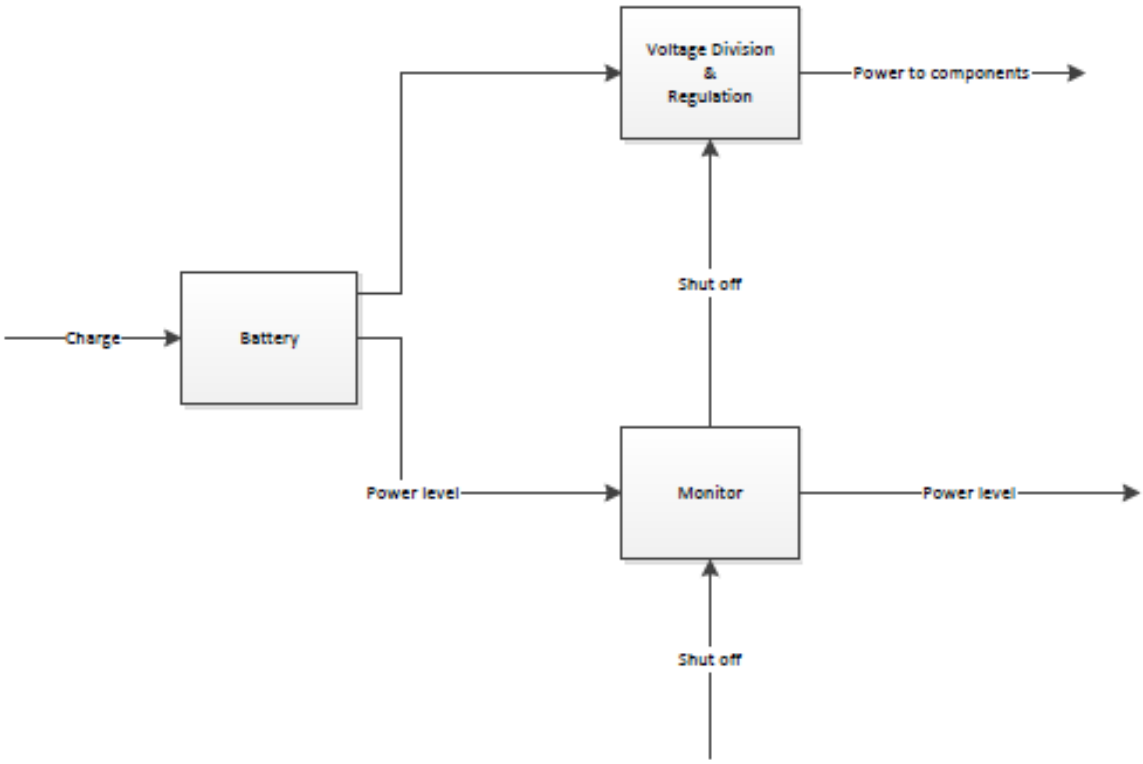


Figure 4.1.6 Power Systems Flow Chart, covered extensively in section 4.3

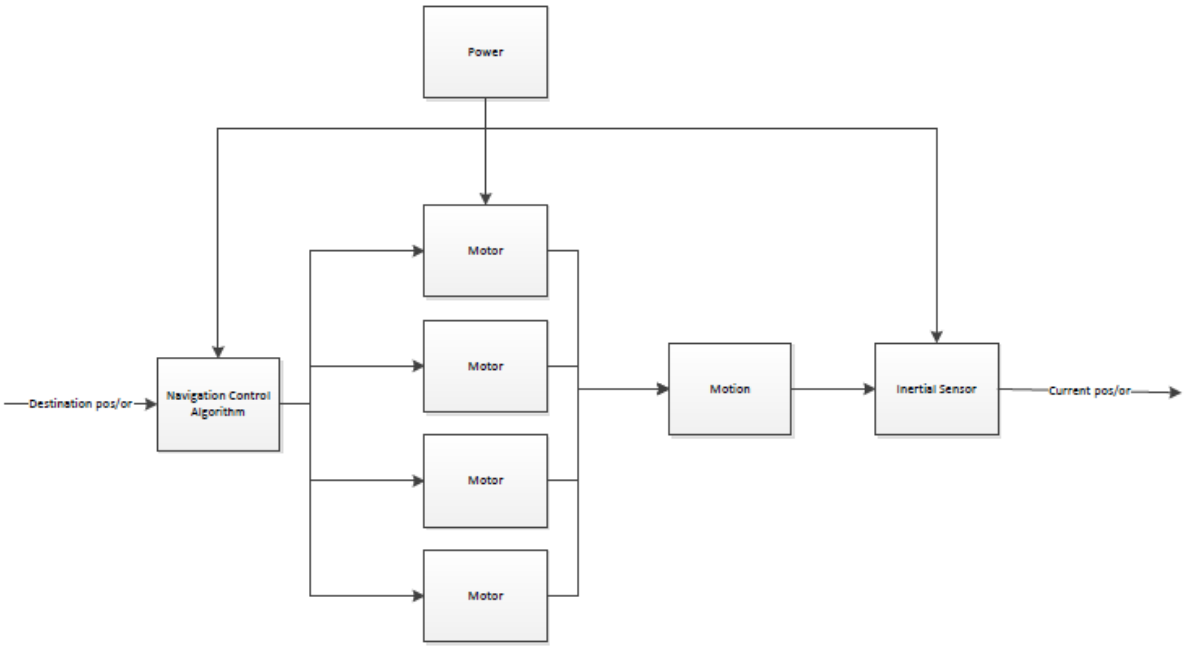


Figure 4.1.6 Flight Control Subsystem covered extensively in section 4.4

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

For the image processer, an image will be sent from the camera to the main system for detection, once motion is detected within the video stream it decides if it is human or not, if it is it marks this as a target, does some conversions, and sends out the stream to the proper authority on executing an attack.

The power system is a quite simple system that will just disperse 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 will be made, either regulating, amplifying or not changing the voltage at all. Once the correct voltage is established it can be sent out to all the parts.

The flight control subsystem will take readings from the Inertial Measurement Unit and convert them into viable readings in order to stabilize the quadrotors as well as determine the quadrotors position using dead-reckoning. Once the position is established it sends the information to the AI and get 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

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 for use 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.

4.2.2 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 will have substantially more devastating results than the failure of any other Artificial Intelligence component. If the quadrotors are unable to evade an opponent, they are simply eliminated from the competition once they are defeated. Similarly, if they are unable to acquire a target and defeat opponents, they will merely be unable to win the competition. However, if the quadrotors cannot find a safe path through the environment, they will almost certainly crash into an obstacle causing damage to themselves, and, if that obstacle happens to be a human opponent, they could cause injury to that person.

While there are a few alternatives, the path finding algorithm that is expected to perform best 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 will all be applied to an example environment, shown in Figure 4.2.2.1.

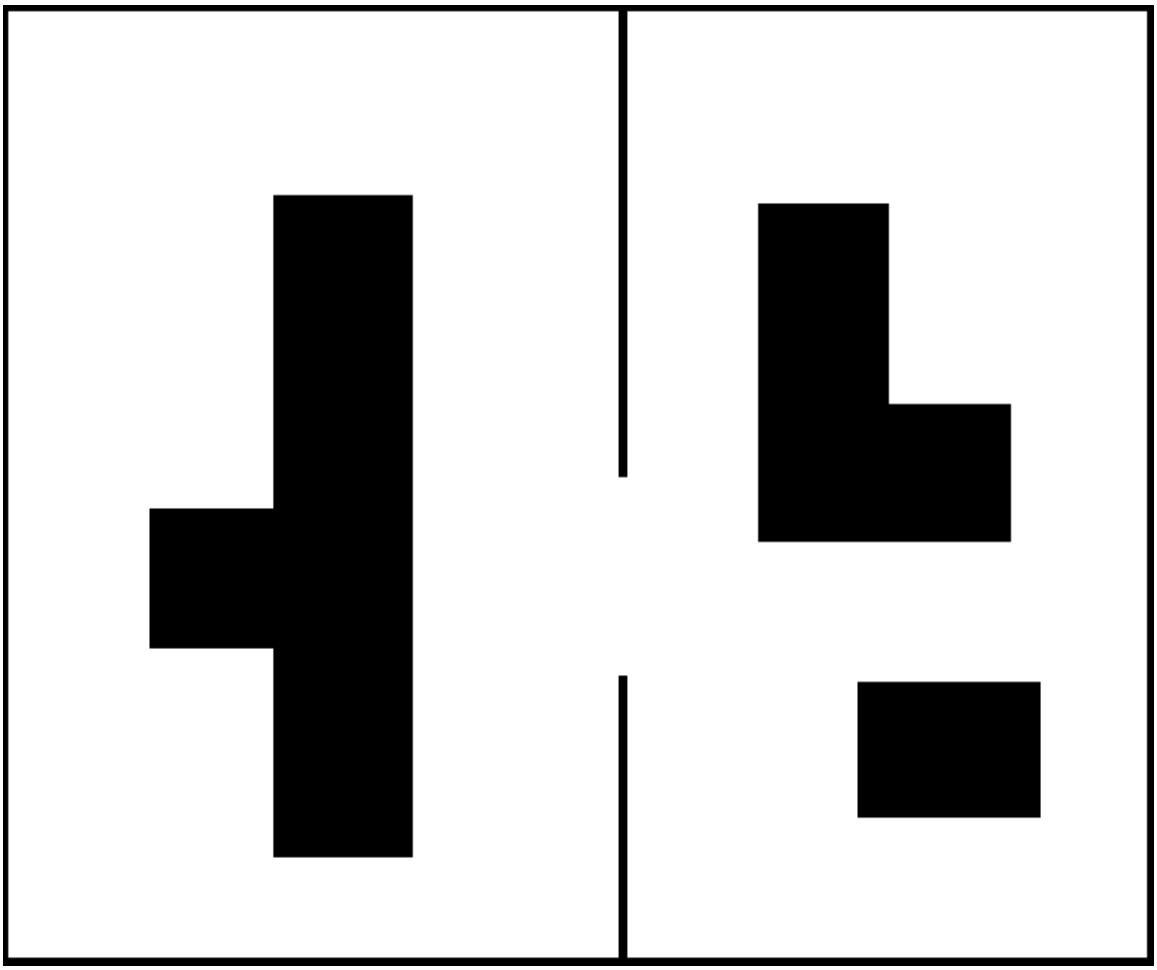


Figure 4.2.2.1: Simple Example Environment

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 thereof, of a third dimension.

Regardless of the specific representation employed, an interruption feature will be included, primarily to avoid collision with a human opponent if they move into the path of the quadrotor. The quadrotors are able to to be given an updated path before they finish following a path they are currently on; the system will remain dynamic at all times. This will allow 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 lays a grid over the entire area and labels each resulting cube as either clear or an obstacle. The center of each cube will contain a node. A path is then 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.

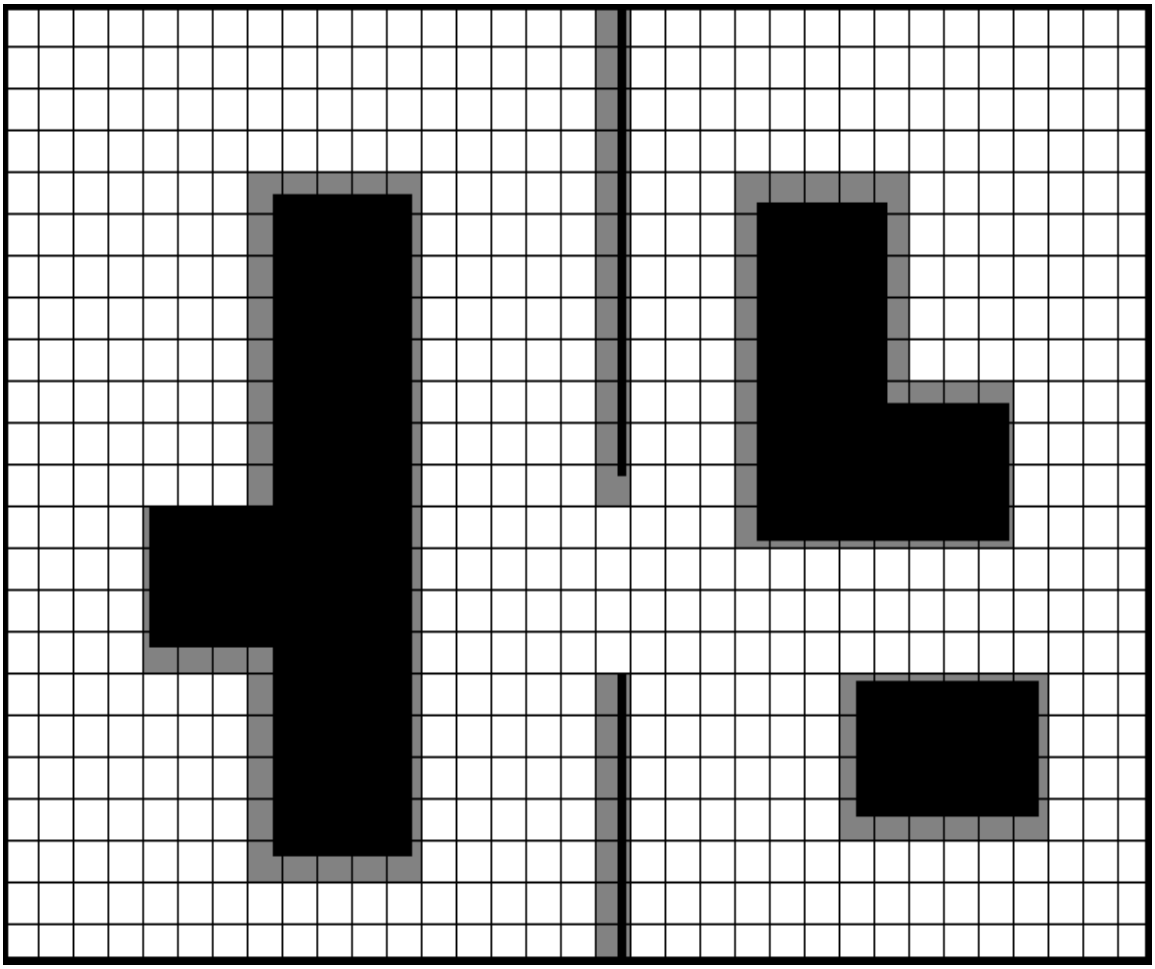


Figure 4.2.2.2: Grid Representation of Example Environment

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 should not be simply the Euclidean distance from the evaluate node to the target node, as movement within the grid is limited to adjacent nodes. This means that all motion happens along one of the three cardinal directions or a diagonal that results from an even combination of one or two of these directions. As such, the heuristic used is the sum of three values. The first value is the length of the least traveled cardinal direction multiplied by the square root of three. The second value is the difference between the lengths of the median and least traveled

