Final Project Design Document

December 11, 2015

ORQC

Oculus Rift QuadCopter Controller

Group 6

Gustavo Gonzalez
Matthew Grayford
Gunnar Skotnicki
Craig Thompson

Contents

1. Executive Summary

Everyone has their own different ideas of what the “future” is going to look like. Varying levels of advanced technology that could only exist in our wildest dreams. Many people share the dream of getting one step closer to those “futuristic” pieces of tech, and with that in mind we wanted something beyond the ordinary, something novel.Our idea first originated with designing a Quadcopter and allowing it to be controlled from a wrist watch. But we immediately found that that had already been done, we needed something with more of a kick. Next we discovered that a member wished to try virtual reality headset technology like the Oculus Rift or Google Glass. With these ideas at hand we came up with ORQC, an Oculus Rift QuadCopter Controller. A combination of ideas brought together to create an all new experience.

Technology has been advancing before humankind at an extraordinary rate. Each day newer and better items are created to better a cause or mission. These items can include new personal devices, medical equipment, vehicles, games, etc.. Noting that all these new creations can come together to form something grander than the original is a specialty of engineering. Through this project we work to bring multiple tools and utilities together to create something useful to many and usable in many situations. There has always been an interest for “virtual reality”, it has been a concept quite common in Sci-Fi and a common fragment of how the “future” is perceived. We want to make use of this interest, and show just how much we can do with current technology. We plan to bring virtual reality to a newer and more immersive experience that everyone can enjoy.

The base component of our project is the copter the user will be piloting. This copter will have cameras attached to the front of the copter streaming live feed to the user to view through the Oculus Rift. We want this to be our “eyes in the sky” to give an exciting view to the user, but we want to go beyond just seeing through the eyes of a drone. One of the best types of immersion is to enjoy a real hands on experience. We feel that current implementations are being limited by the controllers we have available. Keyboard and gamepads limit the experience. Everything focuses entirely on just the visual feed, and even if you create the most realistic possible visual immersion, you can't help but wish you could go further, touch and grasp the things you are looking at. We can't go quite that far realistically with what we have available to us, but we really want to integrate that hands on feel.

To do this we are going to create a glove that will allow a user to operate a quadcopter remotely and experience what they see first-hand in front of their eyes. With the flick of a wrist we want to have all of the controls necessary to pilot our device, and leave room for further possibilities (Game applications and hobbies). The controller will have an accelerometer and gyroscopes to allow for motion based controls, as well as pressure sensors to create “button” inputs using the fingertips. We hope that the combination of “VR” visual feed and the implementation of will give users a brand new experience, that will hopefully inspire similar projects from hobbyists and grow the VR community.
Listed below are the motivations and goals we have for our project from our current team members, their biographies, and acknowledgements to supporters of the project.

1.1 Motivations

When we had all individually registered for this course none of us were quite sure what we planned to design going into this. But we all knew we wanted something interesting, and importantly something fun. We had seen previous projects, the home security systems the rovers lots of interesting ideas but none of them really clicked. When the basis of a copter was suggested we began to build off that, just a drone on its own was not enough, we needed more. The initial suggestion was an automated drone that would float around and take pictures of the user, a selfie-drone, however that project was already done in a much grander scale, we did not think we could match it or even come close, but the idea of a drone was set. The Oculus Rift was put on the table as a possible accompaniment to the drone and we were sold, while the actual purpose of the ORQC system was left in the air, we decided we wanted to make it possible and sought to design the copter/controller combination. This could be used for games (Utilizing the graphic overlay we wish to incorporate, or what was also considered was an artistic approach. You could carry this setup in a portable case, bring it with you on a hiking trip and take some pictures of hard to reach views. A group member during a trip to WonderWorks mentioned running into a child who was disabled who was unable to take part in the Laser Tag Experience, and were motivated to incorporate something similar into our own project. Virtual Reality does not have to only be something we use to experience fantasies in greater detail, but could also bridge a gap to experiences we take for granted, and to help individuals who would otherwise not be able to experience that first hand , take part. Overall we were excited to create something that had so many possibilities for improvement as we go through the design process.

1.2 Goals and Planned Objectives

The ultimate goal is of course to have our fully functional copter interacting with little to no delay with our custom hand controller as well as the oculus rift. This controller should also be able to run completely on its own for use in any other applications as any other USB device. The Camera attached to the copter should be able to transmit live feed through a stable Wifi Connection to our host machine, which then takes the feed, and depending on the setting either distorts the feed before outputting to the screen on the oculus, or it will display the feed on a regular monitor (for testing purposes). This copter should be able to be fully controlled using the hand controller, and ideally the controls themselves should be easy to understand. While not immediately planned, a possible addition would be to add information about the position of the hand controller into the graphical overlay itself to help the user with orientation and centering.

The hardest part of this project will likely be two separate issues. Properly utilizing the Oculus to receive the camera feed in a viewable matter, the other is the creation of the controller itself and the circuit board that will be required as well as the programming of the arduino micro controller. We decided the best way to tackle this is to split the bulk of the work into teams within the group itself, to help focus our time and efforts more clearly. Extensive research by each group member will be required in their appropriate fields. A

brief outline of proposed research and responsibilities, as well as group subdivisions is detailed below.

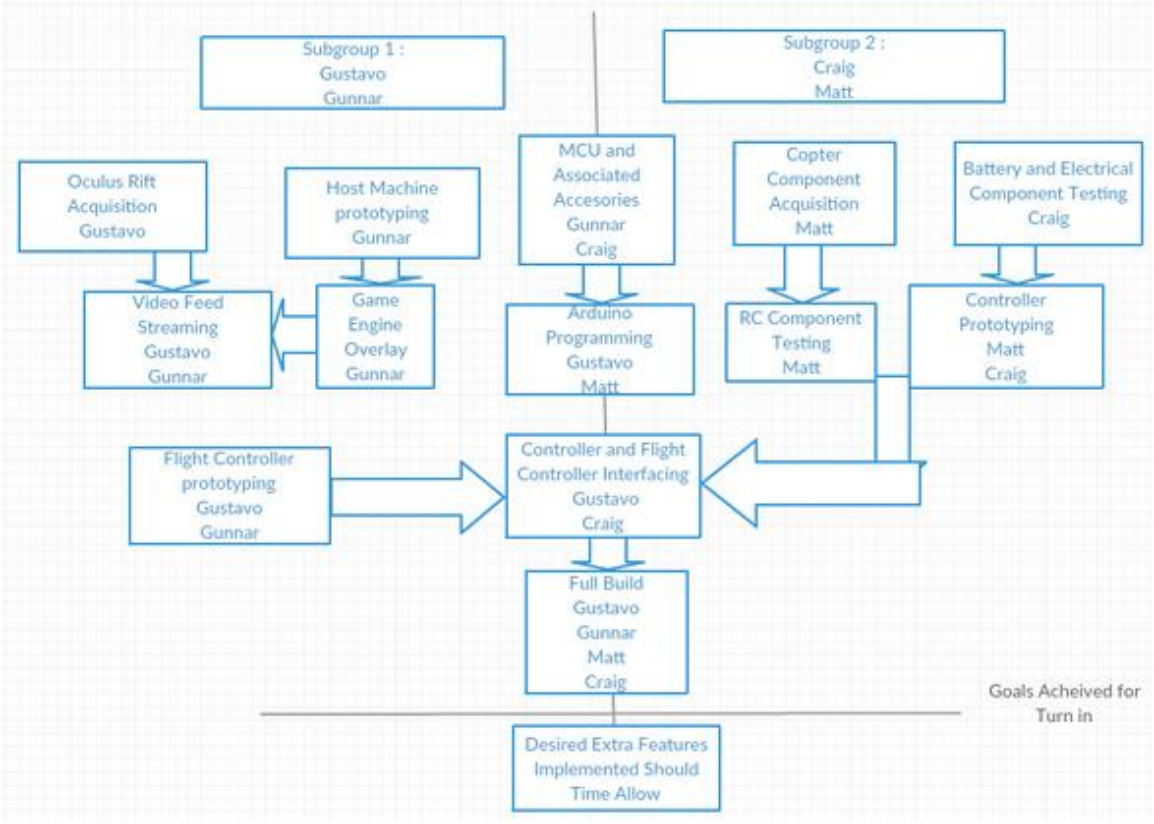


Table 1.2A Workflow Outline for Expected Milestones

Subgroup 1 : Gunnar and Gustavo shall focus on any software aspects of the project, learning how to use the Oculus as well as studying the applications of the Raspberry Pi. They will learn how to get a steady stream of video from the camera using the Raspberry Pi device and how to properly receive it on the host machine. They must also study the software involved with the Oculus and research a way to incorporate any features that could help with the immersion. To do this the group shall attempt to acquire the OR DevKit2 to get hands on experience and access to the developer's libraries and API's for a better understanding of how to proceed in the implementation. They shall also assist Subgroup 2 with the programming of the arduino in the development of the hand controller.

Subgroup 2 : Craig and Matt will be in charge of researching the EE aspects of the design and will primarily be working on the hand controller. They will also be assisting in power consumption and the batteries that will be powering our quadcopter drone. The group was supplied information towards tutorials on learning how to create a circuit board, and they will spend time studying those materials and make the executive decision on whether we have the capability of mounting the components onto the board itself or if we shall have to pursue alternative methods.

1.3 Team Member Biographies

A team consisting of three computer engineers and one electrical engineer. On the first day of group discussion after the group was formed, the members decided who specifically planned to do certain parts of the project. Gus and Gunnar will focus on the software aspect of the project while Matt and Craig will focus on the hardware and component construction of the project. By dividing and conquering separate portions of the project the team can finish different components at nearly the same time without bottlenecking any portion in particular

1.3.1 Gustavo Gonzalez

Gustavo.Gonzalez93@knights.ucf.edu

Gustavo was born April 23rd, 1993 in Caracas, Venezuela. Moved to Florida from a young age, and has held an interest in technology from a young age. Graduated from Freedom High School in Orlando, Florida , and moved on to further his career at UCF as a Computer Engineering student, and has spent his entire college career at UCF.

No Internship experience, but has experience in group software development through various personal projects such as programming a BASIC 2 Stamp microcontroller to regulate servo motors in the navigation of a miniature paddleboat, as well as a software development project titled Cylindris in which the group developed an android game app modeled after 3D tetris utilizing OpenGL.

Focus on the project is helping with the development the software to bridge the Oculus with the drone and assist anywhere else possible, and past experience with OpenGL will be helpful in modifying video feeds. After graduation Gustavo hopes to sign with a software company where he can continue to learn and improve on his skillset.

1.3.2 Matt Grayford

dragonbolt1@knights.ucf.edu

Matt is Computer Engineering student with 2 years at Embry Riddle Aeronautical University and 3 years at UCF. He has experience in Technical Writing and Software Configuration Management with two companies, first as an intern and after as an independent contractor. His current job focus is Troubleshooting and potential Circuit design.

His main focus in this project relates to the construction and operation of the Glove and it’s components to the project. He is also working as the Semi manager of the project by keeping the group coordinated and up to date with various tasks assigned to each member to ensure collaboration is at a max.

1.3.3 Gunnar Skotnicki

GunnarSkotnicki@knights.ucf.edu

As a Computer Engineering student Gunnar started his college career at UCF. Early on he knew he wanted to work with software, but wanted to learn about hardware as well. His current focus is creating the network communication between components, game engine support, as well as configuration of all machines at the software level.

Gunnar is staffed as a software engineering intern with AVT Simulation (Applied Visual Technologies). His experience with simulators, both large and small, contributes experience of distributed simulation, which he believes can apply to this project's scope. Other previous experience consists of Linux server administration and configuration, as well as embedded programming specifically Arduino Uno Rev 3 / TI MSP430 based.

By tying personal interests in microcontrollers, work, and school experience Gunnar will further his knowledge of engineering complex systems, and designing elegant robust software. He hopes to pursue a software or systems engineering career in simulation as it has increasingly becoming a passion.

1.3.4 Craig Thompson

Craig.T.UCF@knights.ucf.edu

Electrical Engineering student who spent his entire college career at UCF. Interning for a contracting company. His focus on the project is to keep the quadcopter running with the additional components and to help with the construction and operation of the Glove.

Craig's work at SGM engineering helps him slightly in determining load for the systems that are to be designed. Although the MEP work done at SGM does not directly co-relate with such topics as Quadcopters, Oculus Rifts, or PCB boards, caution and code that requires panels to be loaded for no more than 80% of total possible load and then a reduced 90% of that for further receptacles and other miscellaneous loads.

Craig hopes to learn more about PCB board layouts and how to make components co-relate and work together without frying any of the components. He also hopes to learn about RC and WiFi communication; how they work and their limitations. This, he will learn through observing his teammates, but Craig will not be taking a direct hand in the RC or WiFi configurations other than what will be considered for power consumption.

1.4 Acknowledgements

High 6 Senior Design project team for allowing us to note their project with use of glove for sign language as well as use a picture from their assignment to illustrate its capabilities

A thanks as well to Dronenet Senior Design project team for allowing us to use a picture from their project and discuss with us the power use of the Quadcopters they used and different choices they made with their senior design project.

Alex Berliner : UCF Alumni currently living in Orlando Florida, friend of Gustavo Gonzalez. Has experience with the Oculus Rift DK2, and has provided his kit to the group for a lesser price. He expressed interest in seeing the project develop, and will be a continuing source of knowledge for the group as they begin working with the software.

Henry Ramon Gonzalez, La Boyera Construction: Father of Gustavo Gonzalez, expressed great interest in the groups project, and wished to support them. Granting full access to group to various tools and machinery that could assist in the production and development of the QuadCopter as well as provided additional funding to assist in the acquisition of key project elements, most notably the Oculus Rift.

Reds Hitchon, grandfather to Gunnar Skotnicki, contributed his HPQ1 quadcopter to the group upon hearing the idea. He was fascinated with the idea of flying the quadcopter in a much simpler manner. Even as a long time pilot, he felt the HPQ1 was hard to fly and was eager for us to implement an easier solution than the classic controls.

2. Project Requirements and Design Constraints

This project as detailed before consists of 3 major components. The Oculus Rift, Quadcopter and Controller. Within these major components both the Quadcopter and Controller have their own sub-components that create them. These sub components will be discussed in further detail through the remainder of this report. Below is an initial draft of how we conceptualized our system to function. The Oculus communicates with the host machine for information from the drone, while the drone sends the video feed to the host machine. The Controller sends its commands to the host machine to help display information on the graphical overlay, while the controller sends commands to the drone. Research will be done to see exactly how the signals shall be sent out, whether we can accomplish this through a direct link on the wi-fi connection or if we shall have to incorporate RC signals from the glove. This is not the final draft and will likely see some revisions as we better understand how these systems will interact with each other in a functional manner.

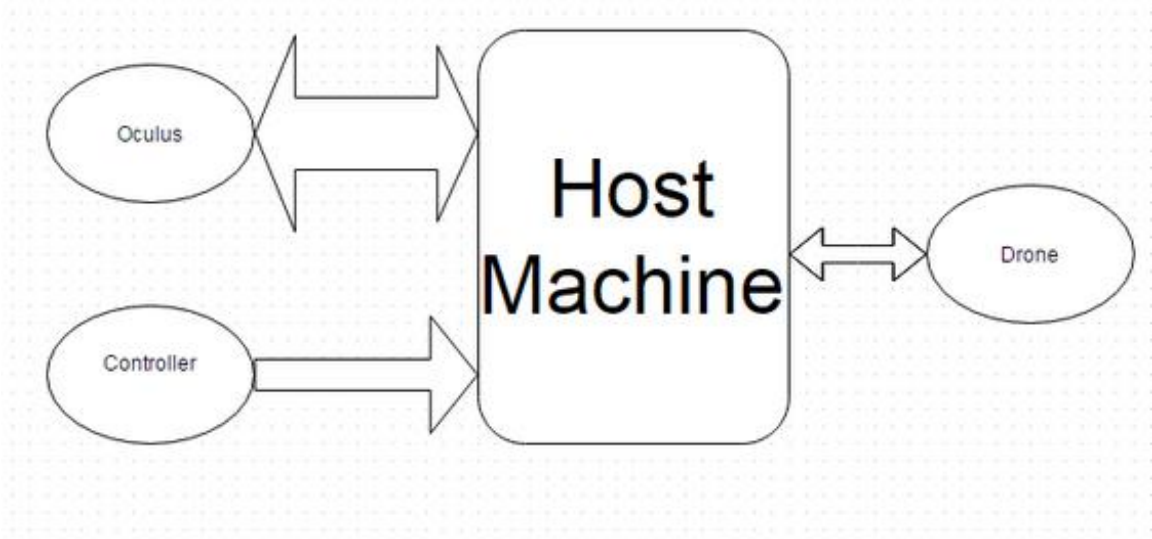


Figure 2A Block Diagram of the System

The below listed Problem statement and customer requirements are to layout the plan behind our project and its full use hereafter. With emphasis on the independent use and combination of the parts in question, the team can create a project that will enhance certain entertainment environments in the current technological age.

2.1 Problem Statement

Today we have video games that give a sense of immersivity with an unreal world like we have never seen before. A person can play as a different race altogether and become the best elvish marksman or the world's most fearsome orc warrior. Video games give us a chance to save the world thousands of times over with thousands of different characters. However, video games are based solely on the unreal. On things intangible and without significant real world consequences.

In today's world, we have incredibly advanced feats of aerospace, mechanical, computer, and electrical engineering to give household toys the ability of flight with tight, responsive controls. We can fly miniature planes, helicopters, and quadcopters remotely and within minutes without any form of professional training. But these forms of home entertainment come with the drawback of not being in the cockpit, of very obviously being a replica of the real thing. The person flying the toy aircraft can feel a significant sense of missing out on something.

To combine these two worlds of video games and tangible miniature aircrafts, we propose combining the technology of virtual reality with instant video streaming. Using the Oculus Rift as our tool for virtual reality and combining a quadcopter with a camera and a WiFi adaptor, we can stream the video feed of the quadcopter to the Oculus Rift to produce a new level of immersion. We hope that our design is not limited only to the field of video games, and that it can find a place in various hobbies that could make use of this added level of immersion.

By combining these two aspects we hope to expand into a more novel form of entertainment than what is currently available. Mixing common hobbies into one newer technological setting. And expanding the field of what is possible for future developers and hobbyists that share our love for technology and combining it with forms of entertainment everyone can enjoy.

2.1.2 Design Specifications

The design specifications we will be focusing on are physical, hardware, and software. These are not standards or constraints but more so guidelines for the team to follow and have a better focus towards completion

Physical Specifications

- Any LED's on Hand Controller should be easily visible
- Hand controller should be adjustable or be able to fit various size hands
- Hand Controller should be comfortable to control
- Exposed wiring should be kept to a minimum on the Hand Controller.
- PCB encasing for Hand Controller should be kept light weight to avoid strain on the user
- Copter should ideally be able to lift around 2-3 lbs worth of cameras and batteries.
- Any components attached to copter should be easily removable and modifiable.
- Copter size should be capable of being piloted in an indoor household setting.

Hardware Specifications

- Components must run off (Insert planned battery voltages) or less
- Copter should be able to run for at least 7 minutes
- Hand Controller should be able to run off powered USB connection.
- Hand Controller drivers should be handled to act preferably as a regular USB Keyboard I/O
- All electrical components should have an off switch for safety.
- Hand Controller should be able to communicate with Host Machine
- Host Machine must have strength to run Oculus.

Software Specifications

- Source Code should be written in possibly C, or C++.
- Programming should not exceed given memory space on chosen Microcontroller and similar variants
- Should be uploadable to microcontrollers through USB connection.
- Should be kept on an easily accessible repository and properly maintained

2.2 Customer and Customer Requirements

By combining new and improving technology with a well known past time (Driving RC cars and copters) we hope to capture a wide target audience. The younger generation and the old will be drawn to our product (accessibility and familiarity, and the lure of "next-gen" tech possibilities. An age range of 13-20+ will be our predicted main user base, as this application could find its home in a variety of settings such as tourist locations (eg Wonderworks, Hardknox) with modifications to the base unit. The age of 13 is also a standard we must follow due to the minimum age requirements specified by the Health and Safety warnings on the Oculus Rift.

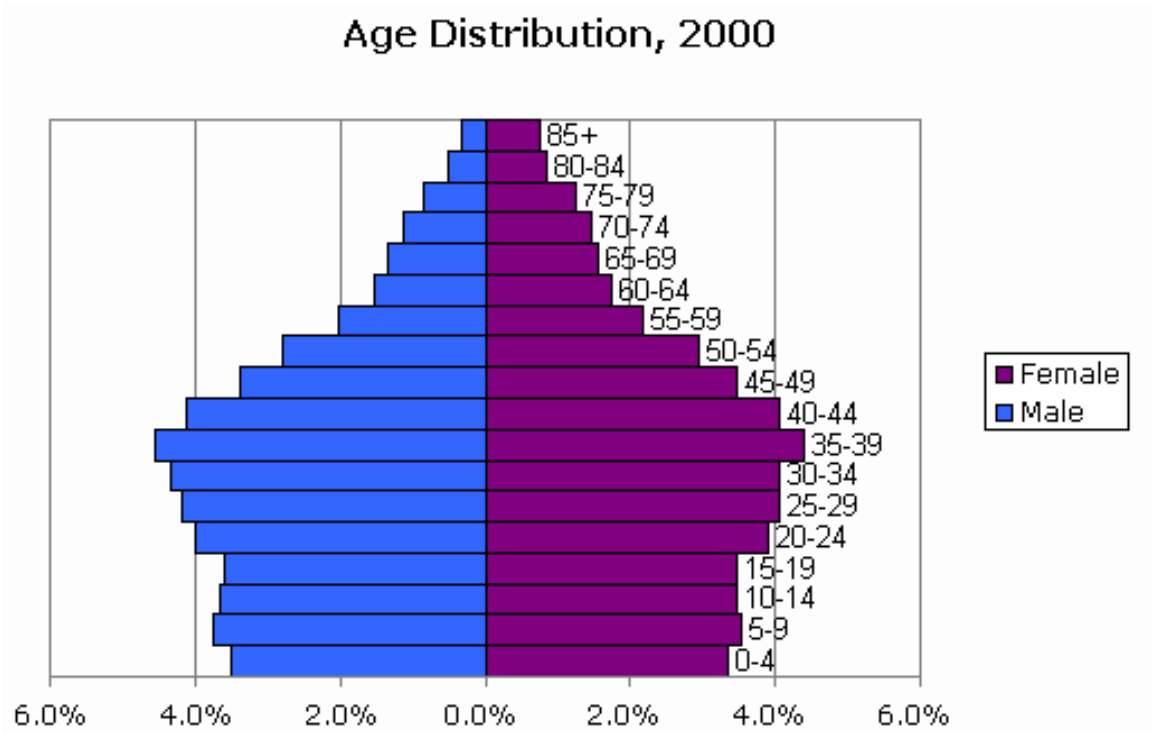


Figure 2.2A Age Distribution Census Chart Orange County Florida (public Government domain data)

As can be seen from Figure 2.2A, a good portion of the audience is from age 15-39. These are ideal marks to hit since many parents want to get something their teens can enjoy or even themselves. Within our county area there is roughly 200,000 individuals who fit our criteria making. The full numbers for the above graph can be seen in Figure 2.2B.

The customer will have to be warned of the health risks involved in using the Oculus Rift. Large, elaborate vocabulary is used in disclaimers that simply come down to risk of nausea in those not used to using the Oculus Rift and risk of seizure in those with epilepsy. Nonetheless, these are important to ensure that the user can have a safe and enjoyable experience. It is important the the customer understands how to properly use our devices, and proper detailed instructions will be included in our user manuals once our product is complete and well documented.