directions, multiplied by the square root of two. The final value is 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 A to the target location if no obstacle is present. An advantage to using a grid is that every location in the environment is accounted for and can be navigated to, if the location to be navigated to isn't an obstacle itself. The most significant disadvantage to using a grid representation is 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 could be examined, and many of them will be examined more than once. Before any of the quadrotors are able to move, they must have a path to follow, which means that they will sit idle until a solution is 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 is to create two grids. (Lester, 2003) The second grid is composed of much larger cubes, thus having a greatly reduced search space. First, a path is found from the starting node to the ending node on the large grid, and then a path on the small grid is found, but only from the starting point to the large node two steps further on the first path. This micro-level path is then updated every time a new macro-level cube is 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 is 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.

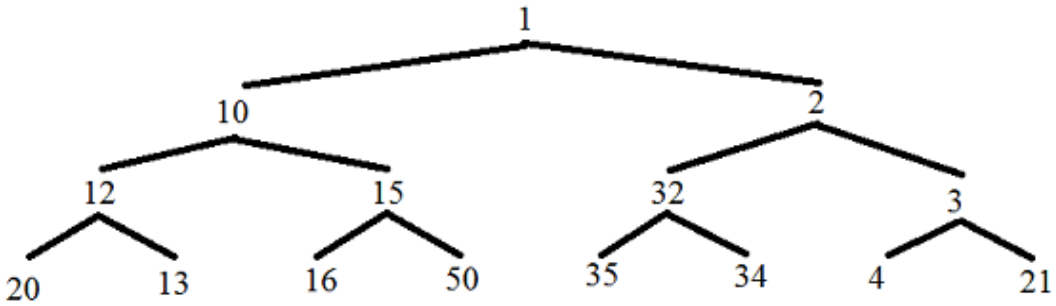


Figure 4.2.2.3: Binary Heap Example of F scores for A Algorithm*

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 also comes with methods for re-organizing. Once a node has been examined and removed from the array of nodes to be searched, it is discovered to be a better alternative, and thus is moved to a more prominent position in the heap, or when a new node is added to the list to be searched. A binary heap can be used in any representation of the environment used for the A* algorithm, but it has 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 is moved within the heap, it is compared with each successive parent or child, depending on the nature of the reorganization, until it is more desirable than either child, but less desirable than its parent. Because of this, the maximum number of steps to find the correct place for a node is 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.

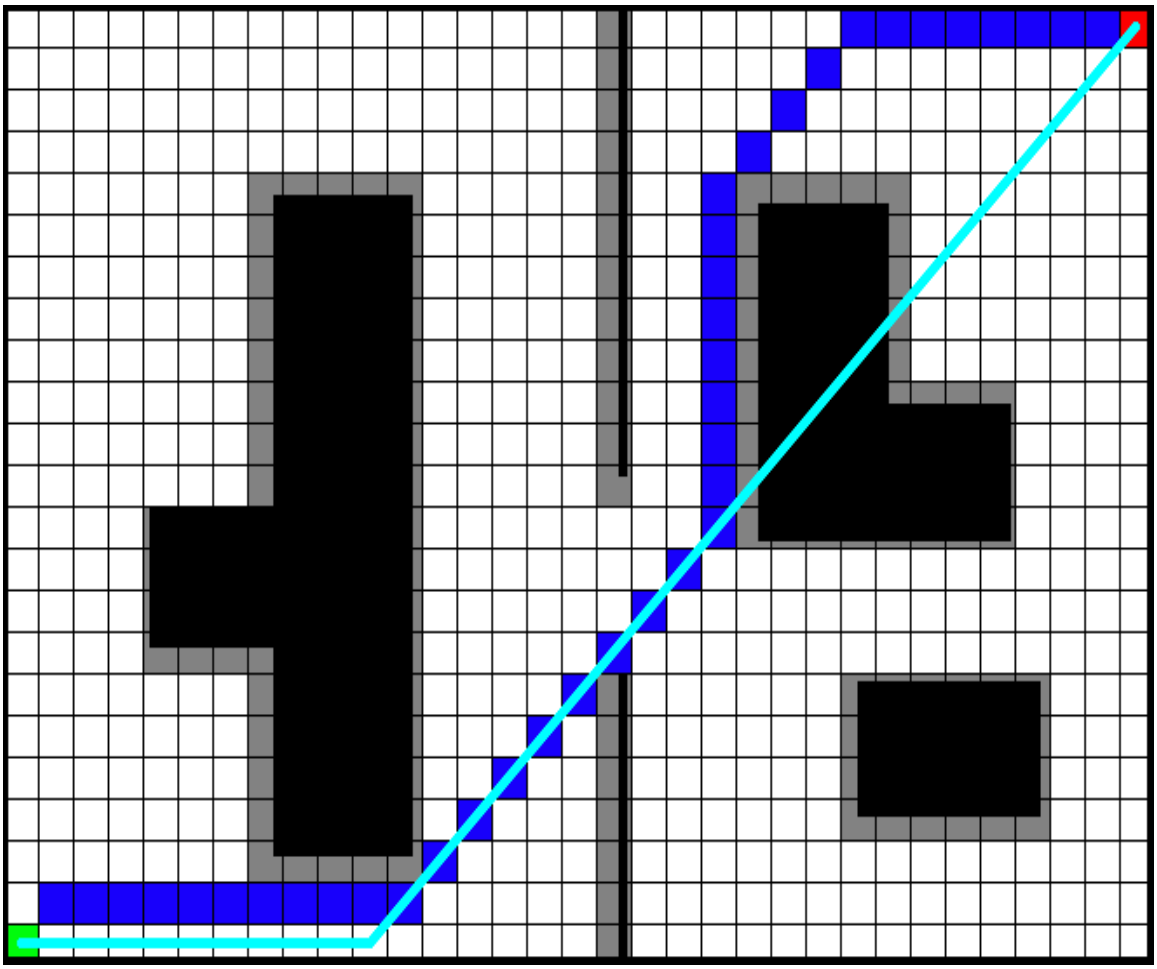


Figure 4.2.2.4: Example Path Using a Grid Representation

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

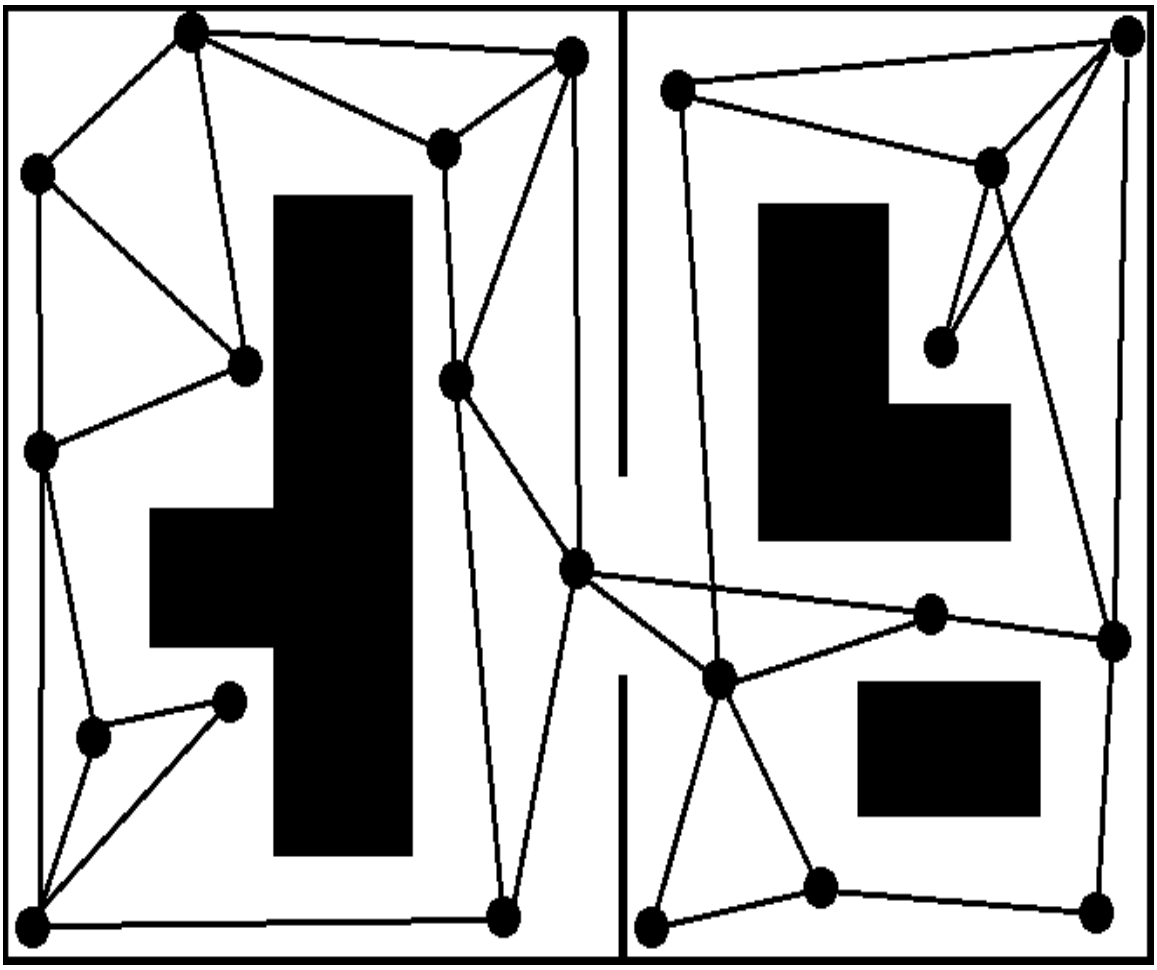


Figure 4.2.2.5: Way Point Graph Representation of Example Environment

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 is merely the Euclidean distance from the node being evaluated to the target node. This heuristic, as before, represents the minimum possible distance from the evaluated node to the target node if a straight line can be followed. Way point graphs allow, and require, the developer to make choices between speed and flexibility. On one end of the spectrum, a way point graph becomes 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 means and 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 only cost a small amount of extra time per iteration, it can add up to be a substantial increase with over 300,000 nodes, as in the example above. This burden is 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 can be used to basically draw a circuit through the environment. In this case the only decision to be made is which direction to go around in, and no matter how long the circuit is, the decision can be made almost instantaneously. Regardless of this speed, this level of complexity has little practical use in a dynamic combat system, as it does not allow for flexible, fluid movement through the environment. The last flaw that can be found with a way point graph is 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 cause it to crash. While the issue of converging paths is unlikely given the movement strategies of the A.Q.U.A.L.U.N.G. project, it is not altogether impossible in a highly dynamic environment, such as laser tag combat. The easiest solution to this conundrum is 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 is the ability to meet in the middle of the complexity spectrum, to place and connect nodes in such a fashion that the quadrotors will be able to move dynamically through the environment, but reduce the search space from that of a complete grid layout. Additionally, the complexity can be 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.

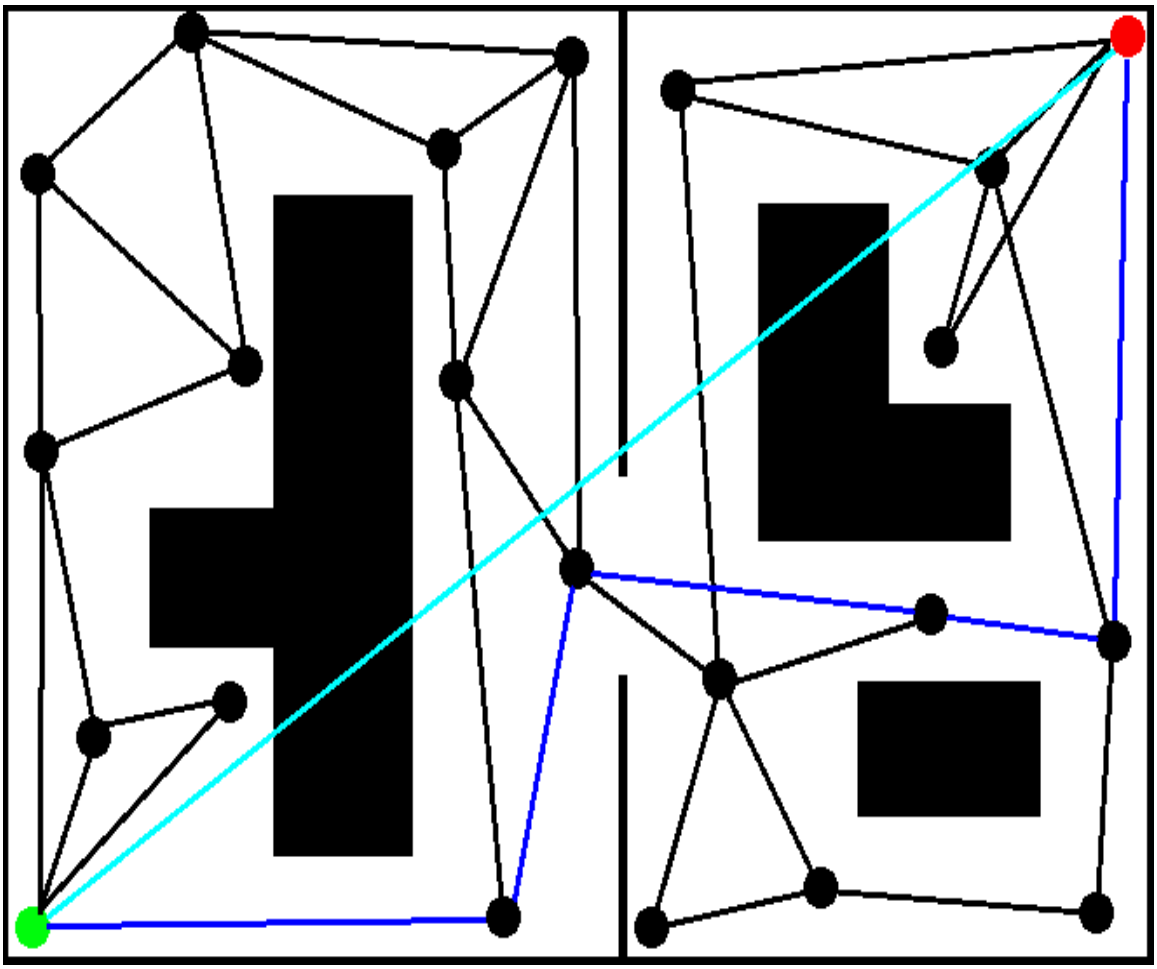


Figure 4.2.2.6: Example Path Using a Way Point Graph Representation

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 allows 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 will store information about its vertices and the neighboring polygons that can be traveled to. The center of each polygon is a node that the A* algorithm will use as a starting point for creating the path to follow, but since information is available describing all moveable areas, the path can 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.

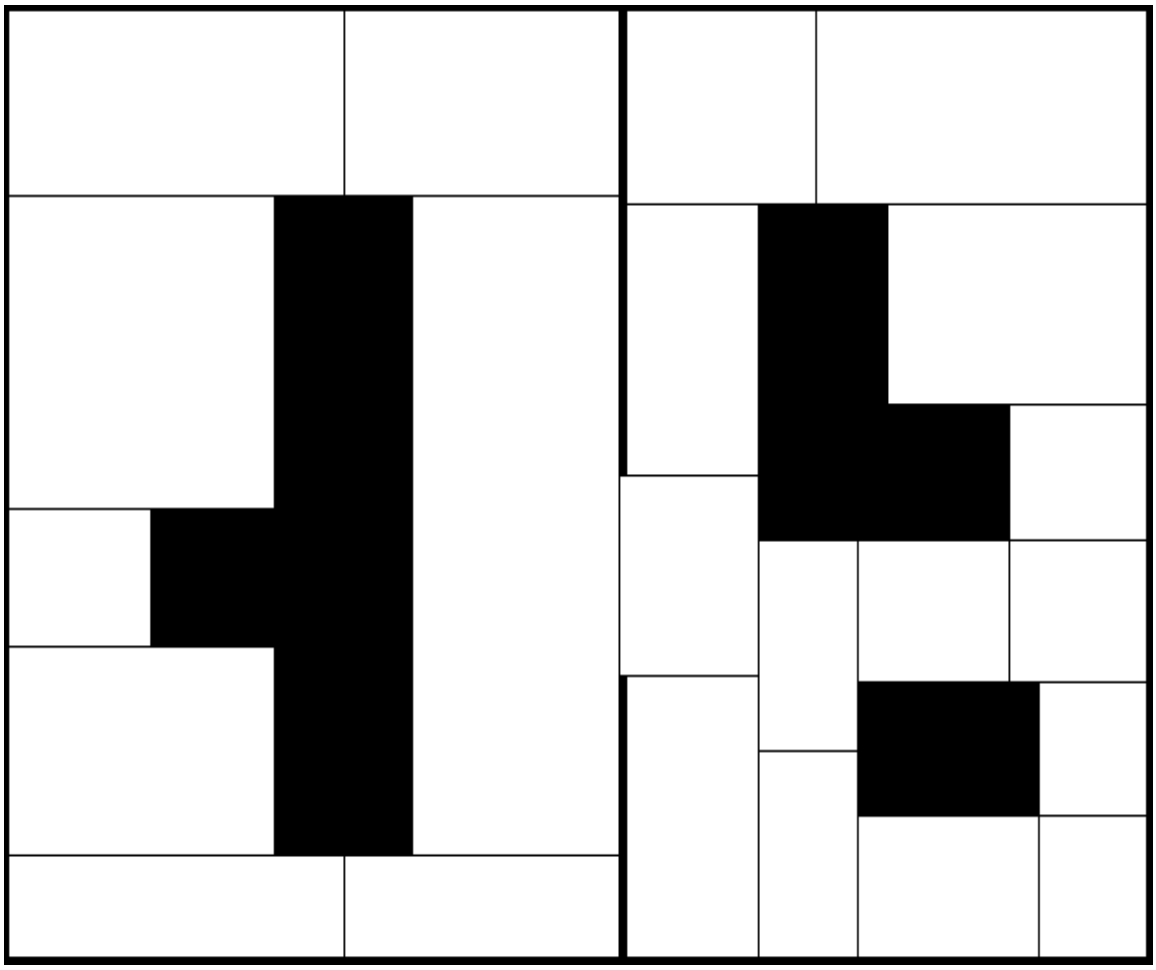


Figure 4.2.2.7: Navigation Mesh Representation of Example Environment

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 is 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 means that the search space is reasonably sized, allowing for rapid determination of the path to use. The positive aspects of using a navigation mesh to represent the environment make this method very attractive. The downside lies within the complexity added with creating more efficient, direct paths. The processing time is not really a concern, as altering the path only involves a small number of nodes once the A* algorithm has finished running. Rather, it requires the developer to design a final step that wraps the path around any obstacles and draw straight lines as often as possible. The method for this step that will be 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 returns to this line. The result is the best possible path that can be 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 is that the flight control algorithms on board each quadrotor will handle 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.

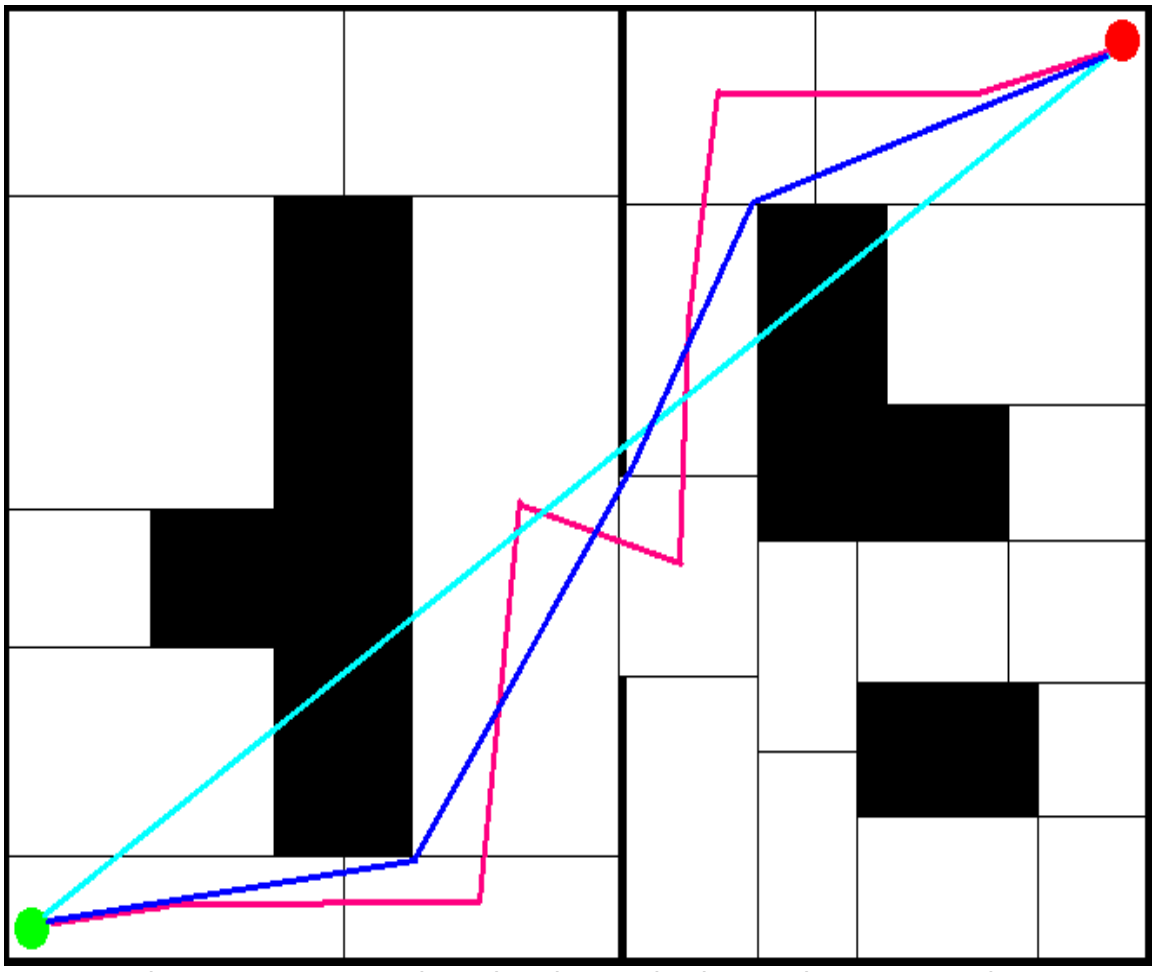


Figure 4.2.2.8: Example Path Using Navigation Mesh Representation

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 are examined, the most desirable representation of the environment is the navigation mesh. To further illustrate this point, Figure 4.2.2.9 compares the final paths returned for each representation.

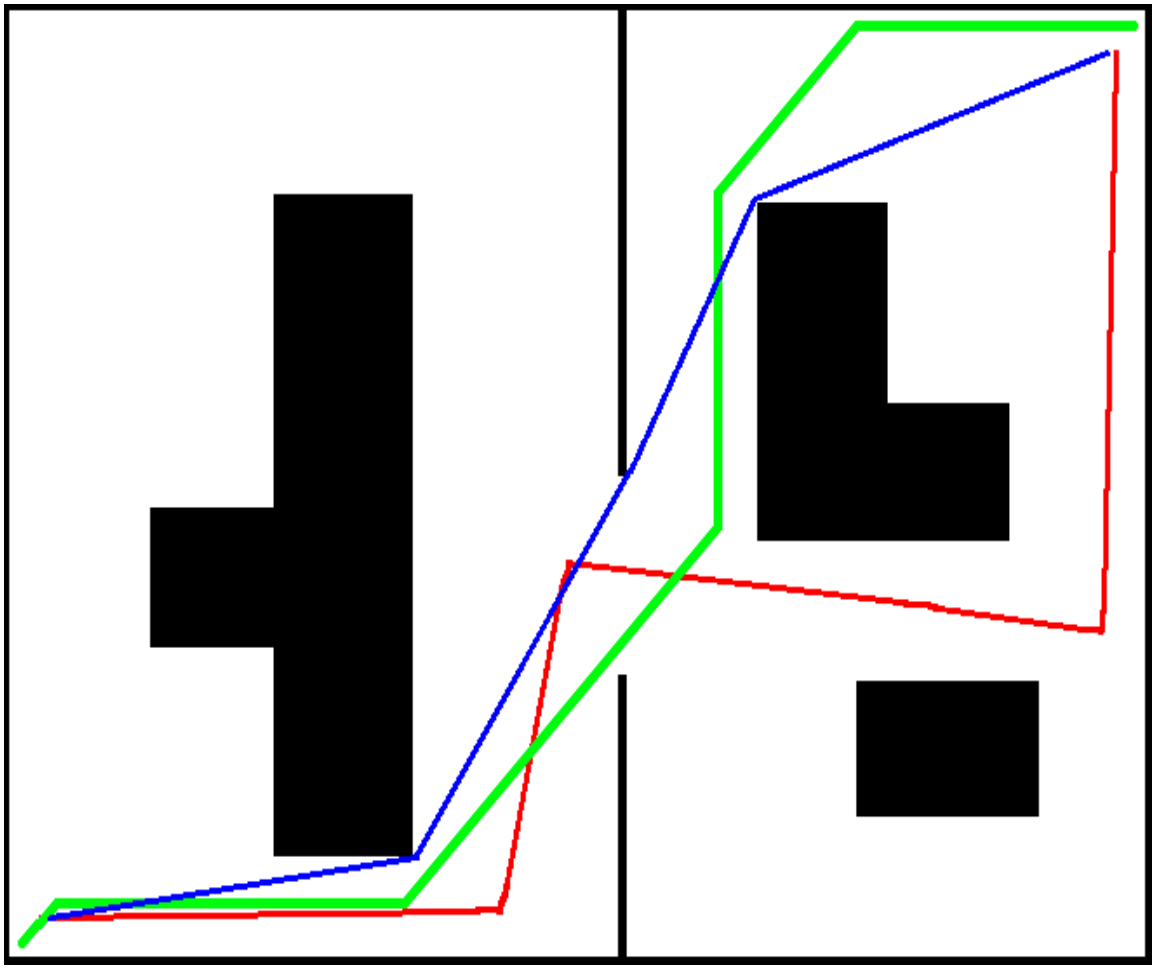


Figure 4.2.2.9: Comparison of Example Paths for Presented Representations

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 is the most direct of the three options. Not only does this representation allow for the most direct path, it also works 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 with a high degree of optimism.