Age Distribution by Sex, 2000

	Male		Female	
	Number	Percent	Number	Percent
Total Population	443,716	49.50%	452,628	50.50%
0-4	31,444	3.51%	29,931	3.34%
5-9	33,477	3.73%	31,764	3.54%
10-14	32,690	3.65%	30,982	3.46%
15-19	32,108	3.58%	31,234	3.48%
20-24	35,810	4.00%	34,953	3.90%
25-29	37,543	4.19%	36,375	4.06%
30-34	38,798	4.33%	36,339	4.05%
35-39	40,870	4.56%	39,405	4.40%
40-44	36,869	4.11%	36,477	4.07%
45-49	30,169	3.37%	31,048	3.46%
50-54	25,010	2.79%	26,286	2.93%
55-59	17,982	2.01%	19,431	2.17%
60-64	13,757	1.53%	15,633	1.74%
65-69	12,022	1.34%	14,042	1.57%
70-74	10,063	1.12%	13,242	1.48%
75-79	7,616	0.85%	11,194	1.25%
80-84	4,663	0.52%	7,474	0.83%
85+	2,825	0.32%	6,818	0.76%

Figure 2.2B Age Distribution Numbers for Figure 2.2a

The above table displays the figures for the census of Orange county region in the year 2000. Age range of 15-39 makes up 20.66% of the 49.5% male audience and 19.89% of the 50.5% female audience. This is 15 years old for a census chart but the numbers for this age bracket have not fallen far from the mark as the 15 years would leave most percentages in the area of 3.5% - 4.5% still factoring in about 100,000 individuals from both genders.

The below sections list out the needed portions for the customer's use and enjoyment of the project devices. The better performance available the better the entire experience. Though the customer must also be aware of the potential hazards of using the equipment involved. The times for use and directions will be explained later. Main focus will include the major components and the environment they are meant to be used in.

Quadcopter

- Quadcopter should be in a controlled environment. No other users in the testing area.
- Flight time should be anywhere from 2-5 minutes.
- Full free range movement within the respective controlled environment.
- Fail safes for loss of control of the Quadcopter.

Glove Controller

- Glove size should be a one size fits all or adjustable glove.
- About a cubic foot of room should be given for where the user will be moving their hand while controlling the Quadcopter.
- Buttons on the Glove and functions designed should be clearly labeled.
- No pieces on the Glove should be loose or detached while in motion.

Oculus Rift

- Headset should be configured for each user by utilizing the configuration software before beginning a virtual reality experience. Discomfort may be experienced if not done.
- Children under 13 should not use this device unless monitored by an adult.
- Safe environment: The headset limits vision and as such the customer must be aware of their surroundings.
- Seizures: Any individual who has experienced a seizure, loss of awareness or other symptoms linked to epilepsy should consider first visiting a doctor before using this device.

Environmental

- System should ideally be run in dry indoor environment
- Withstand indoor household temperatures ranging from 60-80F
- Avoid rainy environments when piloting craft

2.2.1 Performance

The customer will need the controls of the quadcopter to be highly responsive. The video stream will need to be as close to instantaneous as possible. Since this is a reality-based virtual reality performance of a modern-day dog fight (that is, it is a fight between two aircrafts to see who is more maneuverable and whose responses are better) then the customers will need their equipment to be as responsive as if they were in a cockpit. This poses some issues as due to the nature of the system, with the user wearing the Oculus Rift headset their visibility is limited to only that of the copters onboard cameras. The controller should be able to be used with minimal coordination or external inputs. Any delays in the video feed need to also be minimized to ensure that the user does not experience crashes or delayed inputs in their flight.

Task	Expected Time	Ideal Time
Video Feed delay	.01ms	.001ms
Controller signal response (X direction)	0.001ms	0.0001ms
Controller signal response (Y direction)	0.001ms	0.0001ms
Controller signal response (Z direction)	0.001ms	0.0001ms
Controller gyroscope response	0.001ms	0.0001ms
Correction accuracy (in X direction)	0.001ms	0.0001ms
Correction accuracy (in Y direction)	0.001ms	0.0001ms
Optional Laser trigger time	0.001ms	0.0001ms

Table 2.2C Ideal time chart for performances.

The above table is based off of our collective knowledge as it stands with the relating technology. The more we experiment and research with our pieces of equipment, the more we will know on how the equipment reacts. This chart, we expect, will be edited upon numerous times throughout the project.

2.3 Realistic Design Constraints

Below are listed the related Realistic Design Constraints pertaining to the main functions of our project. Functions within the constraints include:

- Wifi and remote control
- First Person View systems
- Cameras
- Unmanned Aerial Devices

The following sections will define the ABET Constraints and ANSI required standards. These will be the listed constraints that the project must abide by to be considered both safe and legal to operate with

2.3.1 Economic Constraints

We will have an economic constraint of a total of $1200. Each of us will contribute $300 of our own money to be pooled together in attempts to create this project. Already, about $300 has been used for buying the Oculus Rift. This gives us a remaining $900 to work with. Besides this, we have a working Quadcopter that will hopefully be able to meet the criteria we have set before it. Though researched, the quadcopter has been used before and as such, could be damaged beyond what we see.

Economic Constraints Summary
1. Keep total costs under $1200
2. Oculus Rift and Quadcopter are most expensive pieces.
3. WiFi Adapter and Raspberry Pis will be second largest expense.
4. Glove components will be the most minor expense.
5. End-goal will be to have cheap, straightforward replacement parts.

Table 2.3A Summary of Economic Constraints.

Optimally, we will not be needing nearly this amount of money. The two large purchases are the Oculus Rift and the Quadcopter. The glove with the microcontroller, accelerometer, gyroscope, and attached host machine will be a minor expense. Then the WiFi adapter and Raspberry Pi's will be more than the glove's components, but less than that of the Oculus or Quadcopter.

The hope of the project, given an ample amount of time, is for us to create a new version of laser tag that would use out virtual reality quadcopters. So it would be optimal that our project would become a product to establishments that have arcade components. This would be the perfect controlled environment while bringing in profits for our establishment customers. In such a setting, we want our product to be as cheap as possible.

The cheaper we can make the product, the easier it is to replace. This would appeal to both us, the ones making and testing the product, and the company, the ones buying and paying for maintenance of the product. Instead of trying to convince a company to buy an extremely large and expensive piece of equipment, we would be selling an extremely portable, relatively inexpensive, product that needs just a small, windowed room.

2.3.2 Environmental, Social, and Political Constraints

As stated before if the eventual hopes of finished production, the quadcopter will be in an enclosed room hosted by the establishment consumer. Thus, in terms of noise pollution, the finished product will be entirely contained and without consequence. The quadcopter,

glove, and host machine will all work off of electricity, so there will be no worries of environmental pollution, either.

Environmental, Social, and Political Constraints Summary
1. Enclosed room will reduce noise pollution.
2. Electricity power will add no environmental pollution
3. Plastic pieces and available sources make disposal of parts environmentally friendly.
4. Single-handed glove will make it more accessible to those with one working hand.
5. Lack for need of movement makes it more accessible to disabled personnel
6. Monetary constraints will depend on establishment holding said technology.
7. The entire project will have no racial or sexual bias.
8. Companies like Arduino will be compensated for their programming environment if this product is, indeed, sold.
9. Product is not meant to encourage violence.
10. Product is not to be used for spying.
11. Employees will be given safety measures to follow.

Table 2.3B Summary of Environmental, Social, and Political Constraints.

Similarly, there will be no water or landscape pollution coming directly from the finished product. If a propeller breaks, it is a lifeless, harmless piece of plastic that is easily recycled. The battery is the closest aspect to environmental pollution, but as battery technology advances, as long as the voltage and current supply are the same, more environmentally safe batteries can easily be used with our product.

In terms of social constraints, the use of the single glove as a controller will hopefully make it more accessible to people than a two-handed controller. The use of one hand will make it accessible to amputees or other such people with a single working hand. Similarly, since the user only has to use their hand and not their legs, paraplegics can also use our product. In terms of monetary availability, that will depend on the establishment and how much it will cost for the a single round of the hopefully achieved laser tag. We are hoping for this to be as accessible as possible. Further in this paper, quadcopter regulations will be mentioned that the consumer must adhere to.

As for political constraints, we are so far self funded and will probably continue to be so. We will be using Linux to program the Raspberry Pis. Linux is an open source operating system that is for the public use. We will be using the Arduino environment to program the microcontroller that comes from the Arduino Uno. This we will have to come to an

agreement with the Italian company on splitting development costs fi it over comes to the point that we actually have a product to sell and are willing to do so.

Because this is a virtual reality game that depends solely off of shooting an inanimate object, there will be no inherent negative aspects of discrimination against race or gender. Along these lines, it is to be advertized as a purely fictional game with no physical consequences. Damage to quadcopters will more than likely be caused from user error than actual violence as the user may bump their quadcopter in such a way that it accidentally crashes.

On a larger scale, this product is for strict use in contained environments away from public eye. The quadcopter is not designed for use of spying or disturbing others. violation of such parameters will be at the hand of the consumer and out of our reach as producers of material. The product being in an enclosed space will be a large protector from physical harm that may be caused by a crashing quadcopter. The limited battery will allow for the establishment housing the product to safely be able to acquire and fix the quadcopters in such a manner that no physical harm can possibly become of them given that they take the correct safety measures. To expand upon this, the battery of the quadcopter can only last so long. So if there is a malfunctioning quadcopter and an employee needs to acquire it, then the employee can wait the respective time until there is no possible energy left in the battery. As a precautionary measure, the employee can also turn off all RC equipment relating to the quadcopters so that none can accidentally cause a quadcopter to be active when the employee is attempting to handle said quadcopter.

2.3.3 Health and Safety Constraints

As stated previously, minor measures can be taken to create a large effect. In the situation of a malfunctioning quadcopter, the employee can do a number of things before retrieving the quadcopter from its enclosed space.

Health and Safety Constraints Summary
1. Reiteration: Employees will be given safety measures to follow.
2. Enclosed space will provide employee and public safety.
3. Enclosed space will provide lack of noise pollution.
4. Heat from batteries must be handled carefully.
5. No toxic or radioactive materials.
6. Oculus rift will have extensive personal health safety measures.

Table 2.3C Summary of Health and Safety Constraints.

Firstly, the employee can have all of the equipment that may be giving the quadcopter its commands turned off. Secondly, the employee can wait an allotted amount of time until the battery in the quadcopter is sufficiently drained.

The employee must be careful when handling the battery. The batteries inside the quadcopter can become very hot when used excessively. As such, the employee can wait another allotted amount of time from when the quadcopter has finished being used to when the battery should be sufficiently cooled. Besides when the quadcopter is in use, the battery can become heated when being charged. As such, the employee can use a cloth to dampen the heat between the battery and they employee's hand. These instructions will be expressed to the purchasing establishment and it will be their duty to inform their employees.

The fact that the quadcopter is to be in a contained, windowed room means that there should be no possibility for the safety of the public to be encroached upon. Similarly, the materials of the walls of the rooms will help in dampening the sound coming from the quadcopters. So fear of sound pollution should be deterred.

There are no products of radioactive nature. There are no products that have form of toxicity when handled correctly. The only possibility of toxicity is if an employee uses tools to physically crack open a battery, which is highly unlikely.

The only cause for safety concern is the extended use time of the Oculus Rift. When used for a long amount of time, the Oculus Rift can cause nausea. This is easily subsided by not using the Oculus Rift and having the customer take deep breaths and possibly consume food that calms the stomach. Thus, very blatant warnings must be executed for those susceptible to nausea. Additionally, a seizure warning must be displayed for those with epilepsy or sensitivity to bright, flashing lights.

2.3.4 Manufacturability and Sustainability Constraints

The end result of this project is to have complete instructions on how to make this product with relatively no thought. There will be layouts for making a printed circuit board where to replace it, the customer must only give the instructions to someone with experience in creating printed circuit boards. Additionally, the parts for the printed circuit board will cost very little. We will give sources for the parts that can be replaced on the quadcopter. Past the parts given, a new quadcopter will have to be purchased and the quadcopter model will be provided. Additionally, the Oculus Rift is name branded, so the customer should have no problem contacting the Oculus Rift company if we cannot help them.

The quadcopters, host machine, and glove will all work off of electricity. With this design, the entire system will be entirely sustainable given that the establishment customer that purchases the product has access to an electrical outlet and can recharge the batteries given.

2.4 Deliverables

Amongst the constraints and specifications filling this paper, we need to lay out exactly what we want to do with this project. Here, we will do that. Below is a bullet point summary detailing what we want to get done. We are aware that each of us is very busy in our day-to-day lives and that time is of the essence. Thus, we have separated our deliverables into two sections: one for what we must get done and one for what we would like to get done.

Deliverables determined for this project are:

- Working Quadcopter system/drone
- Working glove controller for Host machine or Raspberry Pi
- Glove successfully controls Quadcopter
- Oculus rift programmed for use with camera to the drone
- All documents required with the associated parts and components, such as Oculus Rift user manual. Any documents to accompany will either be added directly or subdirectly and associated at project completion

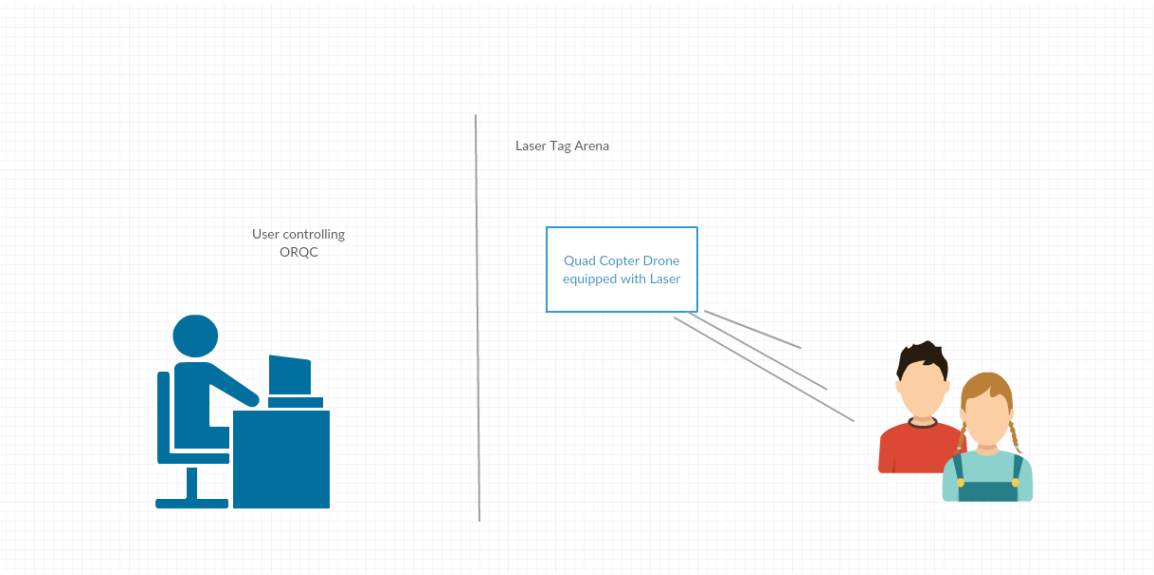
Now that we have laid out what we must get done, below is what we would like to get done. This is a very ambitious project to us since it will be a first time for a lot of us in working in so many different environments. For example, one group member has never worked with a Raspberry Pi and as such will have to familiarize himself in order to become useful to the overall group. As such, the following are things that we aspire to complete given that the metaphorical planets align.

Optional further Deliverables should time permit:

- Laser calibrated for potential laser tag use
- Additional buttons calibrated to the controller for other uses
- Complete independence of the Controller as a stand alone USB plugin for external use
- Ambidexterity to glove components

Laser Tag system : There are a couple of issues that would need to be addressed and design decisions should we decide to implement this. Depending on how acrobatic our copter ends up being in the final draft would affect some of the major decisions regarding implementing some laser tag. Because we already plan to overlay a game engine to keep track of possible readings from the glove, further customizing that and adding crosshairs and “score” counts might be an easy addition. We can't be sure until we actually begin testing however. We will be looking into similar DIY laser tag projects and seeing what we can learn from them. Realistically if we want Copter v.s Copter laser tag to work, that might be beyond the scope of our project to setup two copters and have them fight in unison, but we would at least like to try to get a single laser on our copter (Or perhaps implement a shotgun type spread to increase accuracy, and make up for any difficulties caused by attempting to aim a laser mounted onto the Copter). If we can setup a single stationary laser we could possible move onto attempting to synchronize some servos and get a wider range of movement for the

laser. If we can get mounted lasers to be able to target an average sized person wearing some sort of Laser Tag vest, our product could be marketed as a way for disabled kids.



2.4A : User piloting ORQC in laser tag arena

Kids who would normally not be able to enter the laser tag arenas would then be able to take part and enjoy the experience playing with other children. This of course would not be limited entirely to children, but as they are a large proportion of the laser tag community, we would focus on marketing towards them. We feel this gives a nice spin on our product while also ultimately accomplishing a good goal. Virtual Reality does not have to only be for the average persons enjoyment, but can help closer bridge that gap for people who may not have had the chance to enjoy things we take for granted.

2.5 Estimated Budgets and Finance Plans

Group members understood at the conception of our project, that we would likely run into some large costs. After discussing with Dr.Richie we were warned to expect a minimum of a $1000 budget, that could go upwards to $1500 when taking into account for replacements and prototypes. Each group member has an income, and found this estimate to be reasonable.In the early planning stages most of the group was unfamiliar with the average price of a majority of the components involved in the project, as such its possible we are underestimating the costs. The cost towards the consumer would likely be much smaller than the development costs as we get a better grasp of our system and where we can cut costs and be more efficient in our usage components. Because of the nature of the implementation of Oculus and copter, user costs would likely be mainly centered around the Hand Controller, and it would be our goal to ensure compatibility with a variety of copter setups, assuming the user has access to an Oculus Rift. Once more research has been done, a more detailed budget plan will be outlined.

Due to the high cost of this project, the group will be seeking out alternate methods of financing. Ideally we hope to seek out some sponsors that may be interested in the concept

and could possibly rent an Oculus for us to use in development for the duration of Senior Design I & II. Group member Gunnar also expressed possibilities with working together with his company who could be willing to assist in funding. Group Member Gustavo also noted that his father showed interest in the novel idea and would like to see it evolve and saw possible applications in his field of work (Construction), for safe image capturing of construction sites.

We will outline a more detailed budget in a later section once we get a better idea of how much funds we have, as well as how any possible sponsorships could affect our total budget. All costs incurred will be written down as they are made to ensure we have an accurate log of purchases , and ensure that we are always aware of how much funds we have left to allocate where necessary. The table below (2.5a) shows a planned budget distribution of each individual component that will be used.

Parts	**Prices**
Copter Components (Propellers, Hull, Motors)	$100-200+
Flight Controller	$50-100+
Sensors	$20+
Oculus Rift	$300+
Battery	$25-50
RC transmitter/receiver	$50+
MicroControllers	$40+
Raspberry Pi	$60-90+
Camera	$25-50+
Wifi Transmitter/Receiver	$25-50+
Wifi Range Extender	$50+
Integrated Circuit Components	$10-20+
PCB	$10-30+
Total	**Approximately $ 1000**

Table 2.5A : Predicted costs

2.6 Scheduling Concerns and Time Limitations

Due to starting in the Summer semester there are a few considerations we had to make. We have limited time, so it must not be wasted on pursuing alternatives that don't hold strong enough prospects for our project. We have to be precise and efficient to ensure goals are met on consistent schedules. As such we find it important that we have clear goals in mind and establish some estimates of how we wish to divide our work and the time we wish to allot to it all. All research moving forward as well as our planning and scheduling will be done with a few key points in mind.

- Budget constraints must be kept in mind, however speed is top priority to make sure we maximize our time.
- Documentation of every piece of research done by group members should be organized and consistent.
- Plan to meet minimum twice a week on Senior Design 1 class days to discuss new ideas or concerns.
- Weekly Weekend meetings over skype or similar VOIP software to stay up to date and review what has been accomplished and needs to be done.
- Aim to complete report with time to spare to ensure we have time to look over and correct any tiny details.
- Focus early efforts into studying and understanding techniques that we will be using throughout our project to be ready to dive into work for the upcoming fall semester.

The above key points will help place some focus on group efforts. Consistent communication is key to ensure milestones and goals are met in a timely manner. To best make use of our limited time for planning and inexperience with some of the topics we will be discussing , any additional time spent on learning how to use particular software or how to interact with hardware, as well as early acquisition of key elements like the Oculus Rift and Quad Copter Components will be essential to a detailed design report in Senior Design I, ensuring a smooth development process in Senior Design II.

The start of this coming September will indicate that we have begun Senior Design II and have started working on the final portion of the project. From September to December, we are planning to work on the final documentation that will act as our recordings and findings during the executive stage of this project. Spanning the months of September to November, we will be prototyping the quadcopter creation and the glove creation. We will also be testing the softwares created for the glove, the host machine, and the quadcopter all the while recording our findings in our final paper.

The side objectives, though no less important from the overhead objective, during September will be to acquire the parts necessary to create each stage of the project. We will preferably purchase spares of each item to make sure we will not lose time if we have to wait for another shipment. We will create a prototype of the glove that will at least give readouts to a standard laptop computer with the Arduino environment on it to test and make sure that the gyroscope and accelerometer are working. Similarly, we will be creating and testing the Oculus Rift programming that will be done. We will want to create the

quadcopter with its additional Raspberry Pi, camera, and WiFi accessory. We will also be verifying that our initial documents, this paper, are correct and that we have no further research to be done.

During October, testing and writing of the Oculus Rift program will continue. At this point, it is hoped that the quadcopter and glove prototypes will be ready for each other so that testing the communication between the two can begin. Once this has some steady ground, we will be testing the quadcopter, glove, and Oculus Rift together.
Once November comes around, we want to be on more solid ground with the quadcopter, Oculus Rift, and glove communicating together. As such, we plan that November will be a time of final touches to make sure everything works smoothly. Time permitting, this may be when we add supplemental material. Running simultaneous to this, we will be creating our final draft of our documentation papers. And by December, everything will be ready for presentation.

3. Standards

Standards and regulations are required by law to be instilled within the design process as to prevent any huge irregularities in construction or design. Without these rules, users may be at risk or other laws may be broken without prior knowledge. These standards also provide guidelines and limitations for our designs.The standards and regulations herewithin are for the safety and use of the customer and the designers to ensure all requirements can be met without any violations to current laws.

3.1 Standards

Because a quadcopter is considered a small Unmanned Aircraft System(sUAS), it must adhere to the Federal Aviation Administration (FAA) standards. The FAA has worked closely with the RTCA to make the available for free online DO-344 Volume 2-Appendices F & G - Operational and Functional Requirements and Safety Objectives for Unmanned Aircraft System Standards. Below is a table given by the Radio Technical Commision of Aeronautics (RTCA) for Unmanned Aircraft Systems Appendix F.

All of this applies to Unmanned Aircraft Systems of classes A through E and then G. Our quadcopter, being below 20 pounds, is considered to be in class A. There were many more categories in the Appendix F of RTCA's Unmanned Aircraft Systems standards, but they did not adequately apply to our situation and so we deemed them to be superfluous and unnecessary information. What has been given here called to us to be necessary enough to give said information and explain the data with reasoning as to why we will be adhering with the regulations put upon UAS. Besides, of course, adhering to said regulations for obvious physical and social safety measures.

OR ID		OR-UAS.1
UAS MASPS Requirements	UAS Unique Operational Requirements	UAS shall comply with National Airspace System (NAS) operational rules, regulations, guidance, and procedures used for manned aircraft, according to its type certification, unless granted a specific exception by the governing authority.
References		14 CFR § 23 14 CFR § 25 14 CFR § 91 14 CFR § 121 14 CFR § 135 DO-304 DO-320
Same as Manned		
IFR		X
VFR Communicating w/ATC		X
VFR Not Communicating w/ATC		X
Controlled Airport		X
Uncontrolled Airport		X
Off Airport		X

Table 3.1A. Standards given by RTCA for determining regulations.

The table has a lot of wordy language that means that when a UAS falls under a category with an X tagged in it, the aircraft must adhere to the regulations that will follow this section. Because our aircraft definitely falls under the "Class A" category and "Off Airport" category while not being in the "Same as manned" category, we must adhere to the regulations. The regulations, as one will see, are very understandable and will be easily followed given the setting of being in an enclosed, windowed room at the final product.

Moving on to the Virtual Reality standards, we are expecting our Virtual Reality to feel almost instantaneous and fluid. As such, we want a time delay in video feed to be no more

than 0.01ms between when the video is recorded and sent to when it is received and displayed. We will want similar times to go along with virtual reality movement reaction. We want 0.01ms or less for the time between one moves their head and when the Oculus Rift reads this and moves the screen's position to view the related portion of the picture.