4.2.3 Decision Making

The decision-making aspect of the AI subsystem is the high level control that binds together and directs the actions of all of the other functions and algorithms contained within the subsystem. This part of the AI looks at all of the data presented from the other subsystems, as well as data returned from within the AI subsystem, and then sorts through this information for the most relevant bits and uses 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 will be 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 allows for a multitude of behaviors to be designed and run with explicit conditions when one behavior switched into another.

The primary change that will be 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 4.2.3.1 shows a representation of the A.Q.U.A.L.U.N.G. decision-making state machine.

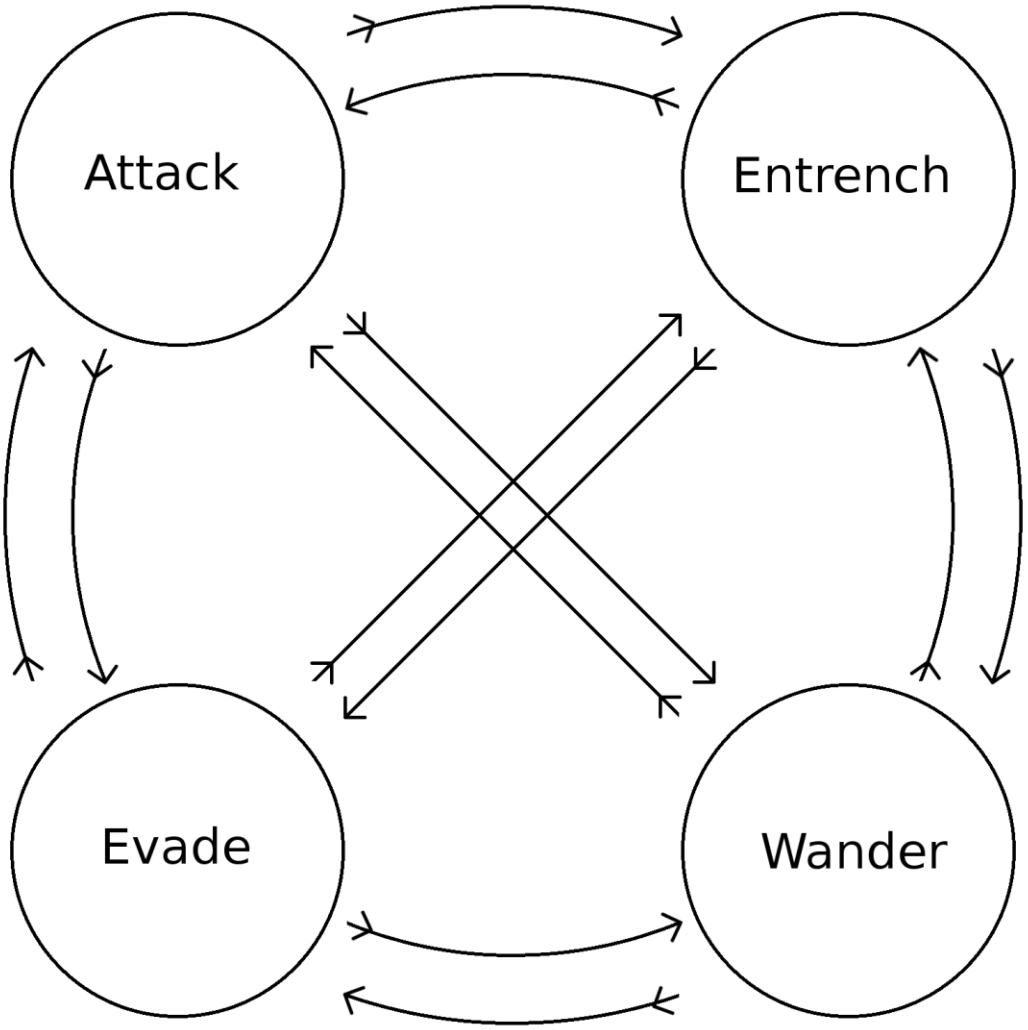


Figure 4.2.3.1

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

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

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

The final state is the “Evade” state. This state is switched into when the humans outnumber the quadrotors. Under these conditions chasing after the enemy is foolish, and maintaining control is less desirable, as the advantage of numbers is often more influential to the outcome of a battle than the advantage of position. When in this state, the tactical planning function is 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 is 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 will 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 will then make a determination of which set to use. Once a set of target locations is chosen, these will be provided to the A* path finding algorithm. The A* algorithm will return a sequence of locations, representing the vertices of the path discovered. These points are the communicated to each associated quadrotor. The quadrotor is also constantly broadcasting its current position back to the AI subsystem, so that progress may be tracked. Additionally the state machine is always listening for a collision warning interruption. If this interruption occurs the state machine will tell the quadrotor in question to immediately stop moving. The state machine with then provide information on the nature of the interruption to the A* algorithm, so a new path may be determined. The final input the state machine must respond to is data from the computer the computer vision subsystem. If an enemy is spotted this information is not only given to the probability field algorithm, but also directly to the state machine. When an opponent is spotted the quadrotor is told to pause its current path. The quadrotor is then given the position of the enemy, and told to change its orientation to face the enemy directly. The quadrotor is instructed to fire once it is aiming at the opponent's center of mass. After the quadrotor has fired, it continues along its path. A diagram of the interactions of each of these functions, algorithms, and objects in the AI subsystem is given in Figure 4.2.3.2.

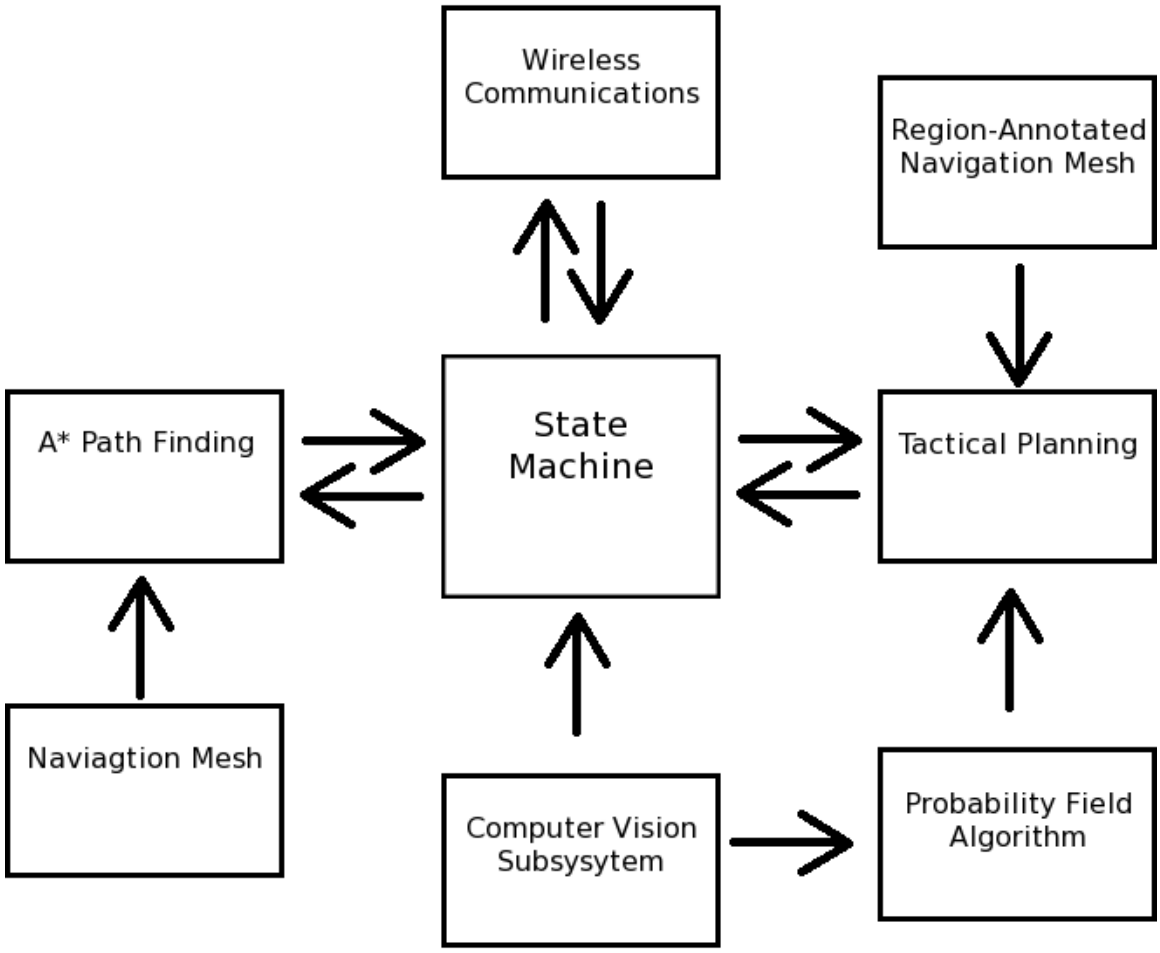


Figure 4.2.3.2: Organizational Layout and Data Flow of AI Subsystem

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.

4.2.4 Hardware for Centralized Control

When building a computer from the ground up, selecting each part, there is always some difficulty deciding on a precise part until the time of purchase. This fact arises due to the dynamic nature of the average cost for a given performance and the transient nature of sales on equivalent components. However, minimum performance requirements are more plausible to define, then specific parts can be selected such that all requirements are met. As such until a purchase is made, specific part numbers will not be chosen, so as to minimize costs.

The only exception to this fact is the processor, as there is not as much competition. For the A.Q.U.A.L.U.N.G. project the Intel core i7-3770K Ivy Bridge processor will be 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 allows for overclocking, should the default speed prove too little. 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 is 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 allows for 4 threads to be dedicated to video processing, one per stream. One thread can be dedicated to communication with each quadrotor unit. One thread will be reserved for path finding. One thread is set aside for high-level decision making. The final thread will handle all other miscellaneous tasks. Being able to thread the system in such a fashion means that each task can run in parallel, independent of the performance of another thread. This results in increased dependability and speed for the system as a whole.

The next component to be discussed is the motherboard. Obviously, the motherboard must support the processor, so it must have an LGA1155 socket. In order to facilitate the possibility for overclocking either a Z77 or Z75 north bridge is also required of the motherboard. In all probability a Z77 north bridge will be selected at the time of purchase due to the fact that Z75 north bridge motherboards are uncommon and tend to be less feature-rich than their Z77 counterparts for the same cost. While, quite rare to find otherwise, four memory slots that support DDR3 up to 1600 are required. Another required aspect that is found on almost all available motherboards is having at least one SATA 6Gb/s

ports, and at least two USB 3.0 ports. The final requirement is 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. A feature that is desirable, but not required, is a WiFi IEEE 802.11a/b/g/n adapter.

Another component is the Random Access Memory RAM. In order to allow for some overhead in the video processing aspect, as well as allow for a complete virtual representation of the environment, a minimum of 16GB will be used. Ideally two chips of 8GB each will be used, as it is slightly faster, but four chips of 4GB each is satisfactory . DDR3 1600 or better speed is required to ensure that the system is able to operate in real time. The final requirement for the RAM is timings of 9-9-9-27 or better.

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 can have adverse effect on the performance of the A.Q.U.A.L.U.N.G. project, the faster the hard drive, the better it is. An example of a problem that bottle-necking can cause is a delay, even if ever so small, in the ability for the path finding system to determine a safe path. This delay in a critical situation could lead to the quadrotors elimination from combat. As such a Solid State Drive is required. The hard drive will be 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 is unnecessary. This means that a hard drive that is at least 60GB large is plenty. 60GB allows for plenty of room for an operating system, any programs used in development, and room for all of the final code and resources.

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

Now to get every component the electric power it needs to run: the power supply. When selecting a power supply it is important not to get the bare minimum required by the sum of each component. The reasons for this are 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 an 750W or better power

supply is required. In addition to the power rating, many power supplies come with an efficiency certification. This certification, called the 80 PLUS certification, means 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 prolong the life of system components. The A.Q.U.A.L.U.N.G. project will insist on using a power supply with an 80 PLUS Bronze certification or better.

The final necessary component for the A.Q.U.A.L.U.N.G. project is a case to house the other components. The most obvious requirement for the case is that it is large enough to fit all of the other components inside, which is 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 is that it supports good airflow in the system. This means 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.

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 quadrotors 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.

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 high temperatures however and must be well insulated.

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.

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 make it very difficult to get aluminum batteries for this project.

The Bunsen cell gets its name from its creator Robert Bunsen, and is a zinc-carbon primary cell also in the wet cell family. It is about 1.9 V and is made of a zinc anode in a dilute sulfuric acid that is separated from a carbon cathode, in chromic or nitric acid, by a porous pot. Its physical layout, which is the very nature of the cell, makes the Bunsen cell not a viable option for this project. As seen in the picture below, a Bunsen cell is not exactly something that can just be mounted to the quadrotors.

Figure: Bunsen Cell

The chromic acid cell is another type of wet cell that uses chromic acid as a depolarizer. Its cell voltage is about 2 V but these types of cells are no longer used. As they are no longer used they are not the battery of choice for this project.

The Clark cell takes its name from its inventor, Josiah Latimer Clark, and produces a highly stable voltage. They do however have a large temperature coefficient and have corrosion problems caused by the materials used to construct it so was later replaced by the Weston cell. The Clark cell is another wet cell battery which would not be useful to the goals of this project.

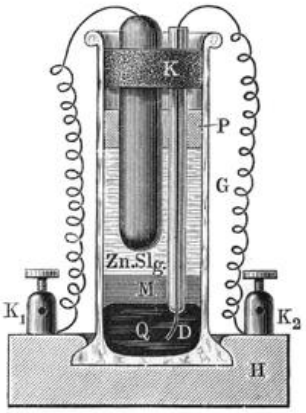


Figure: Clark Cell

The Daniell cell was created by British researcher John Frederich Daniell, and incorporated two electrolytes. It used a less dense liquid than previously made cells. It also was the first battery to use mercury which was used to reduce the amount of corrosion the batteries sustained when they weren't being used. The Daniell cell is only appropriate for stationary purposes or else the two electrolyte liquids would mix. They are now collectrors items and therefore are also not viable options for this project.

The Grove cell was another early wet cell named after Willaim Robert Grove. The Grove cell voltage is about 1.9 V and was popular in its early days because of its high current output and higher voltage than the Daniell Cell. It was later found to discharge poisonous gas and people discontinued its use. It would not be a good power source for this project.

The Leclanche cell was invented and patented by Georges Leclanche in 1866. It was a wet cell that had the ingredients that were later adapted to make dry cells. Leclanche used an electrolyte made of ammonium chloride, a positive terminal of carbon, manganese dioxide for a depolarizer and a zinc negative terminal. This combination of electrochemicals that Leclanche had mixed together was later converted into a paste to make a dry cell. Though his mix was later made into a dry cell, the wet cell version is not an option for the project at hand.

Figure: Leclanche cell

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 quadrotors.

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 sound amazing they can'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 they don't have the time or resources to allocate to getting a larger custom designed battery when there are other battery options available.

Figure: Silver oxide batteries

The Weston cell was introduced by Edward Weston in 1893 and was able to produce a highly stable voltage. It became the International Standard for EMF during the time of 1911 – 1990. The Weston cell is set up in an H-shaped glass

as shown below and contains pure mercury in one of the vessels. This battery type is of the wet cell design and is not suited for this type of project.

Figure: Weston Cell

What is known as the Zamboni pile can also be called the Duluc Dry Pile, it was created by Giuseppe Zamboni in 1812. This type of battery is made of discs of different electrochemical materials that are assembled in stacks of several thousand which are then compressed and then dipped in molten sulfur. Zamboni piles can output in the kilovolt range but only have current output in the nanoampere range. Because of its physical build a Zamboni pile would not be a sufficient option for this project.

Figure: Zamboni pile

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.

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 won’t either.

As the quadrotors 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.

Lead-acid batteries are the oldest type of rechargeable battery invented by Gaston Plante. They have a very large power-to-weight ratio which makes them very valuable for motor vehicles and are relatively low cost. Their size and weight however make them entirely wrong for this application.

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 are a good 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 possibly be 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 will they seem like a viable option, have a lot to monitor and the project doesn't need another thing to focus when other options are 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 would probably 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 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 could be a good option for powering the quadrotors.

Nickel-hydrogen batteries, also known as NiH_2 batteries, but it has one-third the energy density of a lithium battery, but it does have a long life. NiH_2 batteries can be found in certain space probes and satellites like the Internal Space Station (ISS), Mercury Messenger, and Mars Odyssey. These batteries are generally found in satellite applications and are therefore not a good option for this project.

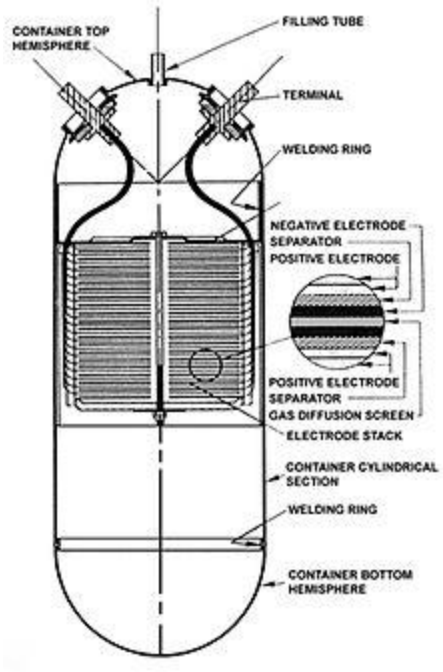


Figure: Inside of part of a NiH_2 battery

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 therefore will not be looked at to power the quadrotors.

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 doesn't make the cut for this project's power system.

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 would be a better idea to pick a battery that doesn't suffer from charging issues.

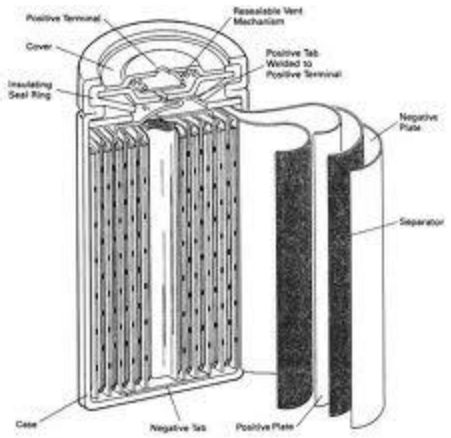


Figure: Cutout of a nickel-metal hydride battery

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 be 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 is not going to 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.

Sodium-sulfur batteries have a high energy density and long life but are of the molten salt type of battery. Molten salt batteries have to operate at very high temperature which is not a characteristic of this project so sodium-sulfur batteries are out of the question.

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.

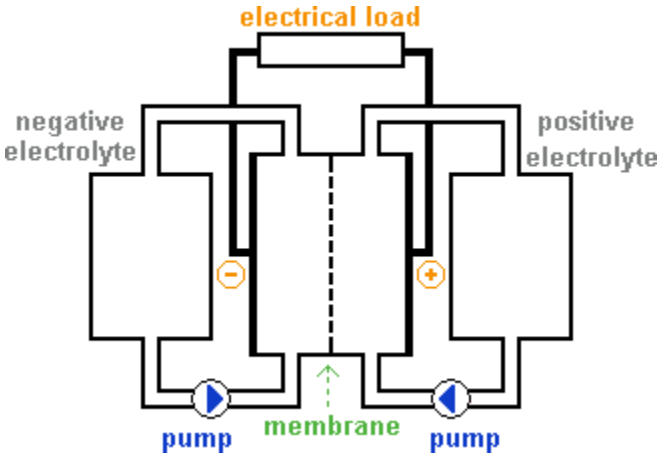


Figure: How a vanadium redox battery works

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 this project is focusing on an aerial vehicle.

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 common among hobbyists, and come in sizes that would be perfect for anyone looking into flying small hobby aircrafts, which this project would fall under. This is the battery for this project !

Figure: An example of a lithium-ion polymer battery

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.

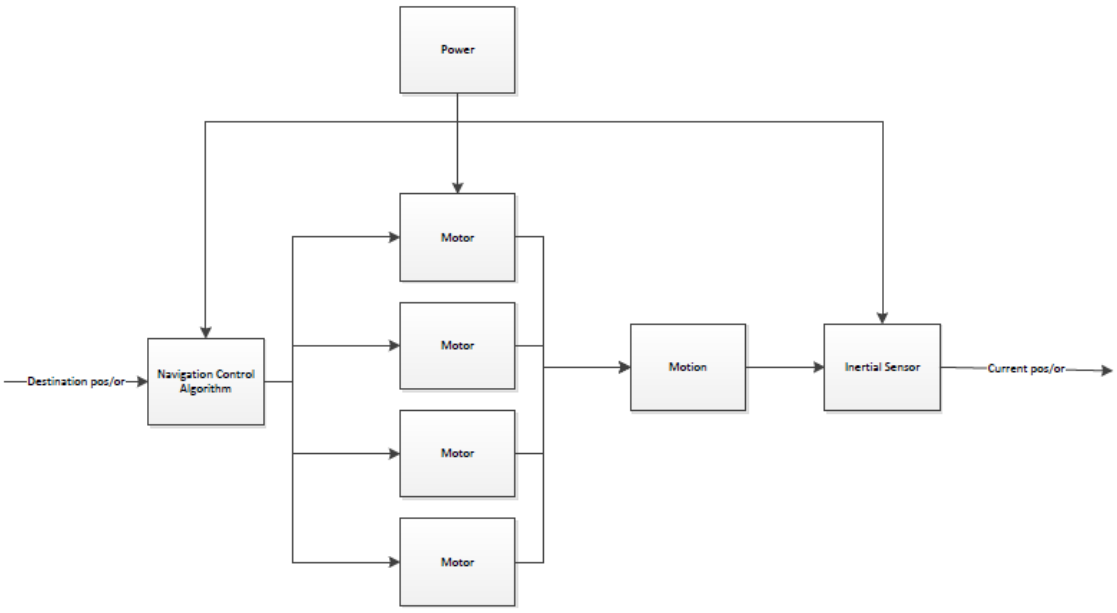


Figure: Block diagram of how the power will be distributed

The group has planned to make a printed circuit board (PCB) to disperse the power to the necessary elements in the system. The idea is to connect the LiPo batteries to the printed circuit board and then the power will go through the voltage dividers and regulators. After the voltage has been divided and regulated

it will be sent to the appropriate components in the system depending on how much voltage each element needs. The block diagram below is a high level visual interpretation of the flow of the distribution of power for the system.

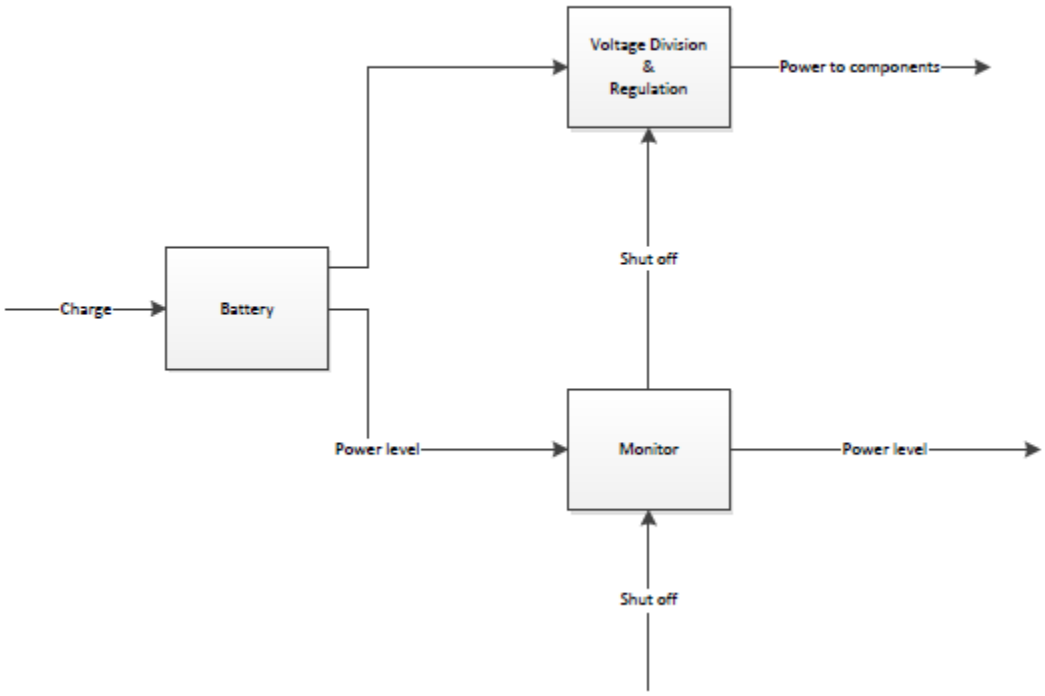


Figure: Block Diagram of how power will be regulated

4.4 Flight Control Subsystem

4.4.1 Summary

The flight control sub system on the A.Q.U.A.L.U.N.G. is 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) will be 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 will be used.