As stated earlier, the Oculus Rift has the possibility to cause nausea in some users given a set amount of time unique to said user. Thus, through testing on each of us and if possible through people who accept our invitation to test, we will determine a reasonable time for an average user to use the Oculus Rift without meeting nausea.

The specific parts for the quadcopter will be held to the standard of lasting at least 100 non-collisional flights. This number based off of personal use and experience. The normal wear and tear of flying may cause structural damage to the propellers which may have to be replaced. Given no collisions at all, the on board Raspberry Pi and its camera and WiFi adapter should last for as many years as any given modern laptop. Similar standards will be held for the glove, host machine, and Oculus Rift. SInce they will be largely computer-based pieces of equipment, their lasting value should match the commercially available personal computers. It cannot be stressed enough, though, that these standards are held under the circumstances that the equipment is well looked after.

3.2 Regulations

Here are the highlighted safety guidelines that we will follow as provided by the Federal Aviation Administration's educational campaign "Know Before You Fly":

- Follow community-based safety guidelines, as developed by organizations such as the Academy of Model Aeronautics (AMA).
- Fly no higher than 400 feet and remain below any surrounding obstacles when possible.
- Keep your sUAS in eyesight at all times, and use an observer to assist if needed.
- Remain well clear of and do not interfere with manned aircraft operations, and you must see and avoid other aircraft and obstacles at all times.
- Do not intentionally fly over unprotected persons or moving vehicles, and remain at least 25 feet away from individuals and vulnerable property.
- Contact the airport or control tower before flying within five miles of an airport.
- Do not fly in adverse weather conditions such as in high winds or reduced visibility.
- Do not fly under the influence of alcohol or drugs.
- Ensure the operating environment is safe and that the operator is competent and proficient in the operation of the sUAS.
- Do not fly near or over sensitive infrastructure or property such as power stations, water treatment facilities, correctional facilities, heavily traveled roadways, government facilities, etc.
- Check and follow all local laws and ordinances before flying over private property.
- Do not conduct surveillance or photograph persons in areas where there is an expectation of privacy without the individual's permission

Due to the settings that we are proposing, some of these are inconsequential, but they still need to be recognized. Since we are proposing the drones to be housed in a room with glass walls, or at least large windows, then we do not need to worry about losing sight of them. Being in a building will ensure the drones do not exceed 400 feet above ground and the room will provide the 25 foot barrier between persons operating or observing. Being inside will also disregard the environmental conditions of weather.

4. System Design

This project contains a complicated system with many components that must communicate with each other and function smoothly while still leaving room for future improvements and features that we discover and may wish to include through the development process. We have to find a way to have all components of the project work together. We need the quadcopter to work with the WiFi adapter; the WiFi adapter to work with the host machine; the PCB glove to work with the quadcopter. This is a grossly oversimplified cycle that needs to work in our project. However, at its core, our system will consist of mostly "stand-alone" modules (The oculus, the host, and the glove) that can function on their own. We want to make sure the copter can handle staying in the air should it lose signal from our Hand controller, and that the Host machine has fail safes should it lose connection with our video stream and ensure a speedy reconnection. Having them be able to function semi autonomously will ensure that we avoid some minor issues like our copter crashing often during testing and such and make our lives much easier. It is our task to ensure that the Rift ,Copter, and Controller can function together and consistently. Below is a visual representation of the quadcopter communicating with the Raspberry Pi which communicates with the microcontroller on the Arduino.

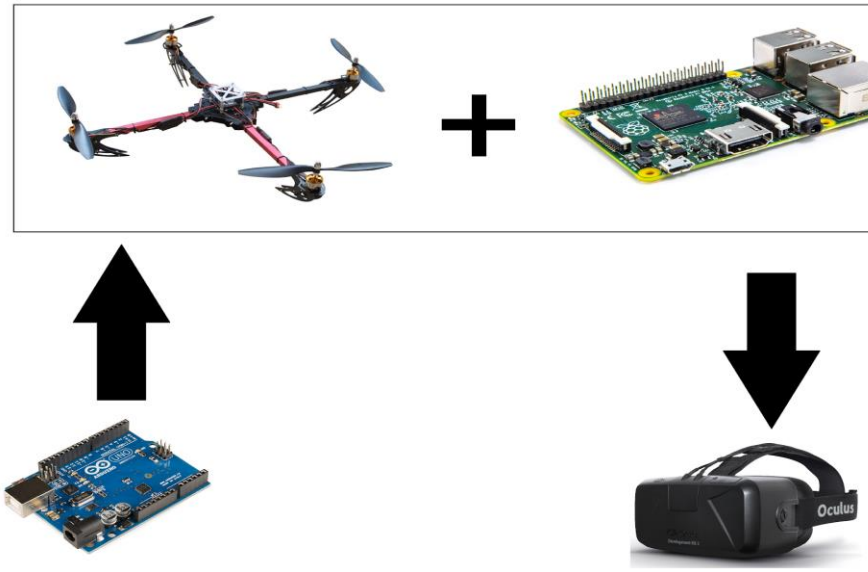


Figure 4A. Quadcopter communicates with Raspberry Pi in addition to the Arduino.

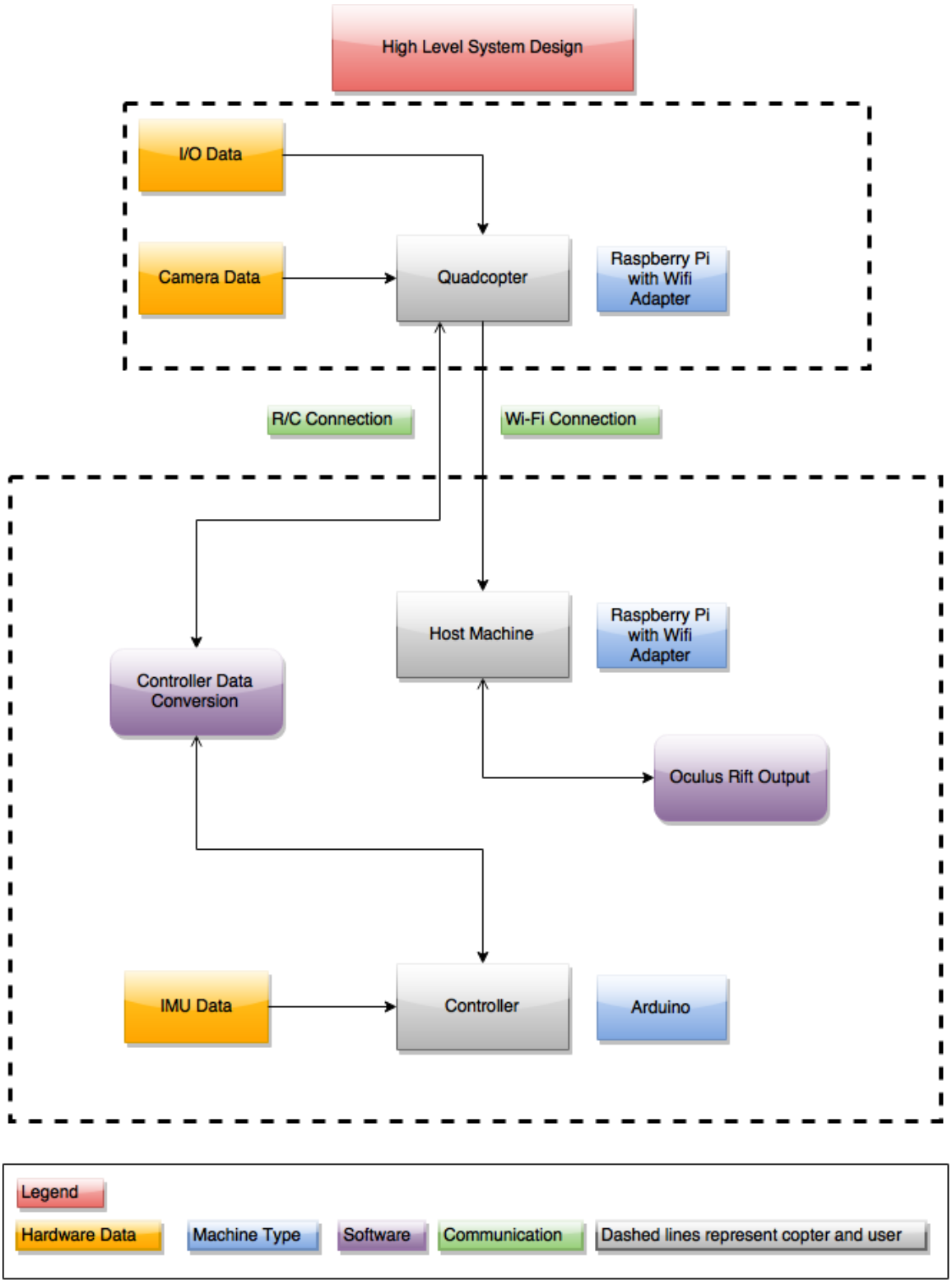


Figure 4B High Level System Design

On a more detailed level, the individual components of our system have to be chosen carefully to ensure that our implementations do not conflict with each other.Intensive research will be done towards ensuring each component we decide on, carries no risk of incompatibility with the rest of the design. Priority will be given to compatibility and ease of use for each component, to help facilitate early prototyping and testing. There are a wide range of methods and techniques to accomplish our goals throughout this project, whether it be the components of the copter (The flight controller, the motor, the weight etc) the glove (the model of accelerometer , the circuit board, etc) or the software we will be utilizing in the integration of the oculus and video feed. Research done by group members on key components and details on the pro's and con's of possible solutions will be detailed below.

4.1 Key Design Elements and Associated Research

As we are combining 4 separate elements together in one project, it is crucial to check for other projects that have done similar work. Along with these projects we need to look over specific elements like batteries, different brands of Quadcopters, flight controllers and cameras. A concentrated effort must be made to ensure that proper research is completed. Due to the complexity of our project, there are various little details that could be overlooked, and cause issues at a later time when we least expect it. During our research there are a few details regarding the components we use that we are looking for in particular.

We want to weigh our decisions with our limited funding and time constraints in mind. As we have a smaller time span in the summer semester to dedicate to research and acquiring components to get hands on experience, care must be taken to ensure any possible choices we make be affordable and accessible. Most of our early prototyping will likely be done in the early stages of the upcoming fall semester, to make better use of the longer development time period. As such most of our research will be towards affordable products and alternatives , as well as researching similar projects to gain a better understanding of how to best proceed in the following months. Future drafts will include more detailed information and concrete decisions as we begin testing our choices and making clear decisions.

4.1.1 Related Projects

High 6 - The basis of this project was to integrate bluetooth from an android and the androids language app to translate signs that were made from a glove with sensors in it. As a person would sign language out signs and words the android app would translate it for the receiver. This was meant to bridge the gap for users that don't know sign language. High 6's relation to our project lies in the use of bluetooth to a wireless system and receiving signals from the glove. In the event we decide to make the glove wireless we can utilize similar methodology to bluetooth connection to the host machine.

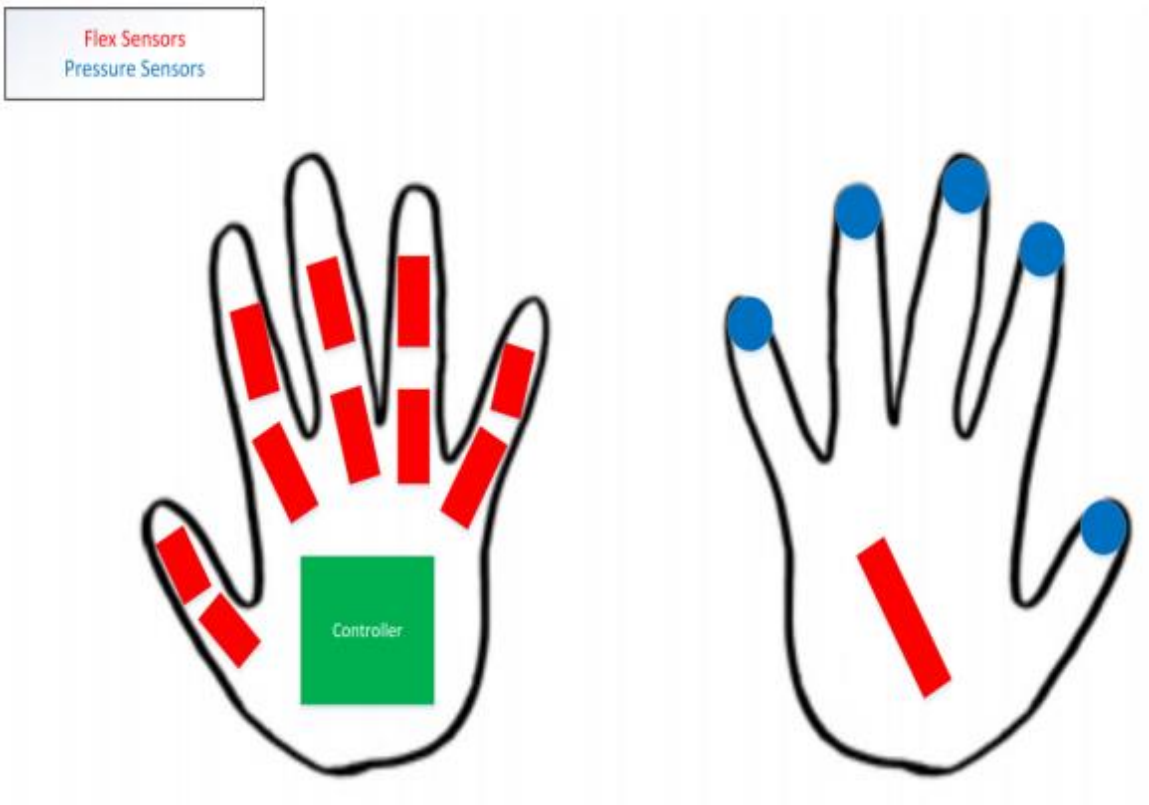


Figure 4.1A Enable Talk High 6 Project glove design plan Permission granted

The main difference in their project to ours for the glove portion is that we will be focusing on the controller aspect of the glove. We are not in need of single digit use though the group has questioned that potential before as we could measure the changes in finger positioning to allow for additional commands to be sent to the controller. The High 6 team's use for the components on the glove does allow for additional functionality to the project but again goes outside the full scope of what our glove will provide.

DroneNet: The Quad Chronicles - This project utilizes battery efficiency with UAV devices. They created a carrier style platform for the UAV to charge its battery from allowing for extended period of mobilization of the UAV. A few key pieces we can pull from this project are their recommendations on camera use for the Quadcopter, battery power choice, and potential methods for resupplying the battery on the go. The carrier drone platform used a home built recharging platform known as the Bedini motor.

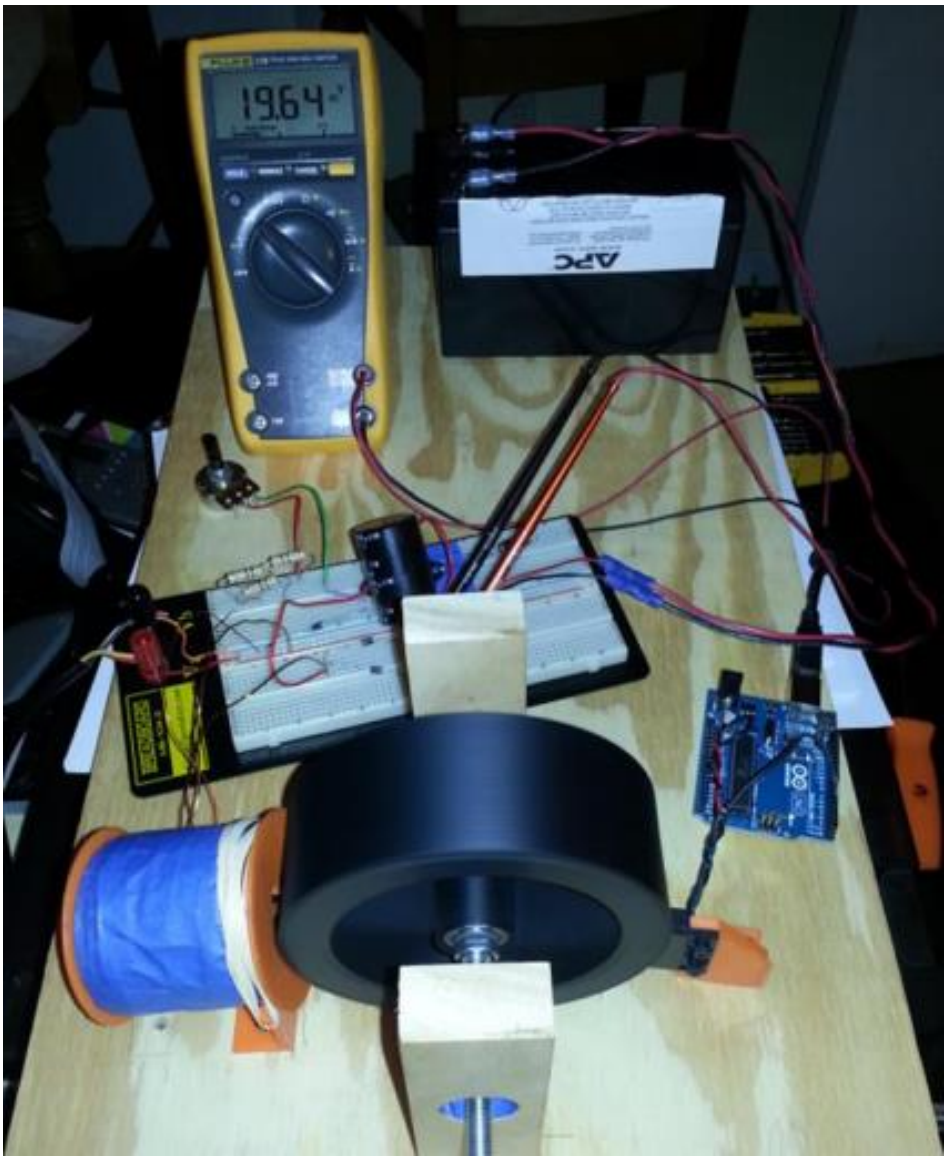


Figure 4.1B: Bedini motor development DroneNet. Permission granted for use of picture

The Motor system displayed above is only one of the many components this team used in completing their project. Another major choice they underwent was deciding what camera system to use with the Quadcopter. Decisions were met with either the GoProHero3 or the Raspberry Pi camera. The Raspberry Pi camera was a solid choice among the the two but with additional funding granted the team decided the GoPro may prove more valuable due to its on personal battery, which could outlast even longer extended usage that the platform would provide.

Live Two-Camera Video Stream from Raspberry Pi to Oculus Rift through GStreamer - Torkel Danielsson (Permissions granted for mention)

The basis of this project was a home live-streaming experience. We stumbled upon this project in the early planning stages as we were researching possible methods to stream

compatible video to an Oculus Rift. This project confirmed our expectations that the Raspberry Pi would be a suitable platform from which to stream video. The Video Stream setup is quite simple, it consisted of a single Raspberry Pi : Compute module, with two Raspberry Pi cameras interfaced to it. Both cameras had a wide angle lens module attached, and were positioned to provide an 180x120 degree FOV with minimal delay over an IP connection. All video encoding was capable of being done on the fly within the Pi itself before being sent to his home computer to be then displayed on the Oculus. This project utilized a VideoCore GPU to provide the power necessary to do the live encoding, which we will be looking into as a possible component of our hst machine. The Raspivid function from the Raspbian OS pipes video feed into Gstreamer which is sent over an RTP protocol to a PC, this setup is almost exactly what we envisioned and will be pursuing further contact with Mr.Torkel for possible assistance and tips regarding how we may utilize similar techniques. His full setup consisted of 4 instances to properly encode this large field of view, so downscaling his project techniques should be quite feasible for what we wish to accomplish. Further contact will be pursued to get a better grasp on how he managed his

Through our early glances at Laser Tag Systems we came upon this project that seemed to fit our interests. A full part list as well as build instructions detailed a quick buzzer system using cheap breadboard components (under $20 total build price) to simulate a target using photoresistors. Being built primarily on a breadboard, with which we all have experience with this is an easy to make little bundle that any member of the group could put together when we want to begin testing lasers. It works with a generic store bought laser pointer so we don't have to worry about looking for fancy parts. There is no programming required either, so of course we would have to pursue better alternatives once we begin to implement a scoring system and such, but this cheap build will let us attach it onto someone's shirt and start getting some target practice with any early prototypes of our laser tag system, and maybe we could even expand upon this by connecting a microcontroller such as a BASIC Stamp 2 board, or an arduino if we want even more functionality.

4.1.2 Batteries

There are quite a few limitations that we must overcome. To start, we have to keep in mind the battery limitation of quadcopters. In an ideal world, quadcopter batteries would last for hours at the least. Realistically, however, an average time for quadcopter flight is about seven minutes. What would be considered a very large amount of flight time is doubling this to fourteen minutes. As well as just powering the copter itself, we have to worry about the batteries being used to power the Raspberry Pi and camera bundles attached to the copter.

So we are going to look at what is available to us and decide which way we need to go. First, we will look at the batteries we have found that would suit the HPQ1 quadcopter that we currently have available. We are going to look at the standard battery that comes with the quadcopter, the Turnigy 4S LiPo Pack, and the Multistar High Capacity 3S Multi-Rotor LiPo pack.

The standard battery that comes with the HPQ1 have a voltage of 3.2V. So when researching other batteries, we are going to want to keep within this range and not deviate too extremely. It has an ampere hour of 2200mAh while boasting an average of 14 minutes on the official site. Its discharge rate is 20C and the entire battery is 180 grams. Costing $69.95, this battery is our lightest option with the lowest ampere hours. We are hoping that this will be enough to power everything we need and keep the quadcopter in the air high enough and long enough. Testing will tell us whether or not we need to switch.

The Turnigy 4S LiPo pack caught our eye with its whopping 5000mAh that would spell twice the amount of flight time as the standard battery. Conversely, its weight is also over twice that of the standard at 556 grams. But depending on the resulting weight of the Raspberry Pi, camera, and WiFi adapter, this may be completely in our range of weight. I say resulting weight in the foresight that one of the chosen components for the video streaming may not be powerful enough and a heavier, more capable part may be chosen. The Turnigy has a discharge rate of 30C with 3.7V per each of the four battery cells. The Turnigy's retail price of $37.39 makes it much more affordable than the standard HPQ1 battery, but its weight deters us and makes us continue looking for better possible options.

Coming to the Multistar High Capacity 3S LiPo pack, it also has 3.7V per cell. With 4000mAh, it is still significantly more than the standard, and with only 244 grams it is less than half the weight of the Turnigy. Multistar has similar discharge rate as the standard at 20C. And at $19.96 , it is our cheapest option. Coming down to price, weight, and ampere hours, the Multistar is our most viable option given that the standard battery does not complete the job needed.

In the event that we need to use a different quadcopter, we also need to analyze the available batteries for it. Thus, the following is an analysis of the batteries we have found available for the Traxxas 6608. Similarly with the HPQ1, we will start with the standard battery that comes with the Traxxas, then move on to the ZOP LiPo battery and then the Syma X5C-1 X5C X5A.

The standard battery that comes with the Traxxas 6608 has a voltage of 3.7V. As such, we will be researching batteries that stay close to this range of voltage. Having 650mAh and 20C discharge rate with boasted flight time of maximum 15 minutes, it should be easy enough to find a replacement battery with more ampere hours. The standard battery weighs 8.5 grams with $10.56 per battery. The lightness of the battery gives us an idea of the amount of weight the quadcopter can carry. This would mean we definitely would have to power the Raspberry Pi off of the quadcopter's battery instead of attaching a battery of its own onto the quadcopter. This is doable, but it is not prefered.

Moving on to the ZOP LiPo battery, it also has a voltage of 3.7V with a 20C discharge rate. The ZOP has an incredible ampere hour of 1000mAh. Though not quite double the time of the standard battery, the difference is significant enough to have a sizable increase in flight time. This does mean, however, a much heavier weight at 35 grams. The manufacturer does not release the exact amount of weight the Traxxas has for it maximum payload, but this

weight may be too much even for estimates found. At $9.98, it is our cheapest option, but our most unlikely due to weight.