At the heart of the flight control system, doing all the calculations is the microcontroller. Within the microcontroller there has 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 is 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 don't get enough voltage (the recommended input voltage) the board might not turn on, rendering it useless. The input voltage, as previously mentioned, is 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 is 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 need 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 need to fit in a certain amount of finite space. So the area becomes important for building our frame, and when the IMU is place onto the microcontroller, the height of the two becomes important as some type of protective shield will most likely incase the electronics for protection. The weight of the flight control system also plays 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 becomes the topic, since the craft has to be able to lift itself off the ground and fly without consuming too much power too quickly.

Last, but certainly not least, will be 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 be purchased. But on a budget, picking and choosing becomes 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 is the accelerometer, for an aircraft such as the quadrotors that will be implemented for this project a 3-axis accelerometer should be 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. [4.4.2.2] 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):

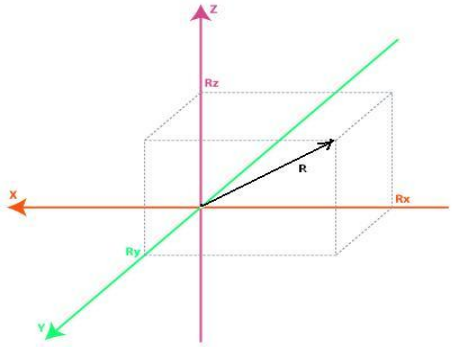


Figure 4.4.1.1.

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. [4.4.2.1]

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]

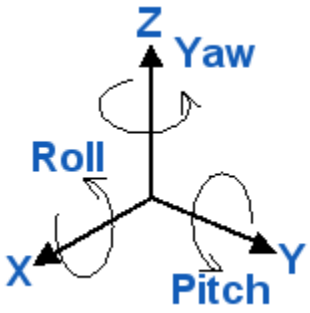


Figure 4.4.2.2

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.

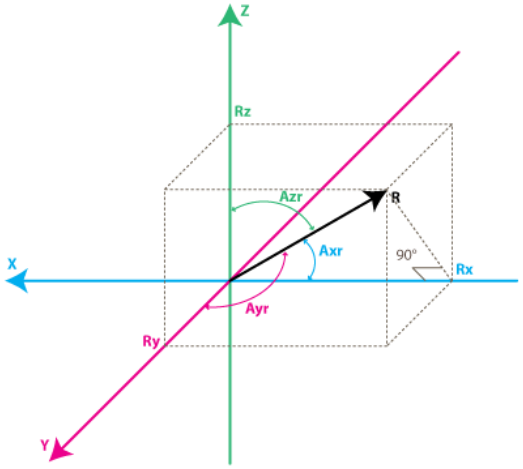


Figure 4.4.2.3

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.

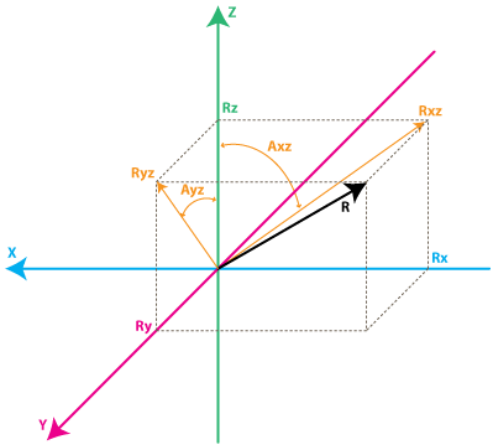


Figure 4.4.2.4

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 >

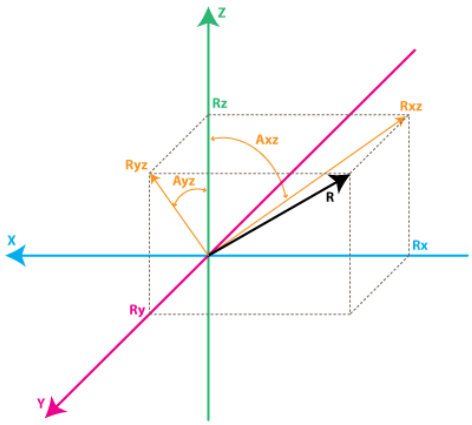


Figure 4.4.2.4

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 will go next, the quadrotor has to relay its position in space back to the central computer. This feat will be 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 will have to be removed and the integral will need to be taken. Once this is integrated the velocity is now found, one more integral will yield a position. The gyroscope outputs a rate of change, which is velocity, so this output will only need be integrated once to find position.

Errors in the integrations will occur; the trick will 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 is needed, any residual bias causes errors that grow with the square of time, so it does 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 will be, finding out approximately how long it takes for the position that is being relayed back to become unreasonably large and off the actual position. Whether it is a few seconds or a few minutes it will have to be corrected before that limit is reached, or else bad things can happen to the quadrotors, consequences discussed in section 8.4 Risk Assessment.

So what are some good ways to correct these errors? It depends on the application the IMU is being used for. If dead-reckoning is being used by ground vehicles than it might be easier to correct these errors, it also takes the z-axis out of the equation since the vehicle is going to be on the ground. For example a sensor could be place upon the wheel of the vehicle and every time it turns one whole revolution the vehicle has moved the distance of the circumference of the wheel. All that would be left to do is figure out direction which can 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 does have one advantage that could be used for checking its position periodically, and that is 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 will have 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 is one possible solution for the quadrotors in the A.Q.U.A.L.U.N.G. system because it uses vision to spot opposing players during the game of laser tag. The camera being used for enemy detection could also be used for visual "check-ups" on the dead-reckoning position. Through-out the course, there can be strategically placed icons, or symbols. These symbols will be mapped into the AI controller's memory just as the rest of the course is, and will be indicators of where the quadrotors are 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 recognizes it and relates it to a location on the map, then sends this position information back to the quadrotor. The quadrotor now has a true position reference to compare its dead-reckoning position too; it can either do one of two things now. It can “zero” out its readings and take the position of the symbol to be its position in space, “zero” is in quotations because this is not the true zero it is just a refreshing of the system to give the dead-reckoning a absolute for sure reference to its position in space. The second option it has is just reference this symbols position in space and use its past data to try and find out exactly where it is in space.

The second option will probably be the best bet for the goal that is trying to be accomplished. If the first option is used, than dead-reckoning is 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 will yield the greatest accuracy when determining position. The trick will be to check the position of the dead-reckoning before massive error sets in, because if the dead-reckoning error is way off and the camera sees a symbol, the average of those two things will still be off. Another thing to consider is not using the average of the two positions, Figure 4.4.1 shows that the average of the two positions will be right in the middle of them which is not what is wanted.

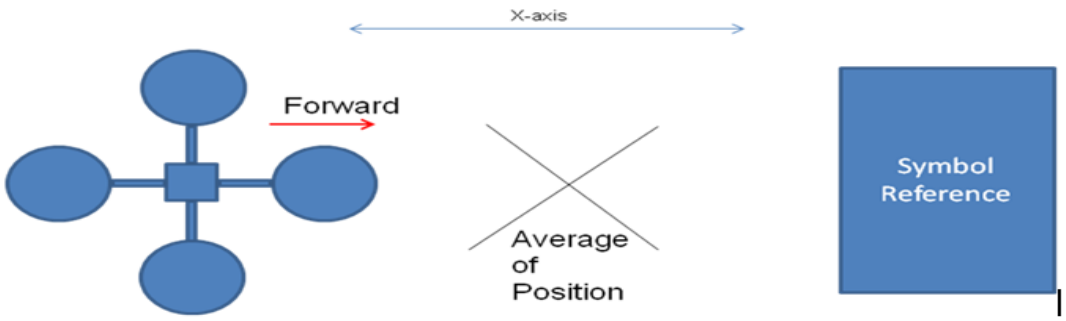


Figure 4.4.3.1

Using this symbol as a reference could however yield an accurate “zeroing” target. If the quadrotor is facing forward when it sees the symbol, it will then know that it is headed straight toward the symbol and can 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 can be accomplished.

Once the quadrotor has a firm grasp on its position within the course, it will be able to receive coordinate updates from the AI controller. The AI controller will be evaluating the positions of the other quadrotors as well as the video stream coming from the cameras; once it has all of these different variables it will make a

decision of where to send each quadrotor next. This is done by wirelessly sending a new coordinate position to the microprocessor, the microprocessor will then look at where the quadrotor is at the moment and maneuver the aircraft to the new coordinate position. This shouldn't be a challenge since all the microprocessor has to do it manipulate the voltages to the motors and away the quadrotor goes.

Once the building and experimentation starts on the project it will be a lot easier to solidify an exact formula for correcting error with dead-reckoning, as many options will be tested for their accuracy and reliability. At this moment in time, nothing is set in stone and all options will be considered. Although this is a huge challenge with heavy consequences for the project, the group is excited to tackle the problem head on and come up with a unique solution that will meet A.Q.U.A.L.U.N.G.s needs.

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.

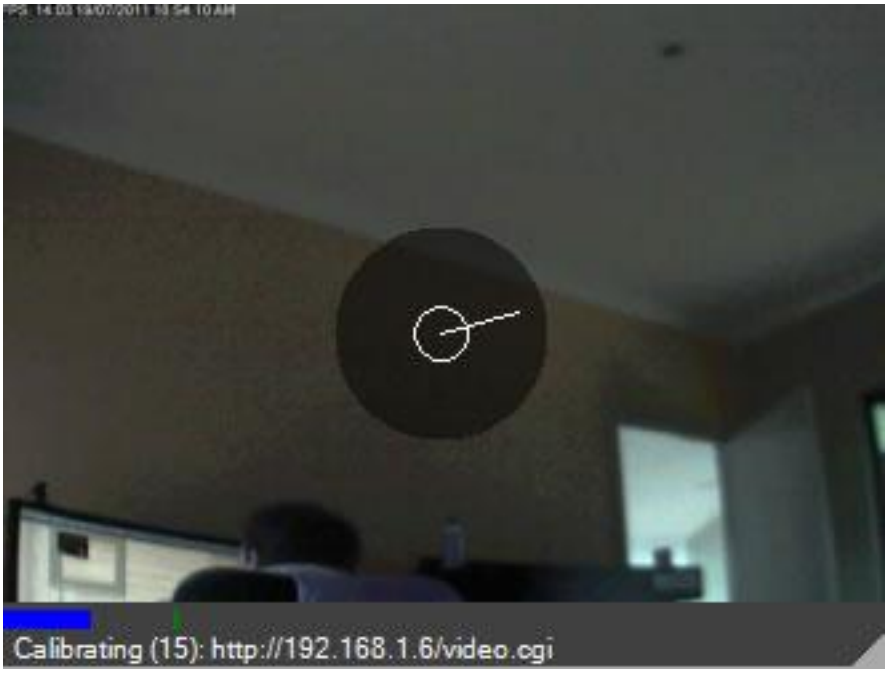


Figure 4.5.3.1-1: ISPY PTZ control. Permission Granted

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.

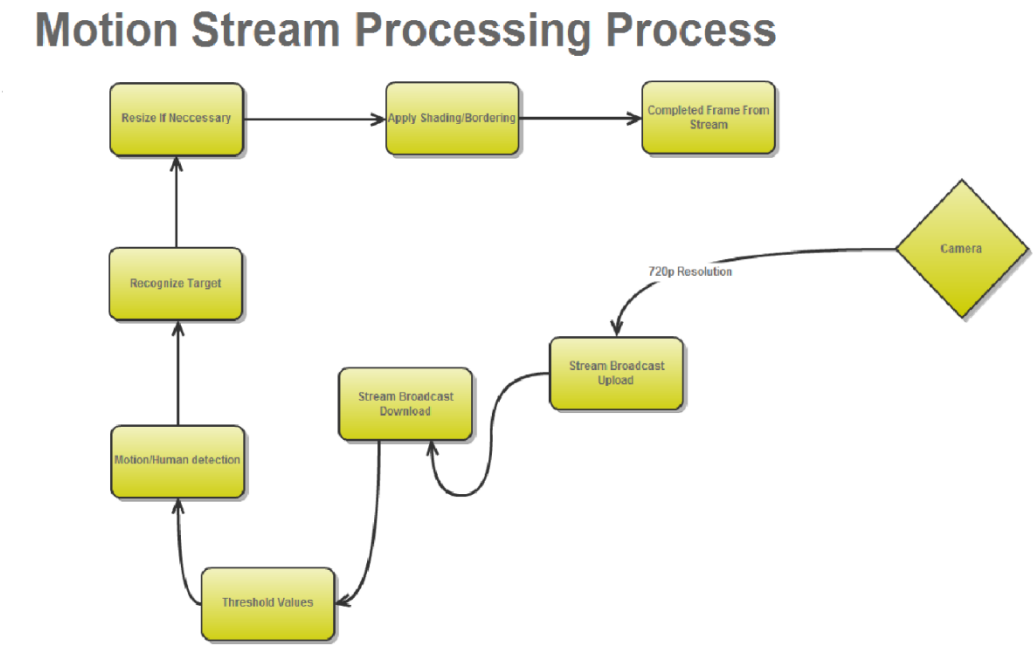


Figure 4.5.4-1: A.Q.U.A.L.U.N.G.'s Video Processing UML Class Diagram

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.