The Syma X5C-1 X5C X5A, as with the other two choices, has a voltage output of 3.7V and a discharge rate of 20C. The Syma has 680mAh, which is larger than the standard battery, is almost twice as heavy at 14.17 grams. For not even 10% the battery life of the standard battery with over 150% the weight, this battery is highly unlikely to be our choice. Its price of $14.00 makes it even more so.

If we need just a little bit extra time, the Syma battery will be our first choice, but until we hit that point, the standard battery that comes with the Traxxas 6608 will be our battery of choice for that quadcopter.

4.1.3 QuadCopters

The quadcopter will also have to support the weight of the components we will want to attach to it. It needs to support the added weight of the raspberry pi, the power supply of the raspberry pi, the camera attachment of the raspberry pi, and the transmitter that will communicate between the camera and the host machine over WiFi. Balancing of the weight must also be taken into consideration so we do not hinder the flight capabilities of the copter. As such we will be looking at a variety of possible build kits as well as researching individual components to be used in any copter prototypes. Early on we found we had access to a model copter called the HPQ1 Rotor Concept controller, this copter is advertised to be fully ready to fly out of the box, and controlled by a standard 6 channel digital radio receiver, as well as an already integrated flight controller board with advanced barometer, accelerometer and magnetometers. The 3 Axis gyro and inertial based self stabilization system (with 6 DOF) as well as fail safes for signal loss and low battery make this a strong start in our testing and development. Some important details about the safety features provided by the copters built in flight controller that are of interest to us are visible in table 4.1A on the next page.

This copter comes with various features that are also included in most high cost copters, itself costing a steep price of $899, so having access to this copter will possibly greatly reduce our budget limitations as we can then focus more on other aspects of our system. Its maximum payload capacity (up to 1.2 pounds) utilizing higher performance blades than what comes bundled with the standard will also be taken into account for any early camera/battery setups.

Loss of RC Signal
● In case of RC signal not being detected during flight, the aircraft will enter the security protection mode (SPM) ● In SPM, the aircraft will emit a long "b-e-e-e-e-p" tone intermittently. ● The aircraft will not fly until an RC signal is received from the controller.
Start-up throttle protection
● During power up, if your radio throttle stick is not in the lowest position (zero throttle), The SPM will be activated. ● In this state, the aircraft will not respond to any command until the throttle stick is placed in the lowest position.
In-flight Protection during RC signal loss
● a. If RC signal is lost or interrupted while in flight, the aircraft will immediately self ● land and an intermittent beep tone will also be emitted. ● b. When RC signal is regained, this protection will be deactivated and the aircraft ● can continue flying
Low battery protection
● Upon detecting low battery, the aircraft will beep intermittently while still flying. ● Please land as soon as possible and replace battery. ● If this warning is ignored, the aircraft will slowly power down and self land. You ● still have flight control during this time, but not throttle control. ● Default is 3.2 volts per LiPo cell.

Table 4.1C : Safety Features List for HPQ1 Copter (Permissions pending from rotorconcept.com)

For backup we will still be considering looking at alternative flight controllers and products in case we need more prototypes or wish to implement multiple copters into our system, as this all comes pre bundled we may run into some complications if we cannot replicate a similar rc signal/receiver's functionality into our hand controller then we might have to drop support for this copter. Regardless, early testing will still be alleviated in the Oculus Rift integration department, as the inner workings of the copter are of little relevance to the camera module system we will be installing onto any copter we end up using. Any complications that arise once we begin integrating the hand controller will be dealt with as the need arises.

4.1.4 Flight Controllers

The decision of the flight controller is very important, as it will decide how much customization we can include into the controller. The implementation of the flight controller could affect whether we will use RC controls or whether we can send input signals over WiFi. These are important issues that need to be taken into consideration. If we utilize RC controls we may lose out on digital features we could have added to our overlay to help with flight controls. But we gain affordability and a wider range of accessibility. Having RC controls would be preferable for any outdoor setting where we might not have access to a strong wifi signal (such as having to resort to a phone hotspot for example), and would keep in line with our attempts to make each major component its own singular functioning system. If we consider keeping our controls over WiFi we have to consider the bandwidth requirements and costs of extra transmitters and receivers, as well as extenders if we wish to expand into longer ranges besides indoor use. There are various flight controllers on the market we can choose from, as copters and miniature aircrafts are a common hobby, we simply need to choose one that will fit our needs, preferably with GUI's that are easy to navigate , possibly GPS features to assist in fail-safe situations or to assist with the controls of the hand controller to avoid crashes and user errors ,as well open source development so we can customize any small details we might require for features. Consideration is also going to be taken into possibly programming our own flight controller.

Flight controllers of particular interest that appeared during research were the OpenPilot controller, MultiWii controller, and the ArduPilot flight controller systems. Of particularly notable mention is the MultiWii series of controllers[?]. The MultiWii software is open source as was originally developed to utilize the the gyroscopes and accelerometers from the Wii Motion:Plus and Nunchuck accessories, but future improvements allowed compatibility with other sensors. It is also commonly integrated using an Arduino board which group members have showed personal interest in utilizing in our project. An extensive list of instructions and forums for help are located on the host site, as well as a detailed overview of a sample project including RC components and the building steps for linking the microcontroller with the PCB from the Wii accessories, which would be of great assistance. The affordability of using WiiMote accessories could help drastically lower costs for replacements ($20 average), and the ease of purchase would possibly save us as we could purchase these accessories off various major retailers that contain any electronic entertainment sections. (Best Buy , GameStop, Target).

There are basic setups for connecting the two pcbs that generally only require a few connecting wires. Below is an example of the module being connected to an Arduino Pro Mini that seems like an affordable and easy to implement option, that can be accomplished with basic soldering. Having easy to build, and more importantly with clear instructions are a vital component of our project, as many of the group members are inexperienced with soldering and have little hands on experience messing with PCB's the abundance of information specifically towards the implementations of the MultiWii could be a great asset. Even should the group discover more effective alternatives, a considering has been made towards acquiring a MultiWii setup for the purposes of experimenting in early

development stages, which could help when we move on to the development of our hand controller.. Below are some images detailing the process of how we could connect an Arduino-Mini to the WiiMote PCB's, with exact pin connections and wires. Similar schematics are available on the forum, should we decide to use an alternative implementation that does not utilize the Arduino Mini.

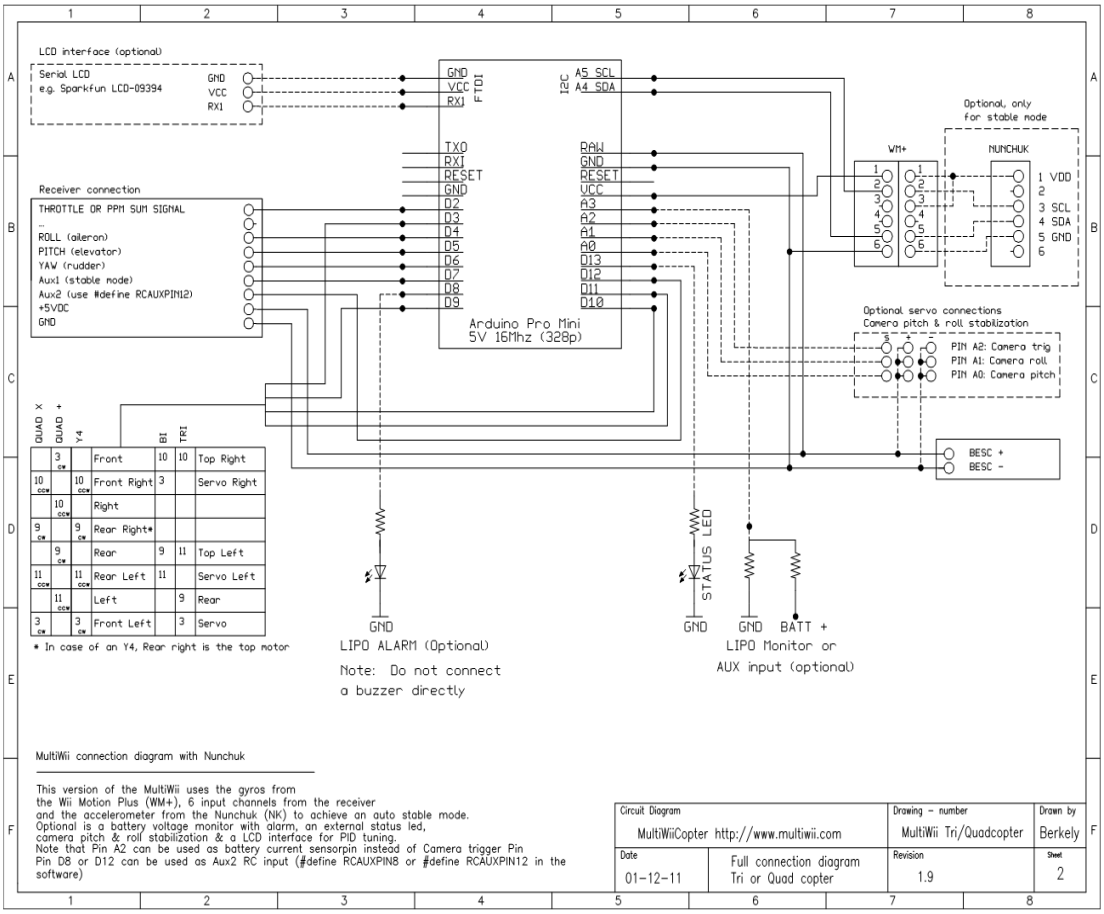


Figure 4.1D: Full diagram of pin connections for MultiWii system (permissions pending)

The ArduPilot series of Flight Controllers were originally based on the Arduino family of MCU's.Future expansions and improvements however outgrew the Arduino environment and support for various MCU's has grown. ArduPilot is an Open Source autopilot software system that supports helicopters, fixed wing aircraft, and more notably for the purposes of our project MultiCopters (Specifically QuadCopter). Boasting a large growing community with support for new developers. There is a plethora of information on site at http://copter.ardupilot.com/ that would leave us access to tutorials related to further developing any base code and how to customize it to our needs. The main flight code is written in C++ which would help us stay consistent in our programming techniques as group users expressed wishes to code in C or C++. Along with the great support, there is compatible software the APM:Copter that we could utilize in simplifying the control of the

copter and incorporating auto-pilot features could help simplify the more fine-tuned controls that might not be feasible on our hand controller.

A list of similar projects are also available on site, and more research will be done to analyze the different techniques used to implement the flight controller and how we could utilize similar approaches in our own development. Something important to take note of however, is the limitations that we would run into, as we would be required to utilize their hardware and associated compatible sensors to make full use all the features available. The group will discuss this in detail, outlining the benefits of time saved and consistency versus the extra costs that would be associated, and how they impact our capability to stay within our budget.

Pixhawk		**APM 2.6**	
Dimensions		**Dimensions**	
Size	1.96 in x 3.21 in x .613 in	Size	2.76 in x 1.77 in x 0.59 in
Weight	1.31oz	Weight	.99 oz
Processor and Sensors		**Processor and Sensors**	
Processor	32-bit STM32F427 Cortex M4 core with FPU 168 MHz/256 KB RAM/2 MB Flash 32 bit STM32F103 failsafe co-processor	Processor	8 bit microcontroller Atmel 2560
Sensors	ST Micro L3GD20 3-axis 16-bit gyroscope ST Micro LSM303D 3-axis 14-bit accelerometer / magnetometer Invensense MPU 6000 3-axis accelerometer/gyroscope MEAS MS5611 barometer	Sensors	3-axis Gyroscope , Accelerometer, High performance Barometer
Power	Servo rail high-power (7 V) and high-current ready	Power	3DR Power Module with XT60 connector (5v)

Table 4.1E : Details comparing the Pixhawk and APM 2.6 flight controllers

Keeping in mind that as our main goal is user controlled flight, we want to ensure we do not detract that experience from the user in favor of taking an easier road in the development process, as such our decisions will be weighed towards the baseline functionality the controllers provide with less importance placed on particularly advanced features that we will not be making use of. Both controllers are readily available so stock is not a worry.

A closer look will be required to see the readymade autopilots and whether they suit our needs, or if it will be important to look at other options to ensure compatibility with our controller. Following is a table detailing some of the differences in two of the leading choices in flight controller's with ArduPilot support should we decide to incorporate them (The Pixhawk and APM 2.6 modules).

As can be noted in 4.1E, the Pixhawk boasts a much stronger set of processors and requires more power. Further research showed that the Pixhawk, due to being more modern, has more consistent support from developers. The APM 2.6 while being a solid controller, is at the absolute limits of its functionality, using every last bit of processing power it has to run the compatible ArduPilot: Copter software, and as such leaves little room for improvement and customization.

The Pixhawk built in fail-safes would be of much interest to prevent damage in our development stages to our copter and anything to increase safety is well desired to prevent extra costs in rebuilding, much like insurance the extra cost could be considered worth the price of safety. A particular detail of importance is PPM functionality common in RC receivers, which could help with compatibility issues on our hand controller. Of the two choices there is a steep $50 price difference, but it is possible that the functionality of the cheaper APM2.6 will be enough for our purposes should we decide to use the ArduPilot line of controllers.

The APM 2.6 does not come packaged with full Autonomy and requires a separate GPS extension, and depending on how much autonomy we want to have on the copter for backup, we could skip that functionality and lower the cost of the APM 2.6 even more by not purchasing the GPS module, the main attraction of these flight controllers is the ready-available flight software for basic functions that we could build upon.

The last flight controller system of note is the OpenPilot line. OpenPilot is another Open Source project, with a thriving community. It was originally developed with low-cost stabilization and auto piloting in mind. This software is compatible with multi rotor systems (from 2 to 8 rotors) so it is more than suitable for our QuadCopter. Because every part of OpenPilot is open source we have access to hardware designs to print out our own PCBs and possibly save costs from using a pre-made controller. Every step of their original software development is accurately detailed and laid out for any new developer to study and recreate their own similar projects. Again, as most of the goals we wish to accomplish are in new territories where most of us are inexperienced, detailed steps and guidelines will greatly streamline our build process in the upcoming Fall semester. The more we research flight controllers, the more notable the advantages of utilizing a pre made piece

of hardware become. The included OpenPilot GCS (Ground Control Station) software, that is also open source and freely available allows for easy configuration of the flight controller's utilizing OpenPilot software. Any time saved on the copter in the long run will allow us to better focus our efforts on creating our functional controller. Should we choose to purchase an OpenPilot flight controller, of notable mention is one of their earlier models the OpenPilot CC3D copter control due to its advertised easy usage with Plug and Play functionality as well as Auto-Level safety fail-safes that would be useful to ensure we do not damage our crafts in the phases of development where we might have issues with consistent signals as we develop our hand controller. Some more in depth details about this controller will be detailed below.

OpenPilot CC3D Copter Control	
Features	● Auto-Level support on Tri,Quad,and Hexa Copters ● FlexiPort : Providing both I2C connectivity and dual serial ports ● All associated software is free and open source ● Direct High Speed USB support, Plug and Play ● 4 MBytes of onboard storage
Sensors	● 3 Axis Gyroscope Array (IDG-500 and ISZ-500 models) ● 3 Axis Accelerometer (ADXL345)
RC Support	Up to 6 PWM Channels, PPM support as well as support for multiple receivers
Size	36mm x 36mm 4 Layer PCB
Weight	.2 oz

Table 4.1F : OpenPilot CC3D copter control specifications

At a first glance, the OpenPilot CC3D has some notable features. The Auto Level support is something that we were looking for in most of our choices, as our primary concern was keeping the copter at a consistent altitude in the case of controller failures or loss of connection. We wanted to avoid the copter dropping back down and possibly damaging it self. Like previous detailed controllers, a list of similar projects are available and well documented on the OpenPilot Wiki, and will be a valuable resource.

The main appeal of the CC3D is its low cost, previously detailed Flight Controllers were in the upper $100 range, some variants in the $200 range. Because we are trying to minimize our budget as much as possible, due to possibility of needing multiple prototypes and to take into account component failure, the $100 price tag of the CC3D is incredibly appealing. It comes with features also available on higher end flight controllers, and while it does not directly support GPS functionality, that is a feature we are more than willing to forego, in favor of a cheap and easy solution to autonomous altitude management.

The high compatibility for various forms of RC signals will likely lead us to use this flight controller in our final build, as the hardest portion of our system to make compatible will be the controls from the glove. Due to the time constraints of the summer semester and funding problems, we are unable to truly get any hands on experience with any of the detailed Flight Controllers yet, but we wish to get a closer look in upcoming Senior Design II and our first approach will likely be with the OpenPilot CC3D.

4.1.5 Camera

In the early stages of the project concept planning we had discussed incorporating cameras into the system, and of course that became a necessary component of our final idea to provide vision to the Oculus Rift. We were however unsure of exactly what kind of camera we would need to utilize, as various factors could heavily impact our overall design. The size of the camera and weight are of course of major concern, as we have a limited mounting platform onto which we can place the camera.

This issue was further expanded upon when the realization (which with further research , a stronger understanding of the Oculus Rift was obtained) that we would possibly have to accommodate more than a single camera to properly integrate the full 3D view of the Oculus Rift and maximize the immersion our customers will experience. Standard webcams posed an immediate concern for space management on the hull of our Quadcopter, and we want to avoid clutter to streamline the final design and avoid balance issues.

Pricing on high quality webcams also seemed rather high ranging from $30+ for a component we expected to be a minor aspect of the build and when we have to get replacements and test different varieties the cost starts adding up. Common USB webcams usually either don't immediately support integration with the Raspberry Pi and of course take up a limited resource on the board, the USB ports, which comes with their own separate concerns of power management. These concerns however were alleviated when we came across a particular piece of hardware.

The Raspberry Pi Camera module is an add-on built specifically for the Raspberry Pi. It clips on through on board sockets using a CSI interface specifically designed for interfacing with cameras. This alleviates the requirements of using up the limited USB ports available to us on a Raspberry Pi board, as well as gives us a fully compatible video feed with high data rates (with the ability to send pixel by pixel data).

This tiny 25mm x 20mm x 9mm camera weighs just under 3g giving us a high range of mobility and keeping us that much closer to the minimum weight requirements we desire. This tiny module boasts of capabilities up to 5 Megapixels (2592 x 1944) resolution on static images, and a video recording support that goes up to 1080p30 which is more than enough for either Oculus Rift development kit we might end up using.

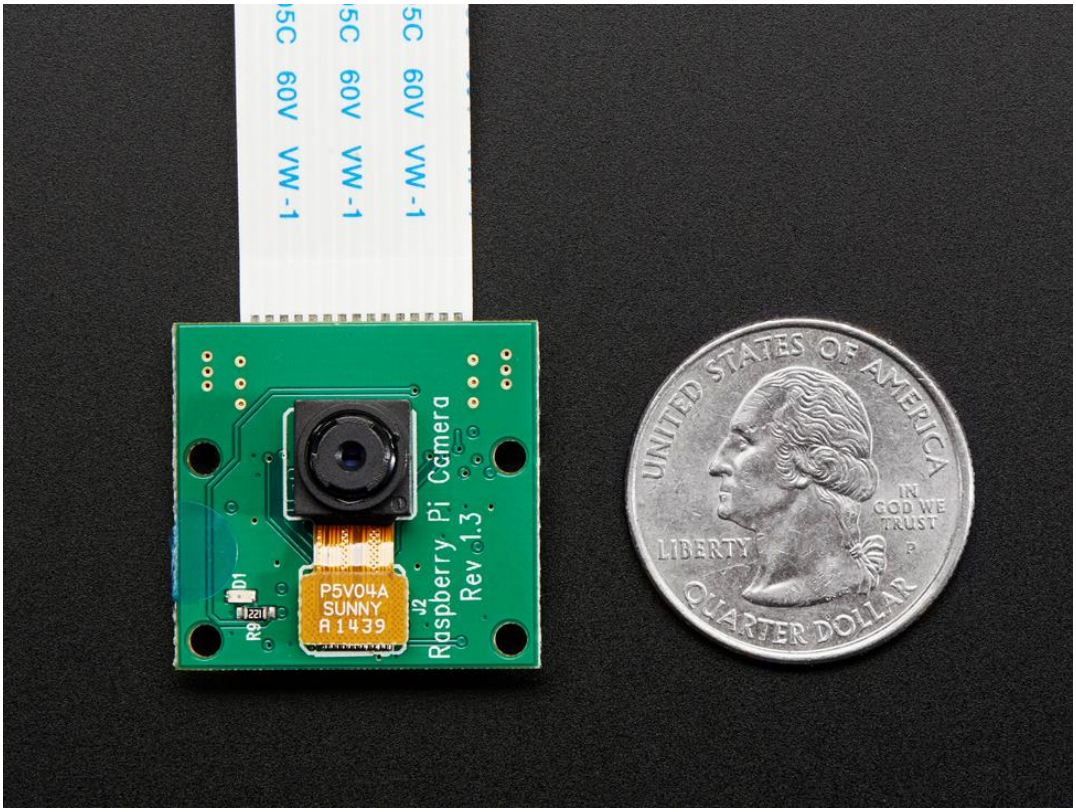


Figure 4.1G : Size comparison of Raspberry Pi camera and a Quarter

This combined with the full support from the Raspbian operating system (Native operative system of the raspberry pi) which we will be working with extensively leads us to believe that this camera will certainly be included in our final build. We plan to get our hands on these modules as soon as possible into the development process to test data rates and optimize latencies before proceeding onto the more technical aspects of our project and attaching it to the copter. The price tag of $30 is manageable within our budget to purchase 2 or more of these early on and begin experimenting, as well as being widely available from multiple retailers, we won't have to worry about stock.

4.1.6 Software

In the process of choosing our programming language we have two main pools of consideration. There were our main choices, Part A, this is due to either prior knowledge or recommendations. The second chart, Part B, talks about languages we considered, however we did not feel they were right for the job.

Since most of our system environment is relatively low level programming we required our programming to be done in a language that allowed us to have full control of the system. This provides detailed control that is also quick and efficient. Our goal is to reduce the hardware requirements as much as possible in an embedded system since our hardware can limit us very quickly if a poor implementation is done.

Programming Languages Chart Part A	
Assembly	Pros: - Learning ARM assembly - Minimize overhead Cons: - "Reinventing the wheel" - Harder debugging - Significant increase in development time - Project's scope is too large - Processor Dependent
C	Pros: - Strong Low Level Language - Manual Control of resources - Arduino/RPi natively run C code - Manual Memory Control - Most of the group knows C Cons: - C can be unforgiving - Manual Memory Control - Lack of OOP - Portability - Lesser "Reinventing the wheel"
C++	Pros: - OOP Concepts - C++ is a superset of C - STL library - More common in Game Engines Cons: - Poor GUI libraries - Portability - Learning curve

Table 4.1H: Programming Languages chart

With our system in mind we planned on using primarily the C language. Where Object Oriented methodologies would help we could easily feed data from a C program into a C++ program. We felt that Java unfortunately could become way too bloated since garbage collection in Java is not nearly as clean or strong a s the C languages. C# really did not fit the cut either since it is a language primarily used for Windows-based machines, our goal is to have everything within a linux environment. However the exception here is that we will likely use C# for the Unity game Engine, trade off with that engine discussed further below.

Programming Languages Chart Part B	
C#	Pros: - Strong Windows support - Unity Game Engine support Cons: - Fairly Windows specific - Runs in a Virtual Machine
Java	Pros: - Relatively easy language - Cross Platform (JVM) Cons: - Runs in a Virtual Machine (JVM) - No Garbage Collection control - Poor application footprint
Languages brought up for consideration: Ada - For employable experience, and it is used in controlled systems. Python - Did not feel it would handle the low level stuff well, in addition to being too high of an abstraction for our needs. PHP/Ruby - Would have been an approach if we had a web interface, since we felt these technologies were better suited for the web. Javascript - This is in the same situation as PHP/Ruby, however Unity game engine does utilize Javascript, so depending on our implementation of Unity we may use JavaScript.	

Table 4.1I Programming Languages Chart

While assembly was considered, we quickly realized that due to the scope and amount of programming going into this project, that a team of two dedicated software developers would not be able to successfully implement everything needed in assembly in such a short time, and we felt that the performance gained from writing assembly would be minimal compared to C.

Our goal was to aim for a system that primarily used Unix-based Linux and try to contain the technologies to as little diversity as possible. However implementation details have lead us to believe we may need to branch out to the Windows operating system as well. Initially we wanted to avoid this as to make our network programming easier, since Window and Unix machines have different libraries built in.

We also felt that Linux would give us a stronger control of our system, particularly if we felt the need to use BASH scripting for our environments or just have general hardware/software configuration control that is not as obfuscated as Windows. The specific distro of linux we are planning on using is the Raspbian distro. We debated between Raspbian and Arch Linux ARM, however the results became clear that the Raspbian OS was better supported for the Raspberry Pi since that was the platform it was targeting. More

particularly we felt that if we ran into any problems the community for the Raspbian OS would be of better support since it revolves entirely around the hardware we plan on using.

Game Engine Comparison	
Unity Game Engine (5.0)	Pros: - Excels at mobile market - Exporting cross platform is easy - Free version Cons: - Pro version - C# - No prior team knowledge
Unreal Engine 4	Pros: - C++ development - Free completely - Visually Stunning Cons: - Traditionally more graphic intensive
CryENGINE	Pros: - Visually Stunning, surpassing the others Cons: - Monthly subscription - Steep learning curve

Table 4.1J: Game Engineer Comparison Chart

After our basic analysis of the few game engines that Oculus Rift supports directly, we decided to go with Unity. While it does create a learning requirement for our team we felt that an engine that would require less resources would be better. However the game engine is mainly used for the overlay and feeding video to the Oculus Rift. The goal with this is ultimately to utilize the Raspberry Pi as the host machine, however if Unity is unable to run sufficiently on the Raspberry Pi, then we will likely attempt the same functionality with the Unreal Engine 4.

Programming of the arduino fits right in with our programming language choices, as it utilizes both C/C++. We will choose to write most of the arduino code in C as we do not feel we need the object oriented features that C++ presents us. The arduino's built in libraries make it easy to read signals from I/O pins. The arduino's codebase will be rather small, it will mainly consist of an aesthetic portion and a functional portion. Our plan is to take in all of the data and convert it into a single memory struct and pass it to the host machine to send to the quadcopter.

4.1.7 Hardware

In researching for the hardware, we will start with the microcontroller that will act as the "brains" for the glove. The microcontroller will be detecting what data comes in from the accelerometer and gyroscope combination that we will discuss later. The microcontroller will then translate this data from the combination into useful directions. For example, the data from the accelerometer and gyroscope will tell the microcontroller that the glove is positioned at relative frame (0, 0, 0).

In real life, this will mean the that user has their hand on a flat surface. The microcontroller will translate this data as the hand being on a flat surface and this will be sent to the host machine. The host machine will then work with the microcontroller to send this direction to the quadcopter which will accept the command to stay still. Once the glove is raised to some position (0,0,Z), meaning that the user has raised their hand to some distance Z, the data that will be sent to the microcontroller will then be different accordingly.

The microcontroller will translate this movement as the hand being raised and will work with the host machine to give the signal to the quadcopter to rise. The quadcopter will then turn on its motors and will raise in scale with the user's hand. This was the plan, but it did not come to fruition with a single glove. Instead, a signal was sent to the quadcopter via the computer the glove was connected to and the quadcopter raised itself to a certain height. The glove was then put in charge of controlling the quadcopter and controlled th forward, backward, left, and right motions.

So in deciding this "brain", we had considered the TI Launchpad, the Arduino Uno, and the Propeller Education Kit - 40. The reason we are looking at kits and boards while looking at microcontrollers is that the kits and boards will act as a good way to prototype. Instead of immediately soldering and making a printed circuit board with our microcontroller, gyroscope, and accelerometer, and any other components that will be included in our package, we will use a breadboard and the kits and boards, along with the later to be decided accelerometer and gyroscope combination, to make a changeable and quickly adaptable prototype.

The TI Launchpad will come with the TI M430G2553 microcontroller. It has 16 data buses and operates at 16MHz. IT has 16KB of storage, 512B of RAM, 8 digital Input/Output channels, 8 analog Input/Output channels, and costs $4.30. While the amount of data busses are incredible for such a cheap piece of equipment, there is not a large amount of RAM or memory to work with the code we will eventually be giving it. The code we will be writing will take a large amount of memory for reading three axis of motion and a spherical degree of rotation. We are very capable programmers and builders, but it would take machine level language code to make it efficient enough to work on such a small frame, we predict.

The Propeller Education Kit - 40 will come with the P8X32A-D40 micro controller. An incredible 32 bit data bus makes this extremely appealing, along with its 80MHz speed. It's storage is a sizable 32.7KB with an incredible 32.7KB of RAM. An interesting aspect is the large size of digital Input/Output channels being at an exorbitant 32 channels.

Meanwhile, the P8X32A-D40 has to analog Input/Output channels to speak of. The kit, itself, is $129.99. While this piece of technology is clearly superior to the TI M430G2553, it is extremely expensive. While it would be luxurious to be able to afford this kit and be able to use it with one try and get everything working the first time, we are very aware that we will be making mistakes. Thus, to work with something so expensive, we have decided to decline using it. Our wallets would hurt too much, and we would be too afraid to experiment in our attempts to out goals.

Finally, we have the Arduino Uno. The Arduino Uno's microcontroller is the ATMega328. Its data bus is a surprisingly small 8 bits while processing at the same speed as the TI M430G2553 at 16MHz. However, better than the TI equipment, the ATMega328 has 32KB of storage, just about matching the expensive Propeller microcontroller. However, quite between the two counterparts, the ATMega has 2KB of RAM. Its digital Input/Output is at a sizable 14 channels with a smaller analog Input/Output at 6 channels. The entire board is $24.95, which makes it significantly cheaper than the Propeller kit.

Although not nearly as powerful as the Propeller, and while more expensive than the TI Launchpad, the Arduino Uno appeals to us as the best choice in dealing with prototyping and further in creating a working printed circuit board. Additionally, we all have more experience in working with the Arduino environment than we ever had with working in the Parallax environment. The Arduino environment also has a few more tools that we have readable and understandable access to than the Launchpad. Thus, we choose the Arduino Uno development environment as our "brain" of the glove.

Next, we looked at the accelerometer and gyroscope combination. As stated earlier, the combination of the two devices will be our sensor to measuring how the user is moving their hand. The direction of movement of the hand (mainly whether the hand is moving up from the flat surface, or down towards the flat surface) will be largely measured by the accelerometer. The direction that the hand is in, (whether the hand is bent inwards towards the user's body or bent outwards away from the user's body) will be largely measured by the gyroscope. It was described before how the position of the hand will affect the quadcopter's movement of up and down.

The quadcopter was planned to also know to move forward, backwards, strafe left, or strafe right as according to the accelerometer. If the user's hand is at some position (0, 0, Z) and the user moves their hand away from their body some distance (0, Y, Z), the Y distance will be measured and put into a range of speed. So if the user puts their hand between distance 0 and Y, the quadcopter will move at speed 1. If the user moves their hand between distance Y and 2Y, the quadcopter will move ahead at speed 2, which is proportionally faster than speed 1. And so on. Similarly, if the glove is in position 0 to X, the quadcopter will move to the side at speed 1, and so on in the same fashion.

Meanwhile, if the user's hand is tilted in such a way without moving the hand, itself, the quadcopter will turn in scale with the user's hand. Similar to the above example, if the user moves their hand between degree 0 and ∅(theta), then the quadcopter will turn to the side in that direction at speed 1, and so on in the same fashion. Together, this is how the

accelerometer and gyroscope combination will work to translate via the microcontroller and host machine to the quadcopter on how to move.

In choosing the accelerometer and gyroscope combination, we initially looked at the SparkFun Triple Axis Accelerometer and Gyro Breakout board, the FLORA 9-DOF Accelerometer/ Gyroscope/ Magnetometer, and finally the 9-DOF IMU Breakout board. Once again, we are looking at pre-made packages that have the components needed inside of them and are ready to be plugged into a breadboard along with a microcontroller "brain" and be tested. Once thoroughly tested, we will move on to having all of the components integrated into a printed circuit board.

To start, we look at the SparkFun breakout board. This board has a single component, the MPU-6000A. It has a gyroscope sensitivity of 250dps, 500dps, 1000dps, and 2000dps. This is about the range we have determined through looking at more and less sensitive gyroscopes and accelerometers and have determined this range to be perfectly suitable. It's noise is at a workable 0.033dps/sqrt(Hz) while operating at 50Hz. This is manageable, but not preferable. Its accelerometer sensitivity is 2g, 4g, 8g, 16g. Once again, this range is perfectly common and coincides with our goals. Its accelerometer noise is at 400 µg/sqrt(Hz) when operating at 10Hz. This, again, is workable, but not preferable. The package borders on the expensive side while being at $39.95. Although it is viable to use this technology, we sense that a better option is on the horizon.

Moving on to the FLORA 9-DOF, it also has a single component that contains both the accelerometer and gyroscope as opposed to containing them as two separate entities. The part identification is the LSM9DS0. It has a gyroscope sensitivity of 245dps, 500dps, and 2000dps. While not as sensitive as the SparkFun, it is still well within our needs. However, it is odd because the public data sheet for this part excludes the mention of noise when it comes to the gyroscope. Further research at professional user reviews show that they have problems with noise and it is suggested that they work with operational amplifiers to try and cancel out the noise. This would be a larger hassle than we need and so we are already starting to shy away from this product. The LSM9DS0 has an accelerometer sensitivity at 2g, 4g, 8g, and 16g. This is exactly the same as the SparkFun, so we see that the choice between the two will not be from this portion. Again, however, the data sheet fails to mention the noise that comes from its accelerometer. Neglecting to mention the noise for either of its main components seriously discourages us from wanting to use this product. Coming down to price, it is $19.95. While far cheaper than the SparkFun, we again feel as if there is something better for our use.

Now we arrive at the 9-DOF IMU breakout. This product does come as two separate components for its gyroscope and accelerometer with the L3GD20H and LSM303D, respectively. Exactly like the FLORA, the L3GD20H has a gyroscope sensitivity of 245dps, 500dps, and 2000dps. This works well within our needs. The data sheet reports a gyroscope noise of 0.011dps/sqrt(Hz) while operating at 50Hz. This is one third the noise of the SparkFun and, we assume from the earlier mentioned customer reviews, much better than the FLORA. Moving on to the accelerometer, it has a sensitivity of 2g, 4g, 8g, and

16g. Again, exactly as the first two, the accelerometer will be well within our goal of sensitivity. Its noise comes to 150 µg/sqrt(Hz) when operating at 100Hz.

Though at different frequencies, the noise level is much lower than the SparkFun accelerometer noise. To add to the fact, the price of the 9-DOF IMU breakout board is the same as the FLORA at $19.95. Being much cheaper than the SparkFun with less noise, we have decided to work with the 9-DOF. We realize that with two different parts, it means that we will possibly have twice the work to try and get it all to communicate, but with enough practice with the breakout board, we are certain we will overcome these challenges.

Another addition to using this breakout board is a "guaranteed" voltage overload protection. Thus, theoretically, this board will not burn out and its components will be protected when testing the product. This safety measure will help immensely when testing this product.

In the end, we chose to upgrade the 9-DOF to the 10-DOF for added barometer. The barometer measures the atmospheric pressure to then give a sense of altitude. This was to be used for the Z direction of giving the copter lift with the raising gesture of the hand. This was not able to be utilized and instead the copter was just given a certain amount of thrust from the computer and the glove took over the rest of the controls.

We had attempted to use a raspberry pi as a host machine, further details about the various modules and their capabilities will be outlined at a later point in our discussions about our host machine and its use in our camera module that will be attached to our copter. This was researched and tested to see if the raspberry pi can handle the Oculus Rift, the glove, and the WiFi transmitter that will communicate with the camera on the quadcopter.

We tried to make sure that the raspberry pi used as the host machine had a portable power source that can handle all of the added components. For the purposes of development, we had utilized a group member's personal computer as our host machine, until we could optimize our system to a point where we have a more solid grasp on the requirements and can downgrade to a smaller machine for portability and ease of use. However, for the sake of time, the end product consisted of one Raspberry Pi acting a transmitter to another Raspberry Pi for the Oculus Rift while an Arduino transmitted to another Arduino that was connected to a teammate's computer to act as the controller.

4.1.8 Oculus Rift and related subjects

Another major component is the Oculus Rift itself. The details of the hardware itself are fairly easy to find and are not particularly of interest to the design, however the implementation of common apps and software that contain Oculus Rift compatibility are of great interest to us. There are currently two major models of the Oculus rift. The Development Kit 1 and Development Kit 2.

Oculus Rift DK1		**Oculus Rift DK2**	
Display		**Display**	
Resolution	640 x 800 per eye	Resolution	960 x 1080 per eye
Refresh Rate	60 Hz	Refresh Rate	75 Hz, 72 Hz, 60 Hz
Persistence	~3 ms	Persistence	2 ms, 3 ms, full
Viewing Optics		**Viewing Optics**	
Viewing Optics	110° Field of View (nominal)	Viewing Optics	100° Field of View (nominal)
Interfaces		**Interfaces**	
Cable	10′	Cable	10′ (detachable)
HDMI included	Yes	HDMI	Yes
USB Device included	Yes	USB Device included	Yes
USB Host	USB 2.0 (requires DC Power Adapter)	USB Host	USB 2.0 (requires DC Power Adapter)
Positional Tracker USB	USB 2.0	Positional Tracker USB	USB 2.0

Table 4.1K: Rift comparison

The decision of which model we will choose to use could greatly affect the ease with which we can integrate the Oculus Rift into our system, so we will be taking a close look at the differences between the two models and the support available to each to better decide which model is realistically obtainable, and whether the limitations associated will be worth the

costs.A detailed comparison. And discussion of the important features between both Dev Kits will be detailed below.

At a brief glance, the two devices are similar in terms of interfacing, as such with either decision we would likely run into the same issues or lack of issues where cabling and power are concerned.

Oculus Rift DK1		**Oculus Rift DK2**	
Internal Tracking		**Internal Tracking**	
Sensors	gyroscope, accelerometer, and magnetometer	Sensors	Gyroscope, Accelerometer, Magnetometer
Update Rate	1000 Hz	Update Rate	1000 Hz
Positional Tracking		**Positional Tracking**	
Sensors	CMOS Sensor	Sensors	Near Infrared CMOS Sensor
Refresh Rate	60 Hz	Update Rate	60 Hz
Weight		Weight	
Weight	380 grams	Weight	440 grams (without cable)
Included Accessories	HDMI to DVI Adapter DC Power Adapter International Power Plugs Nearsighted lens cups Lens cleaning cloth Control Box containing switches for Brightness,contrast etc	Included Accessories	HDMI to DVI Adapter DC Power Adapter International Power Plugs Nearsighted lens cups Lens cleaning cloth

Table 4.1L: Detailed breakdown of differences between DK1 and Dk2 (Permissions granted from riftinfo.com, see appendix)

The most notable improvements between the DK1 and DK2 models however, is the quality of Optics. While the DK1 has 110° FOV (vs 100° FOV for DK2), the DK2 offers a variety of refresh rates rather than the static 60hz as well as a much higher resolution of 960x1080 per eye as well as better persistence rates. To achieve our highest possible quality we will most likely be looking more closely into the DK2 model for our purposes.

To make the best use of the HD capabilities of the Raspberry Pi: Camera module we plan on using, our best approach is to make the most out of the possible resolution rather than having to scale down and lower the quality of our video feed. The quality, and range of freedom in our refresh rates will likely be worth the tradeoff of lessened support, as the DK2 has been available for a shorter time, apps and support software available to it are not as widely available and detailed compared to the DK1, as well as the increased costs associated with the newer model. Below is a chart listing the sensors and minor details of the two Development Kits.

Because support is hard to find, and from looking up information of similar projects working with the Oculus it's easy to see how many complications arise. Due to the nature of the screen itself on the oculus (the screen is the lcd from a samsung note 3), to get a true 3d view any video feed must be stereoscopic to get the full effect and the resolutions must be properly configured for each eye.

While it is possible to use a basic 2d video feed, a layer of distortion would need to be applied anyway to help with visibility, as without that layer the users may experience dizziness and disorientation. Further research will be done to see if we can accommodate the need for stereoscopic video on a software level, while still being able to use a single camera. And we will be weighing the costs and work involved with this implementation or opting for a dual camera setup utilizing the Raspberry Pi:Compute version of hardware that comes with two USB ports and dual camera functionality.

As we are fairly confident in our decision to utilize Raspberry Pi's we will be utilizing a particular software program called GStreamer to pipe our feed from the Raspberry Pi to our host machine (utilizing Real Time Protocols), once we receive this feed we must add a layer of distortion to simulate a 3d view.

With a single camera we would be required to first split this feed and then stitch it together, which could cause extra delays, while a double camera setup would be more tasking on bandwidth but would provide an easier to manage video feed (having each camera mimic an eye).

4.1.9 Possible Solutions

Discussed in this section are main ideas for the various components as a resolution to the problems brought out in the previous sections. Such components include battery, Quadcopter, Camera, Flight Controller, Game Engine, and Control Signal

Battery: To help with the power problem, we will have to find a battery that has a better life-span than the pre-equipped battery. When dealing with the quadcopter, however, this is not as simple as getting a battery with a larger milliamp per hour lifespan. This is because the larger the lifespan, the heavier the battery. The heavier the battery, the more the load on the quadcopter. Thus, the new battery must be carefully evaluated and the weight of the Raspberry Pi, the camera, and the WiFi adapter must be weighed in and considered when choosing a battery of larger lifespan.

Conversely, if our design permits us to be able to use the Raspberry Pi as a host machine, then it will be extremely simple to get a battery with a larger life span. Since the user will simply be seated either at a desk or on the ground, then a battery large enough to power the Raspberry Pi's weight will be inconsequential. Batteries advertised specifically for the Raspberry Pi boast a few hours of operation, which is more than enough for a quadcopter's maximum flight time of 14 minutes.

QuadCopter: To help with the weight lifting problem, different propellers will be used than the ones initially equipped. Three-leafed propellers will provide better lift while making the blades longer will increase efficiency. This may reduce battery life due to larger power output, but it is a necessary trade-off. We will be looking into various hobby shops and similar retailers to find effective light weight wings for our copter. No particular propellers have caught our eye yet, but once we have developed our video streaming module and have a more exact load requirement we can make better decisions towards what components we will be using.

Camera: The Raspberry Pi: Camera module offers too much support and accessibility to overlook. The resolution it can achieve alone puts it on a higher grade than any average webcam we could consider purchasing, not even beginning to mention its compact size. The cost also keeps it in line with an affordable baseline, in case we need replacements. A double camera setup is currently appearing to be our most appealing build path, given the previous concerns of syncing two separate camera feeds are greatly simplified due to the built in tools and support offered by the Raspberry Pi itself. Our original issues of requiring a separate Pi for each webcam along with the signal strength to send both these feeds in real time can be nearly completely eliminated , while staying within the original planned budget costs we associated with any camera hardware.

Flight Controller: Of the Flight Controllers we researched the OpenPilot and MultiWii software and associated hardware both seemed to be bright prospects. Both were relatively affordable and provided similar features that interest us , such as the Auto Level support to assist with failsafes. Both were also not fully autonomous compared to the other controller we found (ArduPilot), and that fits better within our interests, as we simply want to get some basic functionality. The end goal is of course that control of the copter is mainly done by the Hand Controller we create, as such all the extra bells and whistles of autonomous flights aren't particularly necessary to us which can help us cut costs and avoid the higher end flight controllers. The guaranteed compatibility with common RC receivers and transmitters make them all promising choices.

Video Feed: Regardless of which engine or operating system we decide to move along with, all of our research leads us to believe that Gstreamer software is our best bet at feeding our video. It is supported by various ARM, Linux, MACOSX and Windows platforms so we would have no worries about compatibility. A well documented Core Library with easy to use and understand AI's, pipelining tools and an advertised capability to provide high data rates/low latency using their light weight data passing functions. Full access to

documentation on their home site , with easy connections to developers for any questions we might have make this a strong option.

Game Engine: Unity seems like a strong contender for our game engine of choice to create our graphical overlays to use with the Oculus Rift. Unity Pro is easily accessible, although the release of Unity Free integration with Oculus will be a great asset in early development to familiarize ourselves with the engine and its tools. Lens corrections, layered cameras, and direct rendering functionalities appear to make it the best choice currently available to get working early and quickly. Pending any further development from other engines that may arise during our production period.

Control Signal: The initial debate on control signal for our QuadCopter was reliant on what flight controller we wished to utilize. We considered RC controls vs sending input signals over Wifi, while we realized that the WiFi signals could give us more free reign over how we control the copter and how much integration we can include between the controls themselves and our graphical overlay on the Oculus Rift. Bandwidth concerns arose of course, and the majority of flight controls are all RC based so it felt like we would be making too much unnecessary work for ourselves. Overall, it seems RC is the best choice, and will cause us the least issue.The individual components we will use have not been decided yet, but there are a wide variety of choices we will take closer looks into.

4.1.10 Decision Criteria and Justification

Weight: The size and weight of the camera is very important. Between the batteries and the camera itself, they will be taking a good portion of the total load the copter can carry. Minimizing weight will be an important criteria for any hardware and components we wish to use in our project.

Resolution: Camera resolution should ideally be as close to the default resolution of the oculus, however any discrepancies could be dealt with on a software level. Bandwidth usage is another issue; we have to take into account the usage between a single and double camera.

Battery Life. We need to power cameras,microcontrollers; the flight controller and any other appliances we may find necessary. This will tie in hand to how we decide to control the drone itself, if we decide against Wifi and split the streaming/controlling we could possibly alleviate some bandwidth. We have to make sure any decisions we make still allow us a significant amount of flight time.

Cost: Earlier in the report we outlined some basic price estimates for each component, we want to aim for staying to the lower end of those estimates to ensure we can afford what we need and take into account any part failures or unexpected issues. Extra spending might be warranted if we take a look at the ratio of cost versus time saved and utility.

Support: Any components, hardware,software we decide on using, most of us have little familiarity with. As such any products that come with information ,guides, tutorials and such will be weighed more heavily.

Ideally we would be able to have the lightest battery, the lightest webcam and such without any drawbacks. But budget is still going to be a consideration, we have to work with what we can realistically acquire and that means we have to make some sacrifices. Any decisions made will be done as a group, before any major purchase the group will discuss and ensure that we agree with the pricing and the quality. As we continue through development, the following section will be outlined clearly, concerning any design decisions, components,software we found to be more effective for our goal,. as well as any concerns or issues we ran into and their proposed solutions.

4.1.12 Design Considerations and Solutions to Issues That Arose

Initially we were leaning towards a single camera with a wide angle lens. This would have given us the possibility of having a resolution larger than the occulus that we can then "scan" through using the motion sensor for the oculus. This seemed to be an easier solution than having an additional rotor attached to the camera and syncing the head movement.

However as further research was made towards the Oculus we decided it was best to proceed with a double camera layout to better simulate the stereoscopic view we require for optimal viewing, our initial bandwidth concerns were alleviated after viewing similar projects accomplishing much more detailed video streaming than we planned to use, on similar hardware.

4.2 Overall System and Associated Diagrams

The next sections will discuss the hardware and software architecture for all the major components of the project. These include the microcontroller components and block diagrams for all software related pieces

4.2.1 Hardware Architecture

The glove will be largely made of the microcontroller, the accelerometer, and the gyroscope. All of this will communicate with the host machine and further, the quadcopter.