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. So this sets the dimensions for everything else that needs 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 package together with the PCB will be 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 battery for this project will be situated underneath the middle interface plate and will be LiPo batteries. They will 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.

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.

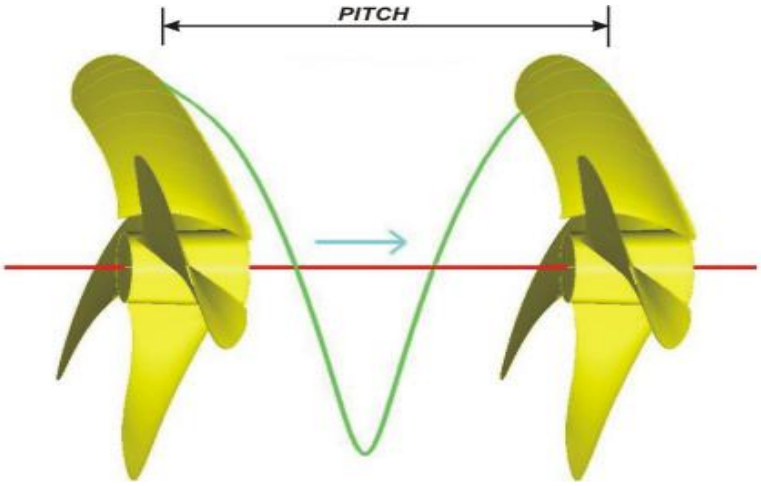


Figure 5.0.1

The pitch is the distance a propeller travels in one complete spin of 360 degrees. [5.0.1]

The camera for the quadrotors site 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. Webcams, although getting sleeker and stylishly small, come with a lot of unwanted plastic casing for the electronics and camera. This plastic casing will not be needed for the quad rotors as it will add extra weight to the aircraft which is definitely not a good thing. The camera cord will stripped, since it will not be using a USB interface anymore, and connected to the power supply and microprocessor.

For the laser gun and sensors, the same kind of idea that was used on the camera will be implemented here. The LAZER TAG multiplayer battle system will be 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 cheat armor looking vests. The lasers will be stripped from the guns and mounted onto the top interface plate, facing forward. While the sensors will be 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 will be stripped and connected to the appropriate component on the aircraft.

Legs for the quadrotors will be somewhat of a design challenge as they will be a completely custom piece added onto the frame. Since the camera will hang from under the bottom interface plate, the legs need to be able to shelter the camera on landings. The best bet for the legs will be two “sled” like components that will screw into the bottom plate. Figure 5.0.2 shows a very rough drawing of what the landing legs will look like.

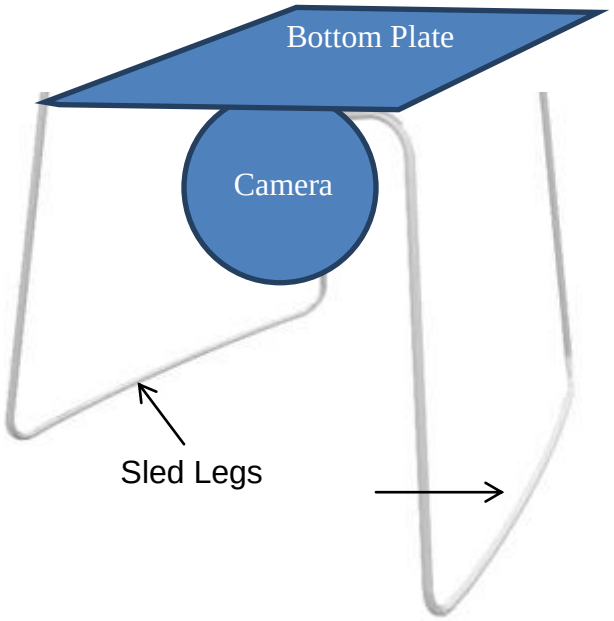


Figure 5.0.2

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.

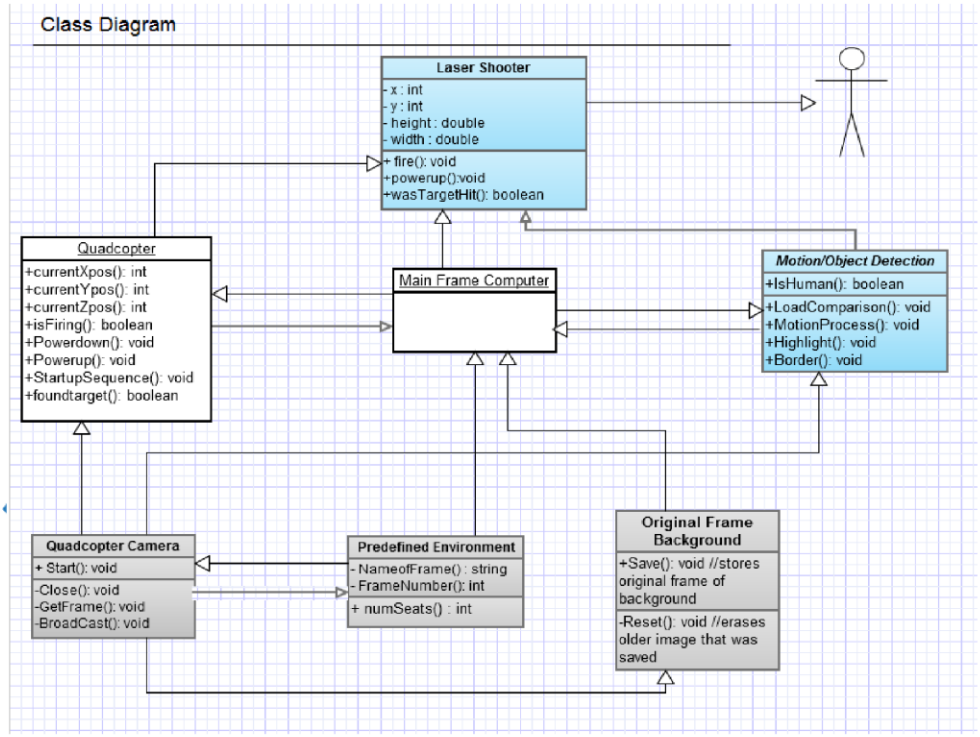


Figure 5.1-1: A.Q.U.A.L.U.N.G.'s Software Class Diagram

- 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 important as testing the quadrotors, is picking the environment in which to test them. For example, it wouldn't make sense to test the quadrotors outside where weather conditions could affect the quadrotors or the testing process so the group chose to pick an environment to test in that is close to what the quadrotors would actually be used in.

The hardware testing should be conducted indoors near a power outlet so that any testing equipment can be plugged in if necessary. The actual location of this stage of testing is not relatively important except that tests should be conducted inside and with plenty of workspace available. During the testing of the motors, its important to make sure that the motors can be properly contained if they lose control and decide to fly off. It is preferred that the room does not have high ceilings or else the motors could fly off and possibly be damaged.

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 is said and done it only makes sense to retest the quadrotors. The group has to check and make sure that all the hardwork and troubleshooting they put in has paid off and that the goals of this project are met. They will retest the quadrotors in the same environment that they were initially tested in.

Also, they will need to test the project in front of their peers, and their project manager, Dr. Richie. This test will be conducted at the University of Central

Florida in the engineering atrium. Dr. Richie has stated that he would like to see the quadrotors power on, lift off the floor and at least be able to hover for a few seconds to satisfy his requirements.

If it is possible the group would also like to test the quadrotors in a laser tag facility as that is what the project was designed for. Though initially Hard Knock, the closest laser tag facility near to campus, was not on board with the idea the group is continuing to reach out to the Hard Knock family to assure them that this project would be an asset for them as well.

7.6 Final Specific Testing

As the quadrotors will be built at the time of final testing, the hardware specific testing conducted previously does not apply. 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 fulfilled the proposed goals of the project.

The quadrotors must each pass a start up test; they should have no issues or power failure during this test. After the startup test is complete a hover test will be done, to ensure that the quadrotors can actually fly. They will be lifted off their standing position and held to hover for no less than 8 seconds. After the quadrotors have passed their hover test, a tilt/rotate/motion test will be done. It is not enough to just check that the quadrotors can hover but it needs to be verified that they can manuveur around. They will need to demonstrate that they can tilt left, and tilt right and move forward and backward. To be able to tilt left or right, the code will need to reflect that opposite rotors are doing opposite things, for example one rotor has to reduce the power to it, while the other opposite rotor has to increase the power so the quad can move in the direction of the 1st rotor.

Table: Flight test check sheet

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

During the final specific testing it is important to test the quadrotors ability to detect. The point of the project was to build quadrotors that will play laser tag with human opponents, so the quadrotors need to be able to detect that an object is in their view. Objects will be placed in view of the quadrotors' detection and they must be able to distinguish if the object is a target or just a part of the landscape of the playing field. If the object is detected to be a target the quadrotor must at least attempt to fire at the target.

Table: Object/ Environment Detection Test Check

	Yes/Pass	No/Fail	Notes
Detect Environment			
Detect objects in environment			
Detect target			
Target fired at?			
Target hit?			
Hit/Miss information recorded?			

The quadrotors also need to relay information back to the main computer about levels of the device. The group needs to check that the quadrotors are able to relay information back to the processor. It will be monitored if the quadrotors are relaying back power level information, the quadrotors constant position and if they are constantly streaming back video of their flight.

Table: Flight information test

	Yes	No	Notes
Power Level			
Constant position			
Streaming video back			

8.0 Design Considerations

8.1 Environmental Impact on Quadrotors

To ensure that the quadrotors are going to be able to perform to their highest potential, the environment in which they roam will have to be taken into account. For this particular project, the quadrotors will 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 will be 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 will, 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 will still take a significant amount of the force when it hits the ground. So it is very important to equip the quadrotors with protection for the weaker parts, such as the electronics. Even with electronics covered up, if it is a big enough fall, structural damage to the frame and propellers could result from the crash. All these things in mind, there is a sweet spot where budget and durability will have to meet in the middle and there will of course be risk of breaking due to an accident.

Another element of the laser tag environment would be humans themselves. The human element is 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 knows where each quadrotor is and therefore can make the appropriate decision on evasive maneuvers. However the quadrotors aren'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 result in a couple things happening, first the propellers would hit first and could cause for some damage to them. If the propellers go, then the quadrotor could 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 be 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 are not all unintentional like an accidental around the corner collision. It is 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 range 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 be to try and make the AI recognized such an attempt and know how to take evasive action.

Air-conditioning could play a factor in the environment of the quadrotors. If a stiff enough wind were to come from a vent, it could push the quadrotor to a certain degree without the quadrotor or central computer knowing it. This could cause the dead-reckoning to be off, and navigation could be 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 just give 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 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

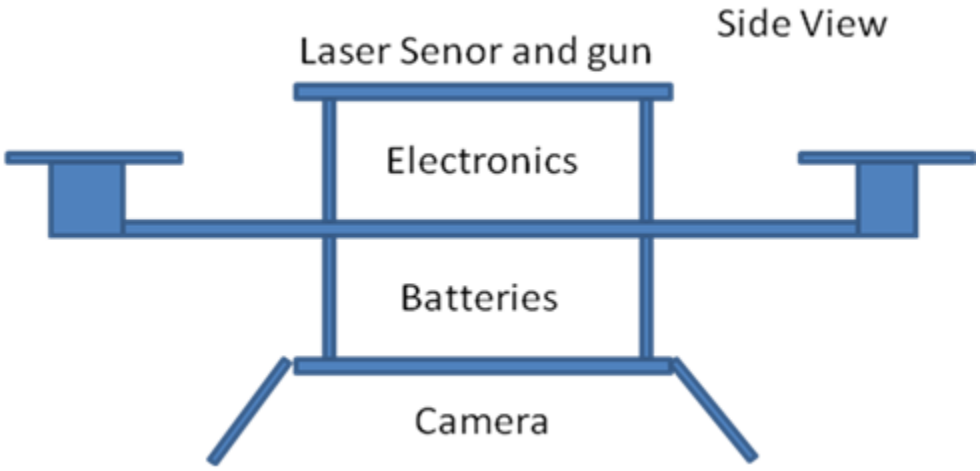


Figure 8.2.1 Prototype Design

The Assembly and Disassembly of the quadrotors for A.Q.U.A.L.U.N.G. starts 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 is 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.

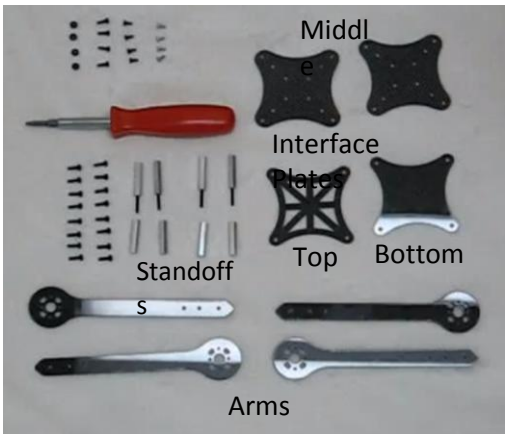


Figure 8.2.2

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.