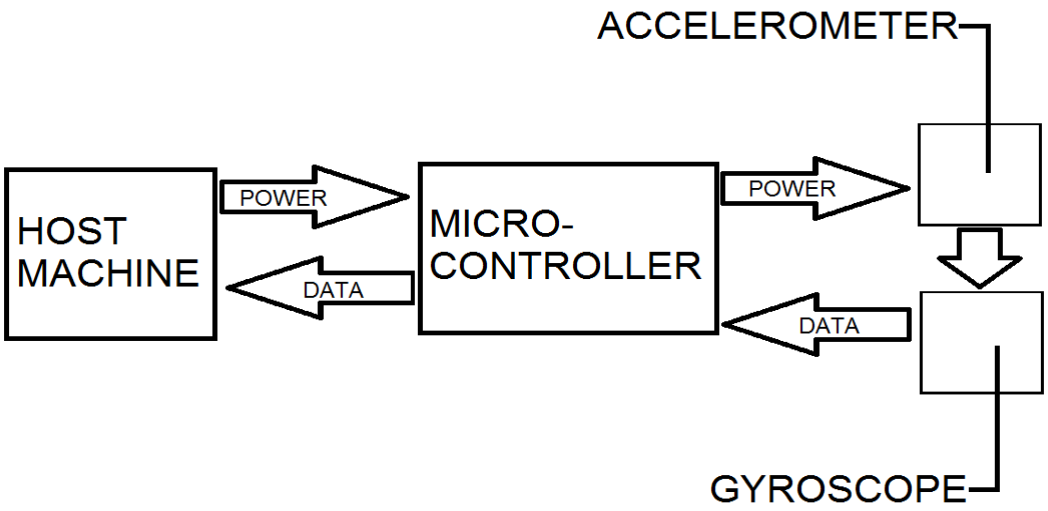


Figure 4.2A flowchart describing basic glove architecture.

The accelerometer and gyroscope are devices that basically give data. If this was issued directly to the host machine, then the host machine would have a lot of data but no commands from the data or instructions on what to do with the data. Thus, the microcontroller will act as a mediator between the host machine and the accelerometer/gyroscope combination that will take in the data, translate the data to useful commands, and give these commands to the host machine that will then give these commands to the quadcopter.

	TI Launchpad	**Arduino Uno**	**Propeller Education Kit - 40**
Microcontroller	TI M430G2553	ATMega328	P8X32A-D40
Data Bus	16 bit	8 bit	32 bit
Speed	16 MHz	16MHz	80 MHz
Storage	16KB	32 KB	32.7 KB
RAM	512 B	2 KB	32.7 KB
Digital I/O	8 channels	14 channels	32 channels
Analog I/O	8 channels	6 channels	0 channels
Kit cost	$4.30 @ TI.com	$24.95 @ Adafruit	$129.99 @ Parallax
MPU cost	$1.90/1 chip	$1.86/1 chip	$7.99/1 chip

Figure 4.2B: Comparison table for micro controllers

So we have to choose from a large selection of microcontrollers. We will be comparing the Arduino, the MSP430, and the Propeller Education Kit - 40. Below is a table with all of the boasted advantages to using each board as stated by their respective specification sheets.

Because the Arduino has more memory with the same amount of speed as the TI Launchpad, it will contour to what we are trying to do more since we are trying to take in a lot of data of positioning and speed and translating this to the host machine. For this reason, we will side with the Arduino over the TI Launchpad.

Because of pricing and because we are more familiar with the Arduino than the Propeller Education Kit - 40, we will side with the Arduino between the two. The P8X32A-D40 is certainly the better microcontroller, given that you are only working with digital I/O, but having to learn to work with an entirely new microcontroller would not be beneficial and probably hinder us in the long run for trying to learn a new environment. Thusly, we chose the Arduino and it's corresponding ATMega328 as our microcontroller of choice.

	SparkFun Triple Axis Accelerometer and Gyro Breakout	**FLORA 9-DOF Accelerometer/ Gyroscope/ Magnetometer**	**9-DOF IMU Breakout**
Part(s) #	MPU-6000A	LSM9DS0	L3GD20H and LSM303D
Gyroscope sensitivity	250 dps 500 dps 1000 dps 2000 dps	245 dps 500 dps 2000 dps	245 dps 500 dps 2000 dps
Gyroscope noise @ 50Hz	0.033 dps/sqrt(Hz)	N/A	0.011 dps/sqrt(Hz)
Accelerometer sensitivity	2 g 4 g 8 g 16 g	2 g 4 g 8 g 16 g	2 g 4 g 8 g 16 g
Accelerometer noise	400 ug/sqrt(Hz) @ 10Hz	N/A	150 ug/sqrt(Hz) @ 100Hz
Price per kit	$39.95	$19.95	$19.95
Price per part	$4.02	$3.69	L3GD20H @ $4.01 LSM303D @ $4.31

Figure 4.2C: Accelerometer/Gyroscope combo comparison

We now have to choose which accelerometer and gyroscope we wish to use. There are combinations that are a single component with both accelerometer and gyroscopes inside them. There are also the individual charts. We will be comparing between the available SparkFun Triple Axis Accelerometer and Gyro Breakout, the FLORA 9-DOF Accelerometer/Gyroscope/Magnetometer, and the 9-DOF IMU Breakout.

Further research has shown that the LSM9DS0 has a lot of problems with noise. The specifications sheet mentions nothing about this as so we are sceptical to try it. Thus between the L3GD20H and LSM303D combination and the LSM9DS0 standalone, we have chosen to use the combination instead.

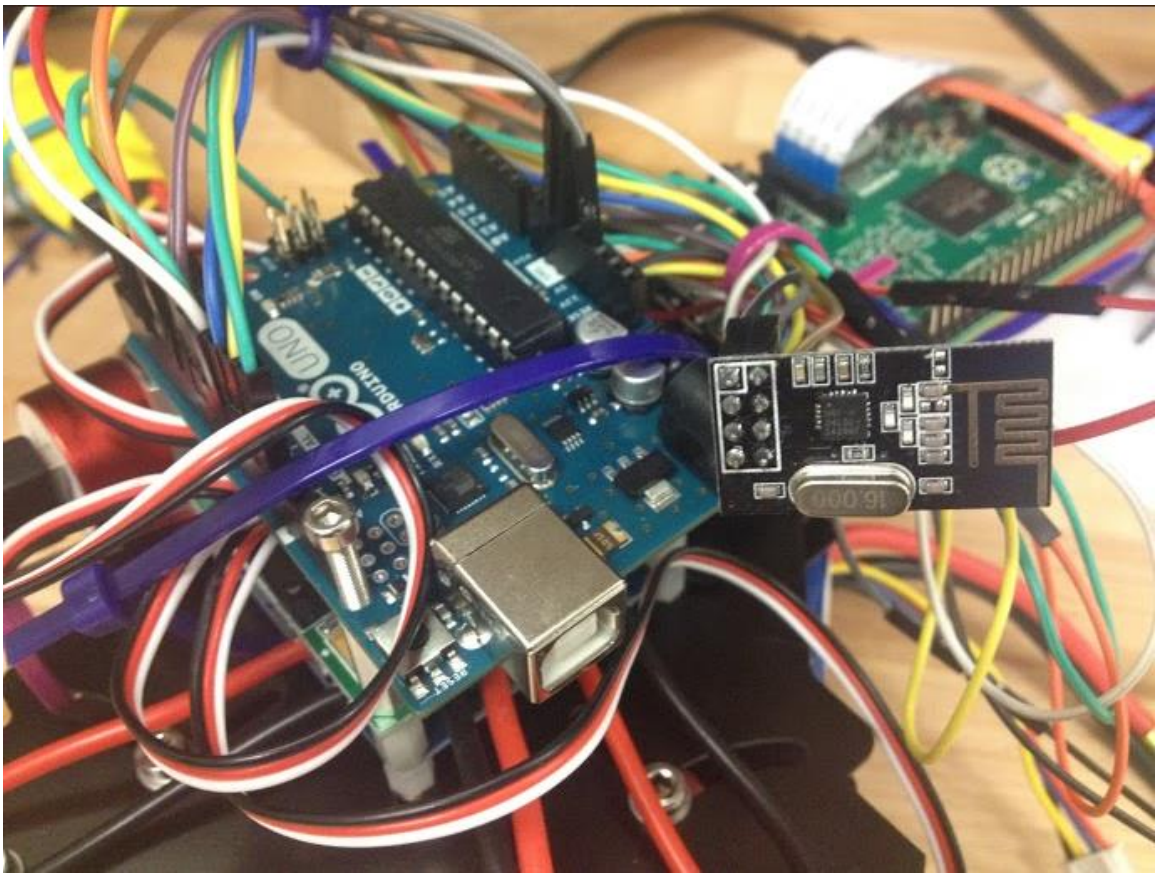


Figure 4.2D Quadcopter Arduino Flight Controller and Receiver

Although it will be twice as expensive to use the L3GD20H and LSM303D combination than the MPU-6000A standalone, the pure amount of noise coming from the standalone is worthy enough for us to continue on and choose the combination. The noise will especially be a problem since we are trying to make the glove as fluid and intuitive as possible. The price difference of an extra $4 will be worth it. We also will buy the 9-DOF IMU Breakout board for extensive testing before attempting to use the combination in a PCB. The breakout board boasts voltage protection, thus making this a viable strategy. Thus, we had chosen the L3GD20H and LSM303D combination. This was modified to be the Adafruit 10-DOF which consisted of the same components as the 9-DOF with the added BMP180 barometer to act as an altimeter if needed.

4.2.2 Software Architecture

For host we will be choosing between Windows and Linux, once we decide on which will be easier to handle the networking and general integrations. On the Raspberry Pi itself we will likely be utilizing the free operating system Raspbian to make use of associated APIs and the compatibility of the various RasPi modules (The camera board and possibly the wifi receiver). This will allow for easy testing, and the usage of convenient software such as Gstreamer to send the camera feed to our host machine.

What will ideally be happening, to minimize any delays and complications. Is to treat the OR itself as a simple monitor. On the software level, what should happen before any camera feed reaches the oculus, the feed needs to be distorted to be viewed more comfortably on the oculus, and a graphical overlay will be applied on top of the feed in real time to keep track of important statistics (Hand position from controller, distances, battery life etc), this should all be handled on the host machine itself.

The controller should ideally be able to "mimic" keyboard inputs, and be read in as such, to test in two stages. Testing the software and input functionality on its own with a keyboard, and then testing the controller functioning properly.

Software block diagrams

Block Diagram Legend	
Gray	Hardware
Red	Operating System
Yellow	Proprietary Software, may use API's
Green	Data Flow
Purple	Oculus Rift Output

Figure 4.2D Legend for block Diagrams

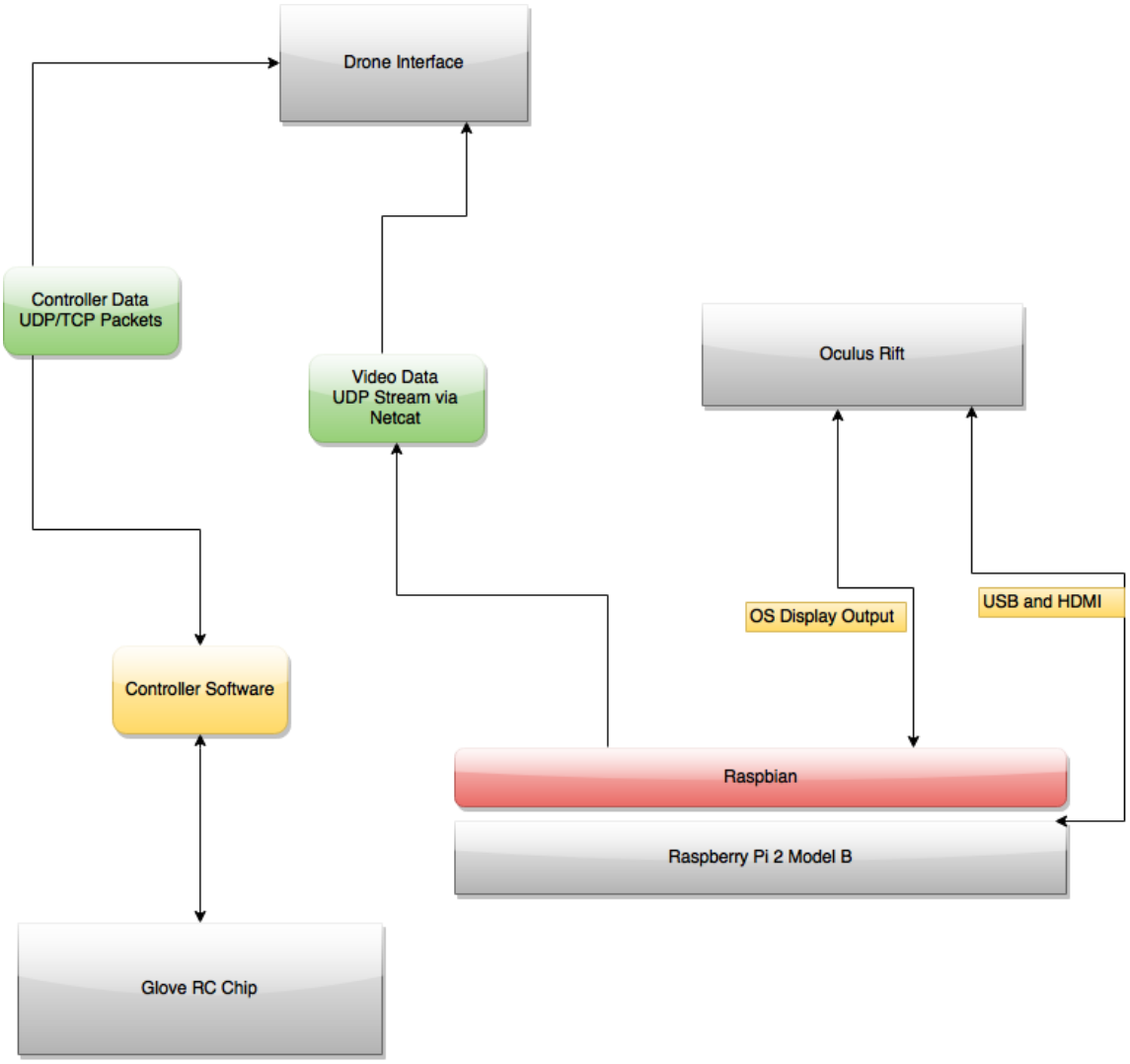


Figure 4.2E Host Machine Setup

Host Machine: Our host machine diagram uses our best case scenario in which the host machine is a Raspberry Pi. This machine will have three connections, two of which feed data for the software layers with the third connection being a wifi adapter. The Oculus Rift will be connected via USB/HDMI and will receive the video output from the quadcopter's camera. The Raspberry Pi on the Copter will stream the camera data via Netcat Linux Module via UDP. The controller software essentially just sends data and translates from the hardware to the quadcopter for flight. The game controller is a software layer that adds game functionality to the quadcopter, primarily just sending packets between each host.

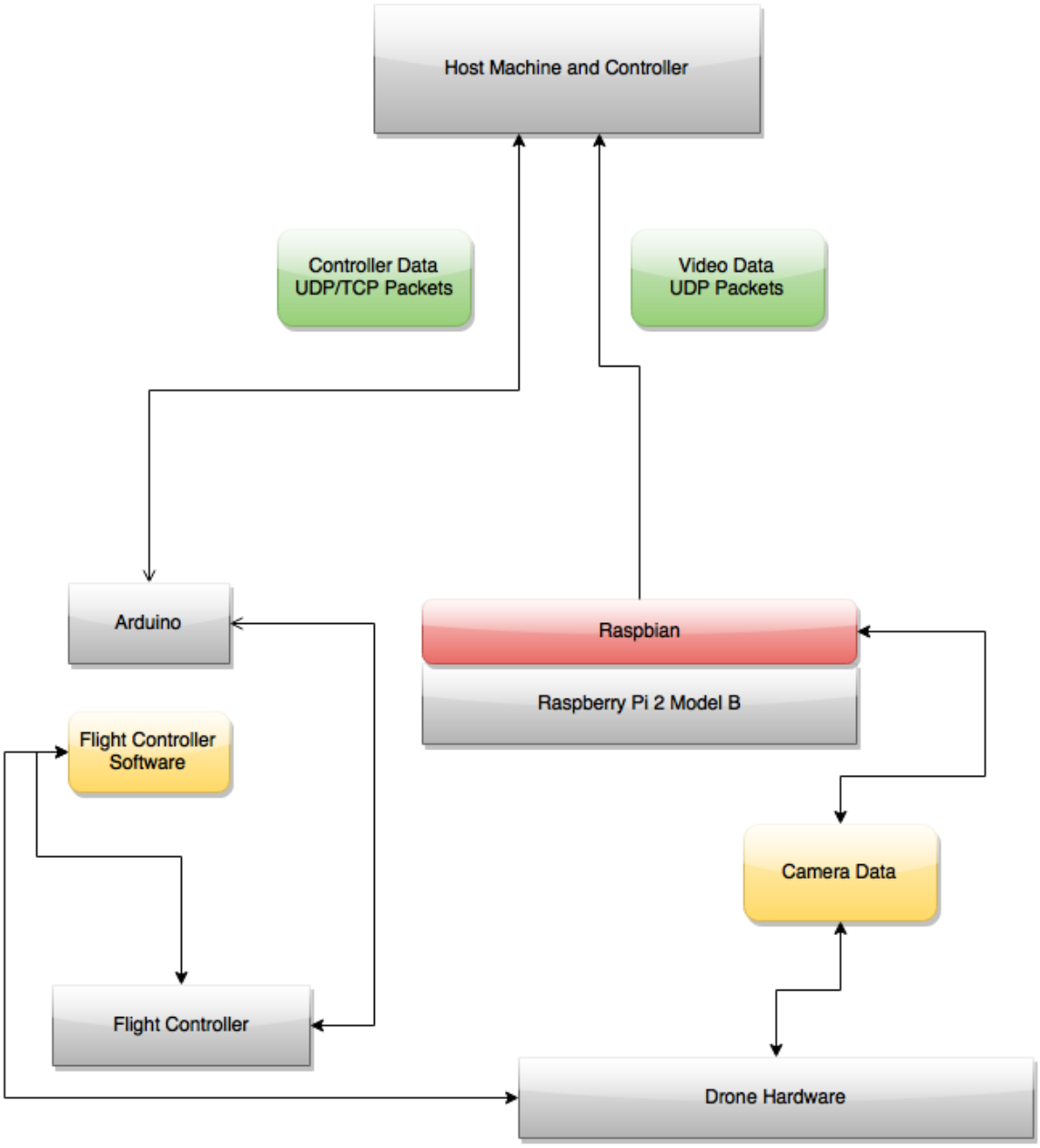


Figure 4.2F Quadcopter Software Diagram

While never implemented, the layout for a game system to be built on top of this project has already been laid out in the event of further pursuit. The purpose of the game controller software is to facilitate a game of laser tag between two users. The software is a thin layer that listens for the hardware triggers to let the quadcopter know it has been “hit”.

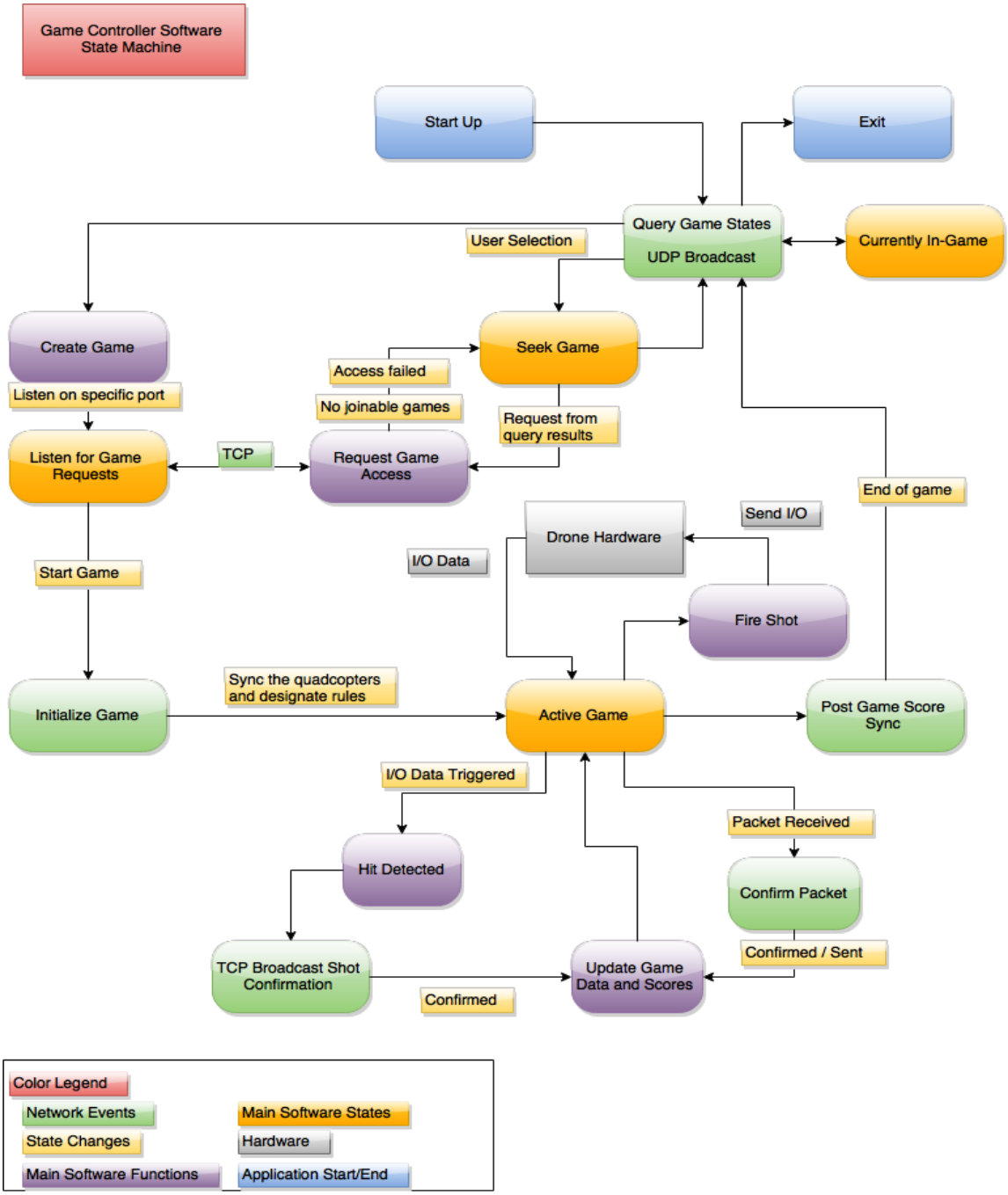


Figure 4.2G Game Controller State Diagram

The hit copter will then broadcast the hit to any other copters in the game. Each copter will update their own local copy of the scores and time remaining. Once the user wants to fire a shot an event is sent from the controller to the quadcopter. As the game finishes each quadcopter will share their individual scores and the game host's machine will discover and resolve any discrepancies.

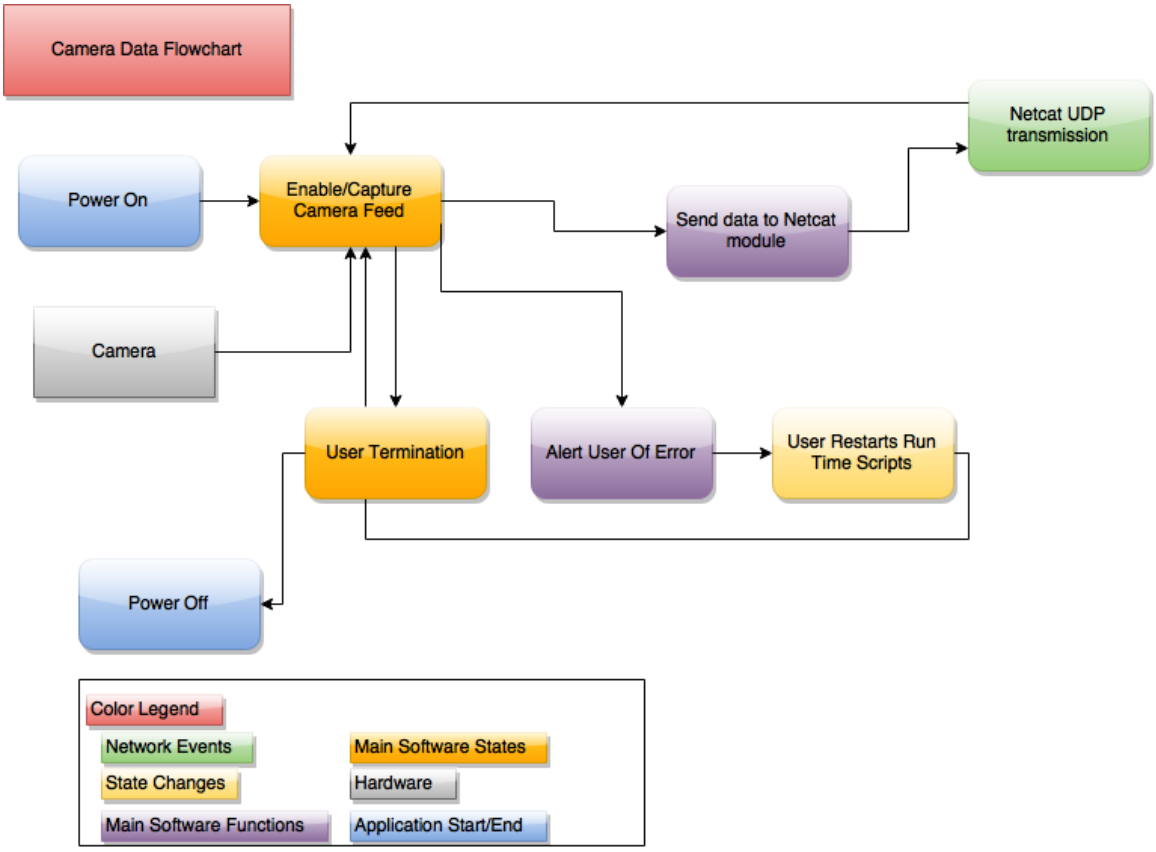


Figure 4.2H Camera Data Flowchart

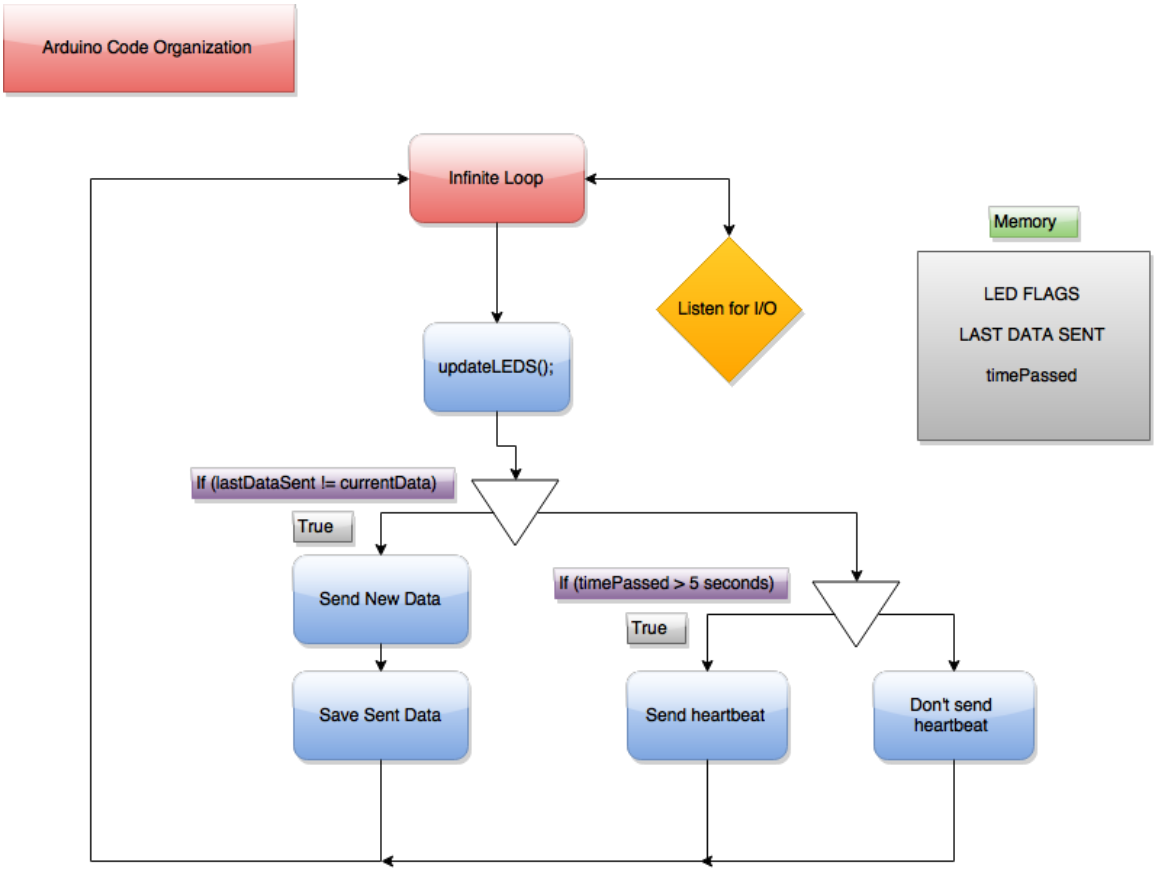


Figure 4.2I Arduino Code Organization

The Above organization is the command structure for the ATMega. These commands will dictate the outcome that the controller receives and sends to the transmitter on the copter itself.

4.4 Feature Results

For the Oculus Rift's Overlay we wanted to design an overlay that was minimal. We wanted to make sure not to obstruct the user's vision while still giving them critical data. The intentions are to make sure the color scheme for each icon is something that can be read over any texture and color combination. We felt the need that the user might find these things intrusive, so we plan on allowing the user to hide elements of the display. This can be toggled during flight, however the configuration as in which icons to hide must be done prior to starting the software. Once the quadcopter reaches a critical point of resources it will switch from the user's controls to an automatic reserve mode, in which certain icons become mandatory on the overlay.

Oculus Rift Overlay Icon Reference			
Icon Number	Name	Function	Can be hidden?
1	User Mode	Show the user the quadcopter is in manual mode	Yes
2	Mode	Represent to the user if the ORQC is in game mode or flight mode	Yes
3	Auto Mode	Quadcopter is in automatic mode	Yes, until reserve mode
4	Game Stat	Reference game points or place, if in game mode	Yes, only exists when in Game Mode
5	Wifi Connection	Show the user the strength of the wifi signal	Yes, while signal is strong
6	Power Indicator	Indict how much power the quadcopter has left	Yes, while in good state (<50%)
7	Warning Signal	Something is wrong with the quadcopter and flight should stop	Always hidden until triggered
8	Toggle UI State	Show whether the icons are hidden or not.	No
9	Crosshair	Visual aid for what is directly in front of the quadcopter.	No
10	Game Messages	Various messages from Game Mode.	No
11	Flight Space warning	Warn the user that they are too close to some object.	Always hidden until triggered
12	Thrust Percentage	Show the user the amount of work the engines are doing.	No

Figure 4.4A Oculus Rift Overlay Icon Reference

The above table gives the overlay detailing in the event an overview is applied for a laser tag addition to the game. These will include shots, a crosshair, and other various user needed data parameters.

5. Oculus Rift Dev Kit 2 Integration

The Oculus Rift is a work-in-progress virtual reality headset that is nearing completion for the consumer market. The version we will be using for this project is Oculus Rift DevKit 2. It is a fixed and modified version but, according to the developers, only half as capable and structurally sounds as the planned consumer version. This version boasts 1080 resolution for both eyes and a 75 Hz refresh rate. The options are also interchangeable within the programs it is used for.

There is a bit of setup required with the current version making it potentially the least user friendly application to the whole project. This setup includes mounting the camera, connecting it to the host machine, ensuring the user can see properly when the display fully comes on, and adjusting resolutions as needed. Tested so far in other applications Dev Kit 2 will prove to be quite valuable to the project and after the project completes further integration with the consumer release version may be applicable to the project. Below is a short diagram highlighting the conceptual interaction between the OR and the rest of our system.

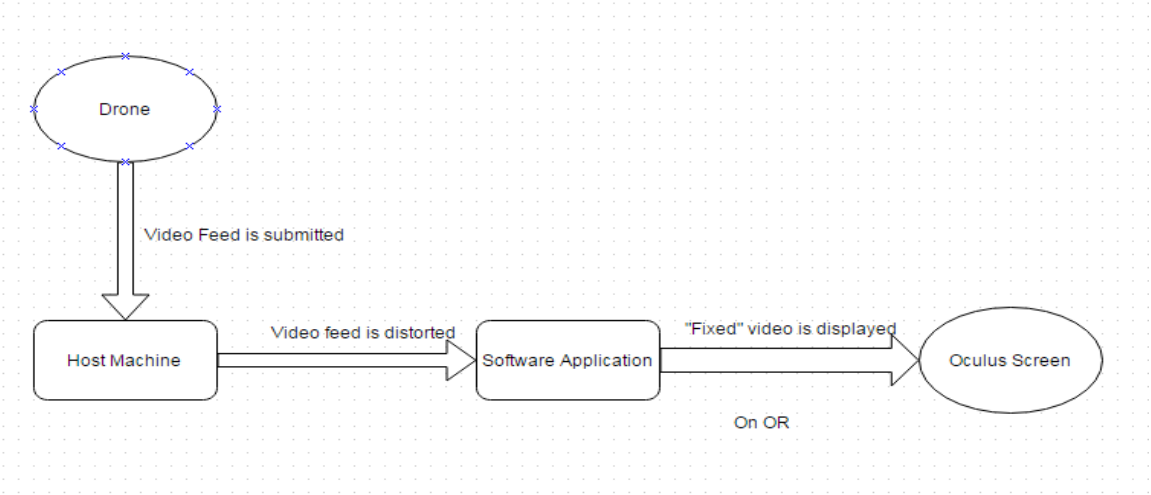


Figure 5A: Conceptual Diagram of OR integration

For the purposes of our project, we will be able to set up without requiring the external camera for positional tracking. Because our goal is just the sight and 360° orientation tracking, we have no use for the added tracking on our fixed viewpoint. The oculus will simply act as a monitor, and all testing will be done through the viewpoint of the oculus itself and a regular desktop screen, this will be to ensure that we can treat each aspect of our system as modules and do individual testing.

Because of the stretched resolution, live camera feed must first be distorted before viewing to ensure that the user can comfortably see without dizziness. Our intention is to have this post-effect on the feed be done on the software level of our graphical overlay/game engine. As well as having a setting to turn this on or off for regular monitor testing. Integrating the orientation tracking will likely be the most difficult portion of our system, but because we

are using a fixed camera feed we don't have to worry about syncing the movement with any onboard controls and can be dealt with individually. To accomplish all of the above we must research exactly how the current programs and games that support the OR communicate with the device. Because the API's are available to owners of the OR once the product code is activated on the site, the sooner we get our hands on an OR the more time we have to look into how it works. So we can make more clear decisions on the handling of the video feed and how we can incorporate it into any possible game engines or overlays and improve the experience. Future drafts will provide more concrete details as we gain experience with the hardware and software.

The data to follow is the Software Developers kit, specifications, and requirements. These will outline the full intended use and control of the Oculus for this project as well as costs and concerns related to the Rift.

5.1 Oculus Dev Kit 2 Software Developers Kit

Figure 5.1A: The components that make up the OR (pending approval from IFIXIT)

The Oculus Rift DK2 came bundled in a hefty cardboard box (A notable change from the DK1's tough plastic casing) which is likely to help cut the price. If costs for a better case were included it would likely push that $350 price tag into the $400 bracket.

Components	Details	Features
Sensors	Accelerometer/Gyroscope/ Magnetometer Near Infrared/CMOS sensor	1000hz refresh rate
Screen	Low Persistence 5.7” Super AMOLED display	960x1080 per Eye
Lenses	A set B set	For users with regular vision or contacts For users who are shortsighted

Table 5.1a: Table of various components and their details

Open developer software provided online at the developer’s main site. The total package price is listed at $350. This will likely be the most expensive portion of our project and as such we are looking into means to fund it. We are attempting to contact companies who might show interest in our project and wish to negotiate some sort of “rental” agreement for the duration of Senior Design for the purposes of testing and development. Barring that one of the group members Gustavo has acquaintances with access to an OR and will attempt to arrange to either purchase or rent the OR self funded by the group members.

5.2 Oculus Specifications and First Impressions

Notable Improvements from the DevKit 1 are the addition of Positional Tracking, Low persistence OLED display, and a built in latency tester.The control box that used to come with the DK1 that had screen brightness and other manual controls has been bundled into the head unit motherboard and can be accessed in software rather than having to access the hardware. This comes in handy as it simplifies any configuration and avoids extra peripherals that the user has to worry about while wearing the headset. After getting our hands on it it seems rather sturdy. The headset itself is durable, and doesn't seem to have any loose hanging pieces that could break.

The main HDMI cable bundled with the usb connector is covered in a thick durable plastic, no worries with wear and tear in that regard. The lense distance markers on the side are easily visible and click smoothly into place. Attempting to wear glasses while wearing the Oculus feels very cramped, and the lenses seem prone to scratching if in contact with lenses from the user's glasses for an extended period of time, considering all the head movement that goes on. The cable is very uncomfortable however, that may be an issue for user comfort and might need to be noted in our usage instructions to alleviate that, it wraps

behind the head and gets easily tangled with all the extra cables and wires required for the set up.

After downloading the supported software and drivers however, the device is easily setup and straightforward to use. The lenses are easily removable and replaceable, as they simply twist into a socket. Once the lenses are removed you can see the Note 3 screen fixed into place, with a partition in between to help further distinguish the boundaries between each eye's view, similar to binoculars.

The screen is easy to reach for cleaning. The positional tracking camera is lightweight and easy to carry around, however between the positional tracking camera and the oculus itself any machine using this device requires a minimum of 4 usb ports, and this will have to be put into consideration if we want to utilize a raspberry Pi as a host machine, due to limited ports.

5.2.1 OLED Display

960x1080 (Per eye), full housing from the Samsung S3 including logo. Straight rip from the product, this allows for manageable replacements should the need arise. Screen can be treated as a single monitor on “Direct Display” mode available in configuration software, however this is disorienting and does not appear fluid. Non-Stereoscopic video causes issues varying between misalignment and stretching and tearing, as the oculus tries to display on each half of the screen.

5.2.2 Sensors

Head positional tracker: Infrared Camera.Mounts easily onto a computer monitor, comes with clips and a flat platform that we could possibly attach velcro or magnets (once we are sure it's safe to do so) to help mount onto more surfaces, powered by usb connection and needs to be connected to PC to ensure we have access to all of the Oculus Rift’s tracking capabilities. We can probably remove this component, if we find the built in orientational tracking is sufficient for our purposes.

Infrared LED Array: Housed in the head mounted casing, is an array of IR LEDs used by the Positional Tracking camera.

5.3 Host Machine Specifications To Run The Oculus Rift

Minimum power and host machine requirements as designated by Oculus are listed in this section. The minimum specifications are the base line and lowest quality recommended to run on a machine with the oculus rift.

5.3.1 Host Machine Minimum and Recommended Requirements

Minimum requirements: A desktop computer running Windows 7 or Windows 8, 2 USB 2.0 ports (at least one powered), and a dedicated DirectX 11 compatible graphics card with DVI-D or HDMI graphics output.

Recommended specifications: In addition to the minimum spec it is recommended that your dedicated graphics card be capable of running current generation 3D games at 1080p resolution at 75 fps or higher. A consistent 60fps is recommended for average use to ensure that the image quality stays smooth and user feels minimal delay in tracking functions.

These specifications may be slightly altered or updated as better versions and materials become available during the progress of the project.

5.3.3 Power

Powered USB port minimum (preferably 3.0), but possible to be powered by a regular 5V battery as long as it uses USB connection. This will help to limit any further required power distribution to split to other components. This also means that it can be powered by a larger motherboard usb source or general power supply.

5.3.4 Oculus as a Display

With the Raspberry Pis running an image of Linux, the Oculus Rift SDK unfortunately no longer works, as support has been pulled. So since the Oculus is in essence a display with some extra sensors we found that we would use the Oculus as a display and a placeholder for better supported VR headsets. With this special configurations need to be set up on the Raspberry Pi's boot process in order to get an optimal picture out of the Oculus Rift.

There were challenges getting the Raspberry Pi to output an HDMI signal that was sufficient. Specifically the RPi needed to be configured to ignore the Extended Display Identification Data (EDID) so the Pi could recognize the display. In the end the Oculus' stereoscopic view unfortunately made for an extremely hard to see picture. This presented us with a new design decision if we planned to expand upon the scope of the project. That being to either use a display similar to that of Google Cardboard or a more fully-featured competitor VR headset.

6. Quadcopter Design

The purpose to the quadcopter design is to have a simple Quadcopter that will allow for a decent flight time as well as payload enough to lift with a camera mounted to it. This section will give a layout to preferences between current quadcopter designs and ideas. There will also be information of a separate chosen quadcopter for comparison.

6.1 Quadcopter Architecture

We are currently supplied with the Rotor Concept HPQ1 Quadcopter in Figure 6.1A

Figure 6.1A The Rotor Concept HPQ1 Quadcopter we are supplied with.

We may also use the Traxxas 6608 LaTrax Alias if our supplied Quadcopter does not meet the requirements we need to continue with the project. The Traxxas 6608 is shown in Figure 6.1B below, consent given by manufacturer.

The quadcopter being used will take a fair amount of coordination to comply with the rest of the project. To start, we will examine the frame to make sure it is strong since the arms of a quadcopter are the parts that are most likely to break and experimenting with the quadcopter can become expensive if we have to continuously replace a pricey frame. We will need to observe the motors to see if it can handle the weight of the quadcopter and the attached camera while giving a smooth ride. The smoothness should not be overlooked especially since the oculus can already be disorienting to those without prior experience.

We can look over the speed controllers of each motor to make sure they are up to par, thankfully the hobby community has progressed and these devices should be more and more readily available as time goes on. Propellers will be needed in plenty since these are also prone to breaking and are relatively cheap.

Figure 6.1B: Traxxas 6608 Quadcopter and its dimensions.

We may replace the battery installed in the quadcopter since we want flight time to be larger than the common ready-made quadcopter's flight time of seven or so minutes. This may be one of the leading expenses surrounding the quadcopter. The next largest expense may be the radio system. Since we want to use the glove to communicate as opposed to a more common transmitter controller, this price may fall with the research of what we can attach to the PCB. The common flight controller for the quadcopter itself is about the same price range as a raspberry pi, so using one to combine Wi-Fi and quadcopter controls will not be monetarily costly. The other spare not-as-flashy bits and bobs such as servo leads, low voltage alarms, and breakout cables should just about exceed the price of a single frame.

6.2 Quadcopter Components

Due to issues with the HPQ1 and the amount of time it would have taken to receive the Traxxas we were forced to use an alternative model of quadcopter on short notice. This quadcopter is the X525. This was a low budget alternative we found that could meet most

of our requirements. The Raspberry Pi, however, is still not part of the original design of the X525, so that will have to be added to the quadcopter. The WiFi adapter will also be added and the battery may be switched out as examined later on.

Figure 6.2A X525 HobbyPower Quadcopter Kit

6.2.1 Flight Controller

The flight controller that comes with the HPQ1 is the Rotor Concept TX7 Transmitter. It is characteristically similar to the Spring RC TG661A six-channel spread. It operates on a 2.4GHz to 2.483 GHz frequency. Power supply is 9.6V to 12V equating from 8 AA batteries. It has a FSK modulation and its receiver type is a RG661A. All of this will be extremely important in determining how to communicate between the glove and the quadcopter, itself. The Spring RC TG661A is about $42.33.

The flight controller that comes with the Traxxas 6608 LaTrax Alias is the Traxxis LaTrax Alias Quadcopter Transmitter with 2.4GHz frequency and 6 channels. The power supply is four AA batteries totalling roughly 4000mAh and 12V. The current pricing on the

Traxxas LaTrax Alias Transmitter is on sale for $4.99. It's regular pricing rests at $62.99. Because both the HPQ1 and the Traxxas 6608 transmitters operate on the same frequency with the same channels, it is assumed that the alternate transmitter for the HPQ1 can be used for the Traxxas 6608.

The Flight Controller included with the X525 copter is the KK2.15 Flight Controller, a low cost flight controller ($26). It operates on a similar frequency to most generic flight controllers and accepts a wide range of common RX/TX pairs. The RX/TX combo will be simulated by our Arduino Uno system.

6.2.2 Propellers

The HPQ1 is equipped with LotusRC "8A" and "8B" quadcopter propellers of approximate size 8 inch length and 3.8 inch pitch. Similar propellers are the Gaui 8" quadcopter propellers that are about a third of the price of the previously stated propellers. They have a length of eight inches and a pitch of three and four fifths inches. Common pitch levels range from 3.5 to 6 inches, so 3.8 will keep us on the low side without being the minimum. In simplified terms, the higher the pitch, the slower the rotation and vehicle speed. This concept is shown in figure (2) from D.L. Engineering's website.

This increase of speed will, of course, increase the power used. Also, the generally lower the pitch number, the higher the torque the propeller can produce. Since we will need to carry more weight than the usual quadcopter, we will want to stay close to the minimum of 3.5 inches but reach just a bit higher to avoid such high torque.. [1] "For larger quadcopters that carry payloads, large propellers and low-kv motors tend to work better. These have more rotational momentum, and will more easily maintain your aircraft's stability."

Quadcopter propellers come in a set of four with two of them being for clockwise rotation and the other two for counterclockwise rotation. Picking up multiple will be very heavily needed since propellers are reported to break often when experimenting. The HPQ1 Quadcopter is advertized so that [2] "use of the optional three-bladed propellers allow a payload of 1.2 pounds." Since we will need to carry a lot of weight, we will opt to have the three-leafed propellers with the length of 8 inches and pitch of 4.5 inches. A set of clockwise and counterclockwise three-leafed propellers will cost $2.50. A set of propellers for $5 is very good, so multiple sets will be bought due to propeller fragility.

The Traxxas 6608 has a Propeller length of 5.51 inches. The price for a pair of propellers from the manufacturer is $2.00. These, however, are two-leaved propellers. Thus, if we were to use the Traxxas 6608, we would use 5 inch long and 3 inch pitch three-leafed propellers. A set of four is about $4.99.

The Propellers on the X525 are the 1045R Propeller models. A replacement pair of these costs us around $5 Dollars for a pack of 4.

6.2.3 Motors

The HPQ1 has four 980Kv brushless outrunners with removable “endbells” for storage. The brushless outrunner motor is an electric motor that spins its outer shell around its windings. Though outrunners are slower than inrunner motors, they have a much higher production of torque. This higher torque output will coincide well with the small pitch of our propellers. this combination can lower the amount of power needed to power the motors, as stated above in the description of propeller pitch. Using an outrunner also eliminates the extra weight and complexity of a gearbox.

The Traxxas 6608 has two high output clockwise and two high output counter clockwise motors. They are brushed and coreless. From the manufacturer, the set of four motors costs roughly $40 while independent sellers sell the set as cheaply as $27 online.

The X525 Motors are the KV1000 XXD2212 brushless motors. Slightly weaker than both the HPQ1 and Traxxas 6608 but enough to suit our needs. A replacement pair is as cheap as $8 for a pack of 4 from amazon.

6.2.4 Battery

The battery equipped on the HPQ1 Quadcopter is a Generic 2400mAh 20C 3S lithium polymer with Deans Ultra-Plug connector and JST-XH balancing harness. It has a claimed flight time of eighteen to twenty five minutes, which would be phenomenal. This is, however, without streaming the live video. This will, undoubtedly drain the battery.This battery weighs about 0.7 pounds.

The Traxxas 6608 has a standard 650mAh capacity LiPo battery with 3.7V voltage equipped with it. It’s flight time is a boasted average 10 minutes without weight and while using the two-leafed propellers. Below is a table containing different battery choices. Because we are looking at two different quadcopters, we will have two different tables. The first table is for the HPQ1 and the second table is for the Traxxas 6608.

HPQ1	Standard	Turnigy 5000mAh 4S 30C Lipo Pack	Multistar High Capacity 3S 4000mAh Multi-Rotor Lipo Pack
Voltage	3.2V	3.7V (per cell)	3.7V(per cell)
Amp Hours	2200mAh	5000mAh	4000mAh
Discharge Rate	20C	30C	20C
Weight	180 grams	556 grams	244 grams
Cost	$69.95	$37.39	$19.96

Figure 6.2A :Table comparing different available batteries for the HPQ1.