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 should be assembled outside of the quadrotor then put in all together. On the bottom layer of these three would be the PCB, this will be doing the power dispersion from the batteries and is 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. Next would be 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. The final step for installing the electronics onto the frame would be to secure the PCB to the frame; this might be 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, is 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 rules out anything permanent for installation, and also rules out screwing the batteries down, as it would be 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 is 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.

Next up would be 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 is established via the IMU, make sure this is 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 are on the sides or front and back.

Camera mounting is the last step before connecting all the wires and installing the motors with propellers. The camera will be on the bottom of the quadrotor and again installation depends on the type of camera used for vision. As with the laser gun, it is 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 reduces the risk of damaging them while mounting and installing all the other central components. The most important part of mounting motor is making sure that the orientations are correct.

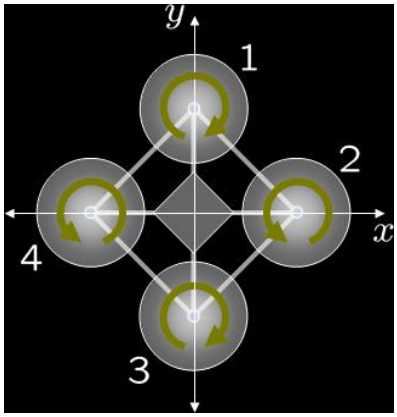


Figure 8.2.4

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 have to be spinning in the same direction, and motors two and four have to be spinning in the same direction, yet two and four's direction has 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 are mounted onto the frame it is 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 is on and pushing air downward, creating upward thrust, than the propeller is on correctly. If it is 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 is 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 are 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 also benefit the quality and lifetime of the quadrotors. The quadrotors are not the only parts to the system that need to be checked and maintained, the central computer system needs 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 is time consuming and a pain, all screws, bolts, and connective devices should be checked to make sure they are 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 can result in unwanted movement or shifting during flight which can 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 propeller. Again more often than not this will not cause the quadrotor to crash but it can put an unwanted strain on the batteries as it will take 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 is important as well. Even though the quadrotors will have battery power checking abilities, it would be disastrous for them to fail and have the quadrotor run out of juice in the middle of the course. Not only could they crash, 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 be to make sure all systems are go. Checking to make sure the quadrotor is talking to the computer and the computer is talking to the quadrotor, checking to make sure that the laser is working; the sensors are working and the camera is able to see. Then it is aggressive tactical flying from then on.

Post-flight checks should not be as intensive as pre-flight checks since, well, everything pretty much gets checked before the flight. However, a post flight check should consist 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 is imperative to have it repaired before the next flight. A couple key areas to look at are the propellers, checking for any nicks, chunks or cracks from hitting something. The frame to make sure everything is straight, as running into something can bend 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 is not as thorough as the disassembly will be, just a quick glance at everything to make sure nothing extreme happened during flight.

The disassembly is a way to make sure everything on the quadrotor is 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 is the motors, check the motor to make sure that the shaft isn'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 is as solid as possible is important. The camera is mounted underneath the quadrotor so making sure no damage has occurred from a possible rough landing or collision is important as well, of course checking the connection between the camera and microprocessor is just as important as the connection between an eyeball and the brain, so upkeep on these is essential. The laser guns and sensors are next, for the gun and sensors the connections need 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 can get in the way laser so the gun and sensors need to be cleaned and taken care of is they are going to perform well. Moving on up, batteries need to be inspected for a couple things. Of course the common theme of every component, the connectors need 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 is 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 has acid in it, acid stratification can occur, knowing what kind of battery is operating the quadrotors and their weaknesses will help keep you ahead of the game in terms of maintaining the batteries. The electronics will take a little bit longer to inspect and it is not just visual inspection needed, measurements of output voltages and currents should be 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 is the last thing in the disassembly that needs to be inspected, this is more of a visual inspection, unless stress testing is available, to make sure all four arms are straight with each other, a level could be 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 is complete, following the assembly instructions will put the quadrotors right back into flying order.

The last component of the system that needs to be maintained is 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 heat, 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 is a lot of risk associated with the project as a whole, risk being the level of complexity and chance of something going wrong. This project is an attempt to combine several advanced topics into one fast moving dynamic system. Breaking this project down into subsystems is definitely a strong strategy for trying to make this project successful, but it can 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 is 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 has is 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 the pathing for the system is working correctly, the quadrotors are 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 is 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 is not an easily fix 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 is 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 it's 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 is. Two major things could go wrong with flight control that could result in catastrophic damage to the quadrotors, stabilization failure and navigation failure. A failure in stabilization would mean 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 occur, because just like the aircraft our guess would be unstable. The second major failure could occur with the navigation of the quadrotors, due to error build up within the dead-reckoning. If the quadrotor does not know where it truly is, 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 has total trust in the AI. So in order for this partnership to work out well for both parties, there needs to be accurate information coming from the quadrotors to the AI controller.

Within the image processing unit a couple things could go wrong, not recognizing humans or false identification of humans. While some of these errors are not as disastrous as the flight controls can be, these errors would cause the project to be unsuccessful. If the quadrotors are not able to properly identify the enemy than they will 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. On the potential for serious damage side of 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 will be considered a success is if all the subsystems work together and work as they should. With this being said, a reality sets in about how challenging this project will be. 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, checkmarks 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 quadrotors 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 quadrotors 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 quadrotors to play laser tag against people.

Figure: Dr Kumar's quadrotor

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 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 present 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 quadrotors 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 quadrotors. 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. They are in the works of putting together a petition so interested participants can waive the liability and play laser tag with the quadrotors and test their gaming abilities.

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 next step is to receive notice back from the companies and individuals that the group had contacted. Once the group has seen who is willing to give a monetary donation they can evaluate their budget again and see if there is a need to pool together more funds to purchase all the necessary items from their bill of materials. After that is decided they can actually begin the process of finding the best deal and lead time if some things need to be bought online, and then they can purchase.

The next step is to start building the code necessary to make the quadrotors autonomous. Consideration needs to be made to get them to lift off and be able to turn and hover and make other related movements. The code also has to account for battery level tracking, “enemy” tracking and also relaying its position in the test or rather game environment back to the control center. The software should be able to record the exact amount of power each motor is outputting through the entire course of its flight time. It must also be efficient enough to do all calculations necessary as everything will be happening in real time, and the navigation calculations are a very important part of the project because of the nature of what the group is trying to accomplish.

To make sure that the quadrotors operate effectively an important task is to run hardware tests on the parts. All parts need to be tested to ensure that they were not harmed or defective before the group receives them. If electronics were harmed before the group had a chance to work with them it could cause a delay in the schedule or defects in the project’s effectiveness.

After receiving parts and compiling most of the code necessary, the next major step for the project is to actually begin the physical build and assembling of all the electronics. Using prebuilt quadrotor frames the group plans to attach A2212-

13 outrunner motors, a wireless microprocessor, a webcam, a lithium polymer batteries and an accelerometer/gryometer to get their final product of a quadrotor that is capable of autonomously playing laser tag against human targets.

After initially putting the hardware and software together it's important to do an initial test to see that everything starts up and can do simple motor movements. A start-up test and small motor tests should be run to insure that the code is correctly doing what the group wrote it to do. This is the time to assess what troubleshooting needs to be done and if any reconfiguration of the physical build is required. After all comments have been recorded and a list of changes and improvements have been made the group will work to correct any issues.

The next major milestone is to conduct final testing for the project. All capabilities and project requirements will be tested during this phase of the project. This is the time to check that everything works as it should and that the project is able to accomplish its intended goals. The quadrotors will be tested as if in true game mode and their performance will be monitored. The group will look for any minor bugs or kinks and note them in case an additional troubleshooting session is needed. If there are any problems the group will address them to get the quadrotors in prime working condition.

The last major milestone is to attend the Senior Design Showcase and present the project to the project manager, Dr. Richie. The group will get to show off their project to the other senior design groups and other interested showcase attendees. They will be able to answer questions and preview to viewers what the project is capable of.

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 quadrotors 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 $500 and a promise from an interested friend to purchase the motors. Other individuals and companies were also contacted for possible sponsorships or parts donations and the group is currently waiting on hearing back from some.

When writing the initial report for their project manager the group got together to discuss what parts would be necessary to achieve their goal of quadrotors to play laser tag against human opponents.

The discussion came up whether or not to buy prebuilt frames for the quadrotors. While it would probably be cheaper to buy parts and assemble the quadrotors themselves it would also create even more work for the group and would lead to additional testing being done to assure that the quadrotors alone were functioning. 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 quadrotors but as their financial resources were limited, four quadrotors sounded a tad bit better.

As a result of the quadrotors 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.

Since the objective of the project to was to play laser tag, it would follow that the quadrotors 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 quadrotors, 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 quadrotors, four motors per assembly were required totaling 12 for the entire project.

Table: Estimated Budget for Project

Item	Quantity	Price per Quantity	Total Price
Quadcopter Frames Prebuilt	4	$250	$1,000
Computer for Processing	1	$500	$500
Option : Spy Net: Spy Strike Laser Dueling System (hit sensor and IRB (laser))	2	$23	$46
A2212-13 Brushless Outrunner Motor	12	$10	$120
Slowfly electric prop 1045R 4-piece set	4	$3	$12
TI msp430 wireless microprocessor	4	$50	$200

Webcams	4	$15	$60
Accelerometer/Gryometer	4	$100	$400
AirSplat 11.1V 1600mAh 25C LiPo Battery	4	$30	$120
Miscellaneous (Wires, Solder Iron, Solder, LED's, etc	Unknown	Unknown	Allocating $150
		Final Price of all parts =	$2,608

The table above represents the estimated budget as first devised by the group. Since this report was written in December and parts have yet to have been bought, specific part numbers are not necessary for most parts. Most of the parts listed have various brand names with equivalent performance capabilities and specifications, and the group plan to shop around when the time comes for the option with the best price for the performance desired.

The only part that has a specific part number is the processor which the group has requirements that it be an intel core i7-3770k Ivy Bridge. The motherboard will be a reliable brand with an LAG1155 socket, a Z77 north bridge, SATA 6Gb/s, USB 3.0, PCIe 3.0 x16, and at least ATX form factor. It will have at least 16GB of RAM with 9-9-9-27 timing or better and at least 60GB SSD HD. It will have an 850W power supply with 80 PLUS Bronze or better certification. For Wi-Fi, the most cost effective of the following options will be chosen: included on motherboard, PCI slot, and separate router.

The group's initial estimated budget totaled $2608. With the sponsorships of $500 and someone promising to buy the motors this leaves a remaining balance of roughly $2000. If the group doesn't hear any positive feedback from the remaining companies and individuals, that would leave a balance of about $500 per person. That amount of money is still high and gives the group motivation to find people to help by giving a monetary donation or by trying to find someone or some company that wouldn't mind donating parts.

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.

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. He was also able to provide a list of contacts that could be of some assistance. Keith Brawner was one such contact given by Dr. Gonzalez that was also able to provide a list of companies that could be potential sponsors.

Some companies and individuals that have yet to respond but have been contacted include: Aurora Flight Services, Aerovironent, National Instruments, FLIR, LaserShot, Boeing, Rockwell Collins, Lockheed Martin, Navy Center for

Applied Research in Artificial Intelligence (NCARAI), Defense Advanced Research Projects Agency (DARPA), Dr. Gita Sukthankar, and Dr. Daniel Barber. Aurora Flight Services is involved with the development and manufacturing of unmanned systems. Aerovironment is a manufacturer of unmanned aircrafts and electric vehicles. National Instruments is a major manufacturer of measurement and control tools to assist scientists and engineers. FLIR is a company that does work with night vision, thermal imaging and cameras, integrated surveillance for both consumer and industry related needs. LaserShot is a weapons and firearms training solutions company that manufacturers for law enforcement and military needs and also for hunters education purposes. Boeing is one of the most well-known aerospace companies and one of the largest manufacturers of military aircrafts and commercial jetliners. Rockwell Collins has a long list of products that include aviation electronics mission communications tools; their systems and products can be found in most aircrafts around the world. Lockheed Martin is a major corporation that has a hand in many different ventures, as it pertains to this project they design and manufacture high tech aircrafts, test systems to maintain the integrity of their product and also do simulation and training to keep their customers in the loop and informed about how to maneuver and use their products. NCARAI does basic and applied research in artificial intelligence. DARPA is an agency that was created to help defend the nation's security, they conduct research and implement creative solutions in different fields including (as it pertains to this project) electronics, sensors, communications, and weapons. Dr. Sukthankar is an assistant professor at UCF with an interest in robotics; she received her Masters in Robotics and her Ph.D from the Robotics Institute at Carnegie Mellon. At UCF she is a director at the Intelligent Agents Lab which deals with social computational systems. Dr. Barber is a research associate at UCF who serves as Faculty Advisor to the Robotics Club at UCF. He has experience with unmanned systems, control systems and computer vision. He has also designed multiple autonomous systems and has developed tools to process data from multiple physiological sensors.

All of these companies and individuals would make great sponsors and mentors as they all in some way, no matter how small, align with the goals of this project.

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/

[4.4.2.1] 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