Traxxas 6608	Standard	ZOP 3.7V 1000mAh 20C Lipo Battery	Syma X5C-1 X5C X5A
Voltage	3.7V	3.7V	3.7V
Amp Hours	650mAh	1000mAh	680mAh
Disharge Rate	20C	20C	20C
Weight	8.5 grams	35 grams	14.17 grams
Cost	$10.56	$9.98	$14.00

Figure 6.2B: Table comparing different available batteries for the Traxxas 6608.

If the flight time turns out to be less than satisfactory, then we will be switching to the Multistar High Capacity 3S 4000mAh Multi-Rotor Lipo Pack. It's ampere output is much higher than the standard battery while being much lighter than the Turnigy. This battery weighs about 0.538 pounds. If the advertizing for the quadcopter is to be believed, then this difference in about half a pound should not alter our plans too much.

The battery we decided to use on the X525 was a Turnigy 2200mAh Li-Po battery pack, weighing at about 85 grams and supplying a 12v total. This gave us about a 8-10minute flight time in testing.

6.2.5 Frame/Landing Gear

The HPQ1 has an aluminum frame for the arms and skid bracket with carbon fiber skids. There is a clear polycarbonate dome with black and yellow shrink wrap tubing to discern between the front and the rear of the quadcopter. The entirety of the quadcopter as it is out of the box is about .95 pounds while empty. When flight ready, that is when a battery is added, it is 1.3 pounds. And when loaded to maximum, it is advertized to be able to fly at 2.9 pounds.

The Traxxas 6608 has a molded composite frame. The frame sells online for $8.11 and at full price for $13.99. This can be cheap enough to buy multiple and not have to worry about breaking the quadcopter during prototype testing.

6.2.6 Camera

The camera used to communicate with the host machine through WiFi will be the raspberry pi-specific camera board. The camera has more resolution than the Oculus Rift, so the plan of using the camera's picture as a large image and using the Oculus Rift's smaller resolution to look around will theoretically work, the supported camera resolutions that will likely interest us the most are 1080p30 and 720p60. The camera should not take a large

amount of wattage to power so this will keep battery use to a minimum. Another benefit to this camera is that it weighs about 0.01 pounds. This will also help to reduce overall weight consumption

6.2.7 Raspberry Pi

It weighs an approximate 0.1 pounds, which definitely falls in the realm of possibility for the HPQ1 Quadcopter to be able to carry. We will be attaching the Raspberry Pi to its own power source. We are thinking of using the USB Battery Pack for Raspberry Pi - 4400mAh - 5V @ 1A which comes out to be about 0.3 pounds. The Raspberry Pi itself and the battery together turn out to be about 0.4 pounds. This being under half the maximum advertised weight of 1.2 pounds makes it perfectly acceptable. The Raspberry Pi has several options available to it for cameras and WiFi adapters. Because of its operating system being on a open source, there are plenty of user-based inquiries and information available to us for reference. As such, there are devices made especially for the Raspberry Pi that will make this project smoother due to their efficient integration.

6.2.8 WiFi Transmitter Adapter

There are a few choices we have between Wifi Adapters, however the cost range of these usually are only $5~10 so it is quite possible we will pursue testing with a series of Adapters to achieve our best latency. There are a number of USB Wifi Adapters specifically marketed towards the Raspberry Pi which have low power requirements that will assist in organizing and planning the power consumption required towards the cameras and peripherals we will be required to attach to our Pi. The differences between the adapters are minor and do not require extensive detailing, most offer 802.11 functionality, powered by USB 2.0 and different transmission speeds and channel settings. For the purposes of early prototyping we will likely be using the RPi Wifi Adapter from ThePiHut, as its $8 cost is easily replaceable and support for Linux and Windows will let us begin testing quickly to ensure we have our camera feed setup ready to go.

Our final choice was the Edimax EW-7811Un 150 Mpbs 11n Wi-Fi USB adapter, as it was highly recommended within the Raspberry Pi community. The size was very compact and fit well within the Raspberry Pis, only extending out from the USB port by approximately 5mm. Additionally the price was comparable to the adapter ThePiHut offered (10$ vs 8$) however the shipping made the Edimax ultimately the cheaper option.

6.2.9 Power Distribution System

Power distribution systems in the hobby world are simply PCBs with color coordinated plugs for the easier consumer use. The Rotor Concept website does not have readily available specifications for their HPQ1 and neither does the LaTrax website. Thus, it is assumed that a power distribution board is used for its efficiency. An example of the board's layout is shown in Figure 6.2C.

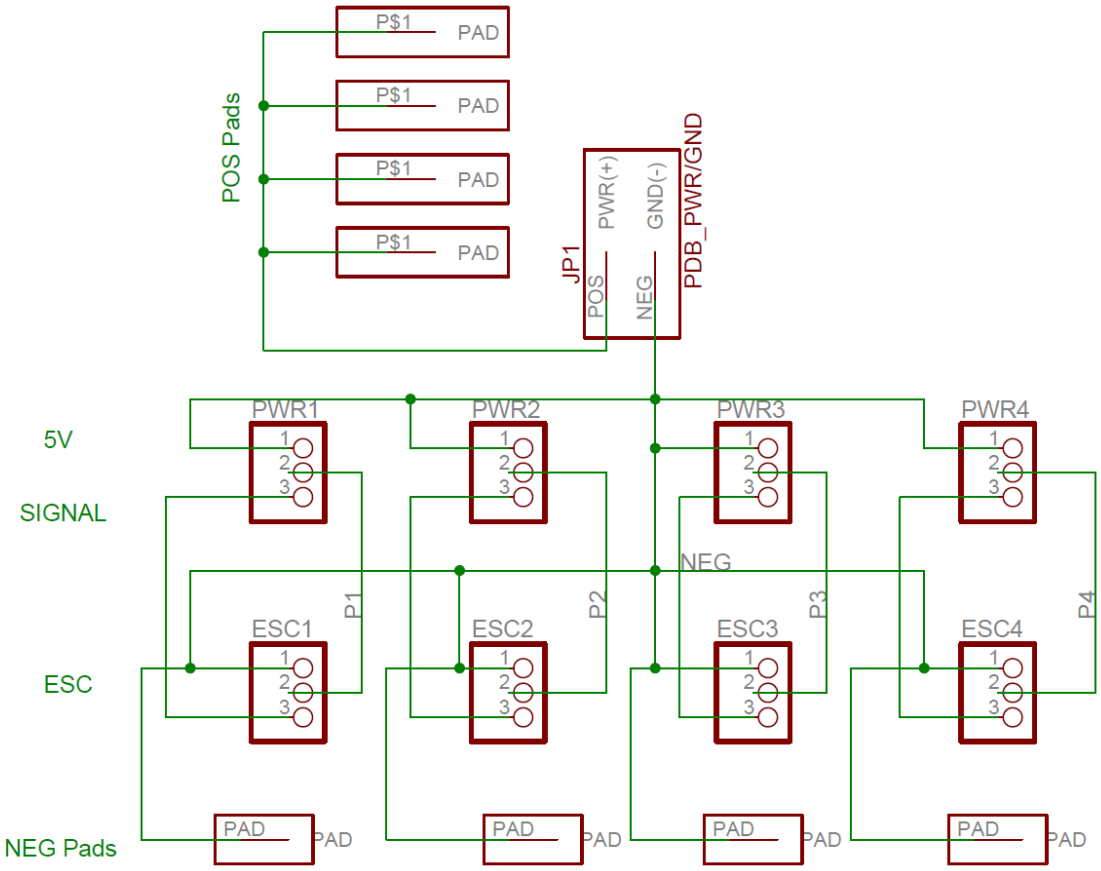


Figure 6.2C: Standard Power Distribution Board layout.

The above Figure 6.2C shows the full outline of the power distribution board pertaining to its use on the copter.

6.3 Quadcopter Component Summary

Weighing in the differences to the Traxxas 6608 and the HPQ1 we can see in Figure x that the flight time for the HPQ1 is longer by about 10 minutes with a heavier frame and taller size. The traxxas on the other hand is smaller and lighter with a molded composite structure and lighter LiPo battery pack. The radio system also allows for 2 more channels for wireless control.

There are many factors to take in with the copters as we choose through them. Key points being weight and if any other features are needed to assist in their flight. The specifications we came up with are placed in the chart on the following page for the HPQ1 and the Traxxas.

Below is a chart of the summary of the information discovered in this section

Specifications	HPQ1	Traxxas 6608	Hobby King X525
Blade length	8 inches	About 6 inches	8 inches
Quadcopter Width	24 inches	12.07 inches	24 inches
Overall Height	5.5 inche	1.69 inches	4 inches
Flight Weight with battery	610 grams	100 grams	583g
Flight System	3-axis gyro and inertia based self-stabilization	Auto-leveling six-axis	KK 210.15
Flight time (maximum)	25 minutes	15 minutes	12 minutes
Radio System	2.4GHz digital spectrum. (also any 4-channel radio)	2.4GHz 6-channel multi-mode	2.4GHz 6-channel multi-mode
Main Frame Structure/Material	Aluminum	Molded composite	Glass Fiber
Battery Type	Lithium Polymer (LiPo)	Lithium Polymer (LiPo)	Lithium Polymer (LiPo)
Battery Voltage	3.2V	3.7V	3.7V
Battery Capacity	2200mAh	650mAh	2200mAh
Battery Discharge Rating	20C	20C	20C

Figure 6.3A: Chart comparing our two choices of quadcopter.

For our specific uses the longer time frame for flight will allow for longer play with the controller and headset. They both also keep under what we would warrant as a safe boundary for eye use of the Oculus. When testing becomes more readily available of the two units we can be more precise on the most beneficial Quadcopter to use based on our criteria of flight time and responsiveness.

7. Glove Controller Design

The controller of the drone will be the Glove controller. It will allow the user to manipulate the drone with his/her hand with movement of the hand in multidirectional motions. Utilizing a gyroscope and accelerometer we will be able to accurately depict the motion of the hand and an intended direction for the quadcopter to move. The intent and purpose of this device is to have a Joystick like feel that is more immersive and responsive in nature than a joystick can offer.

Further implementation of the glove may allow the user to utilize other interfaces rather than just the drone like flight simulator applications or other games. This will ultimately depend on the driver software developed for it and the connection point to the desired host machine.

The goal of the glove is to make an intuitive controller that the user will be able to use without much training. Because of the possible goal of integrating the quadcopter-Oculus Rift combination in a laser tag environment, there will be a limited amount of time for each user and there will be multiple users wanting to play with the product. Thus, the easier it is for the user to learn and start playing with the product, the more appealing it will be thus creating more customers and more users. We also want the glove to be as natural and non-juxtapose to using the Oculus Rift as possible. We want the users to be able to feel as if they are flying the quadcopter, themselves, as opposed to having a screen close to their eyes while still using a conventional controller. We want it to be a seamless experience as opposed to a flashy activity.

Ideal functionality of the glove is to have it accurately represent the movements of the user with the movements of the quadcopter. So as the user raises their hand, the quadcopter will raise from the ground. As the user turns their hand, the quadcopter will turn. And so on. There will also ideally be buttons that act as shortcuts to taring. When the user places their hand on the ground and hits the “tare” button, there will be an automatic reset so that the glove’s RC component will translate as the glove being on the floor and thus the motors of the quadcopter will be off. There will also be buttons for setting power distribution amongst the motors in case there will be listing by the quadcopter. What we mean by “listing” is the quadcopter moving to the left, right, forward, or backward unwantedly. This is a common occurrence with quadcopters and thus buttons to correct such actions would be preferable.

7.1 Glove Architecture

The glove will be a simple store bought one-size-fits-most glove with the fabricated PCB board attached to the back of the glove to rest on the back of the user’s hand. The glove will be a fingerless leather glove. The lack of fingers on the glove will be for ventilation so that the user’s hand will not excessively sweat. Leather being the choice of material is for some insulation between possible heat coming off of the PCB and the user’s hand. Because the user will be seated to use the Oculus Rift, a chair’s arm rest or a nearby table will be an easy tool to use for setting the glove’s sensors at a (0, 0, 0) position. Alternatively, if the

user is sitting on the floor, they can use the ground to set the sensors at (0, 0, 0). The user will be seated in such a way that the wires connecting the glove and the host machine will not get in the way of the user's motions. The prototype will have a small breadboard attached to the back with an microcontroller board and the Adafruit 10-DOF IMU Breakout - L3GD20H + LSM303 + BMP180 breakout board connected and ultimately wired to the host machine.

7.2 Glove Components

The glove will have to have at least four key components: the glove itself, and Accelerometer and Gyroscope combination, USB input and output, and at least one button. The glove itself will act only as a vessel for the PCB so that strapping it onto the user's hand can be easy.

The accelerometer and gyroscope combination is going to be the main focus of the glove. We are going to have to read tilt with a good amount of accuracy so that the user can have tight control over the movement of the quadcopter. For example, we need the glove to sense the nuances between slight and major tilt so that the slight tilt left or right can be a small drift left or right and then a large tilt will cause the Quadcopter to harshly fly left or right.

The barometer was intended to be used as an altimeter in case the accelerometer could not comprehend the Z axis motion as hoped. In the end, neither the barometer nor the accelerometer would work as expected and the quadcopter was given a certain amount of thrust as defined by the computer.

The USB input and output is to be used simply as the communicator between the movements of the accelerometer/gyroscope combination and the host machine. The USB input will also have to act as the incoming wattage to power the glove.

7.2.1 Accelerometer and Gyroscope

The object of the glove is to measure the tilt of the user's hand and translate that data to determine direction of flight and acceleration in said direction. To do this, we will need to combine an accelerometer and a gyroscope to accurately measure the tilt. Given that we master the tilt, we may be able to add additional features for tighter, more intuitive controls. We will be using the Adafruit 10-DOF IMU Breakout - L3GD20H + LSM303 + BMP180 breakout board to test with an Arduino to work out any kinks the programming or parts may have. Adafruit claims to have a "3 volt regulator with reverse-polarity protection" so that we don't have to be worried about burning any of the components and we can test power outputs with more freedom. Once mastered, we can move on to integrating the individual parts (the L3GD20H gyroscope and the LSM303 compass and accelerometer combination) into the printed circuit board. Both components have I^2C digital output, so communication should be relatively easy.

The LSM303 has 16-bit data input, an analog supply voltage of 2.16 volts to 3.6 volts. It boasts tilt-compensated compasses, position detection, motion-activated functions, and power-saving for handheld devices. Since this will be mounted onto the glove, this last feature will be unnecessary. Figure (5) shows the block diagram as provided by the STMicroelectronics data sheet. Below the figure is a table describing the pin layout of the LSM303.

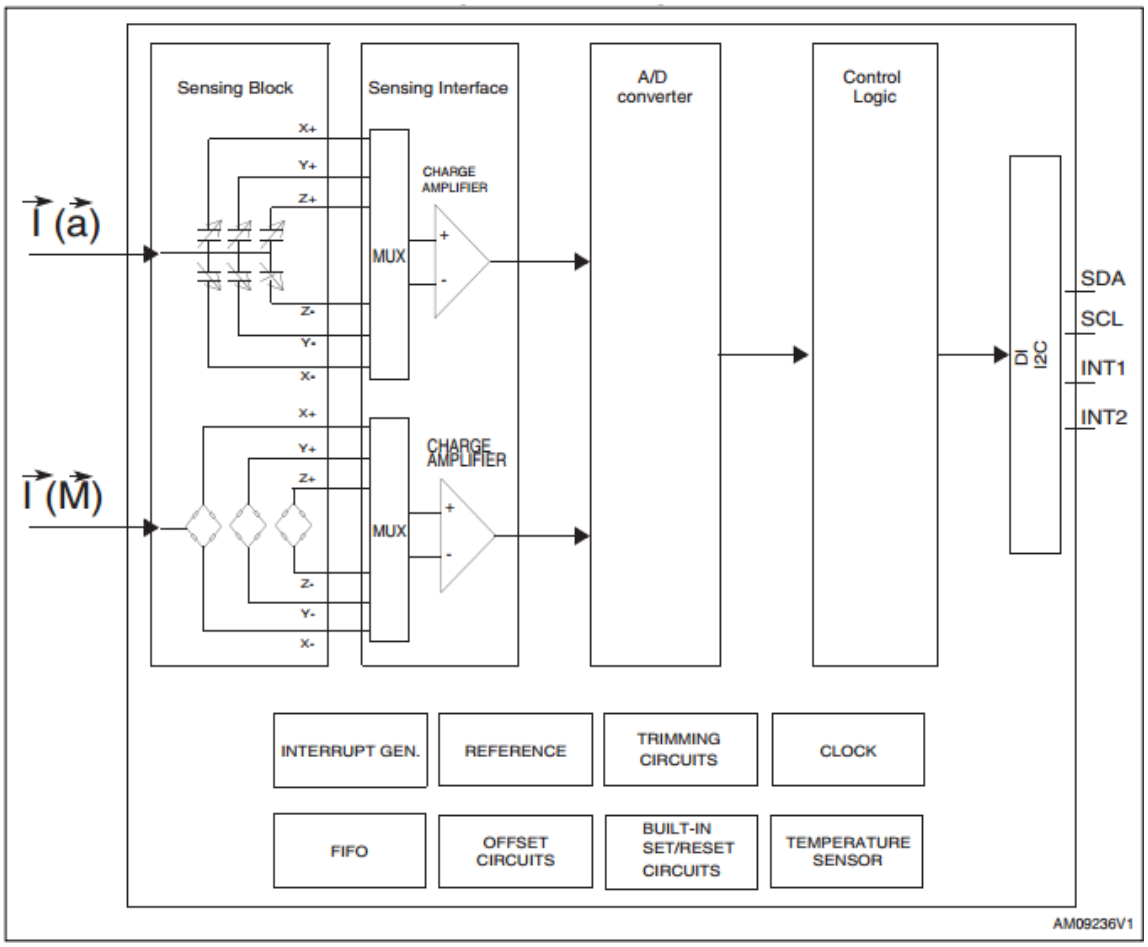


Figure 7.2A. LSM303 general flowchart given by its datasheet.

The above table will mainly be used when creating the printed circuit board. As shown above, pins 7, 10, and 11 will be connected to ground. Pins 1 and 14 will be connected to the microcontroller's power source. The rest of the pins will be connected to either the microcontroller or separate devices depending on what is needed by the accelerometer. We will be using the datasheet and any other supplied resources to combine our knowledge and get the PCB working.

Pin #	Name	Function
1	Vdd_IO	Power supply for I/O pins
2	SCL	Signal interface I2C serial clock (SCL)
3	SDA	Signal interface I2C serial data (SDA)
4	INT2	Internal interrupt 2
5	INT1	Internal interrupt 1
6	C1	Reserved capacitor connection (C1)
7	GND	0V supply
8	Reserved	Leave unconnected
9	DRDY	Data ready
10	Reserved	Connect to GND
11	Reserved	Connect to GND
12	SETP	S/R capacitor connection (C2)
13	SETC	S/R capacitor connection (C2)
14	Vdd	Power supply

Figure 7.2B Pin Layout of LSM303. Will be used when creating PCB.

The L3GD20H also has 16 bit rate value data output with embedded 32 levels of 16 bit data output FIFO. It has a wide supply voltage from 2.2 volts to 3.6 volts. It boasts low power consumption, fast turn-on and wake-up and high shock survivability. The datasheet supplied by STMicroelectronics lists “gaming and virtual reality input devices” as one of the applications, which gives us comfort in knowing that this route has been tested with this specific equipment. Figure (6) shows the block diagram as provided by the STMicroelectronics data sheet. Below the figure is a table describing the pin layout of the L3GD20H.

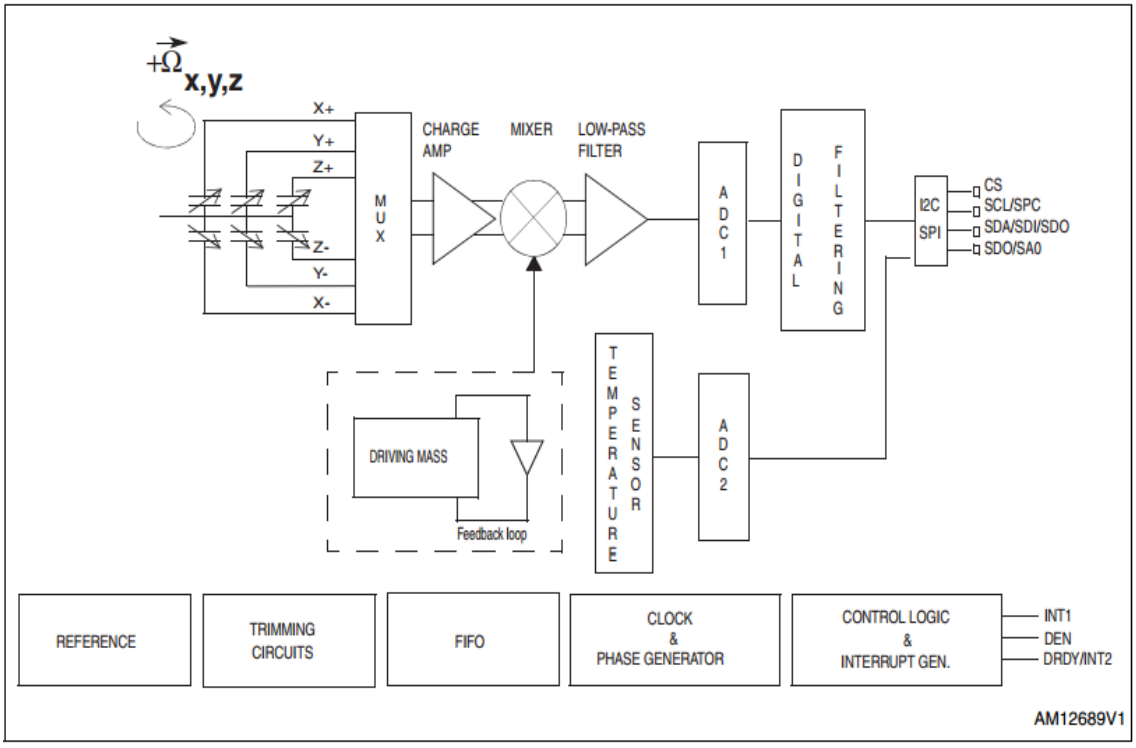


Figure 7.2C: L3GD20H general flowchart as given by its datasheet.

The Figure 7.2D will largely be for the benefit of when we start designing and creating the printed circuit board. We will be sure to follow the given materials on how to create a basic printed circuit board along with following this table and any additional information that is given by the manufacturer on their data sheets. As it stands, we know we will be connecting pins 9 through 15 to ground and 16 will be connected to the microcontroller's voltage output. Largely, we will be working with pins 1 through 8 to establish a secure data connection between the microcontroller and the gyroscope.

Pin #	**Name**	**Function**
1	Vdd_IO	Power supply for I/O pins
2	SCL SPC	I2C serial clock (SCL) SPI serial port clock (SPC)
3	SDA SDI SDO	I2C serial data (SDA) SPI serial data input (SDI) 3-wire interface serial data output (SDO)
4	SDO SA0	SPI serial data output (SDO) I2C less significant bit of the device address (SA0)
5	CS	I2C/SPI mode selection (1: SPI idle mode/I2C

		communication enabled; 0: SPI communication mode/I2C disabled)
6	DRDY/INT2	Data read/fifo interrupt IFIFO theshold/overrun/empty)
7	INT1	Programmable interrupt
8	DEN	Gyroscope data enable
9	Reserved	Connect to GND
10	Reserved	Connect to GND
11	Reserved	Connect to GND or VDD
12	GND	0V supply
13	GND	0V supply
14	Cap	Connect to GND with ceramic capacitor
15	Reserved	Connect to GND or VDD
16	Vdd	Power supply

Figure7.2D: Pin layout of the L3GD20H. For use in creating the PCB.

7.2.2 Arduino Uno

We will be using the Arduino Uno to test the Adafruit accelerometer/gyroscope combination. We will then use the Arduino microcontroller to create the final glove design. The following is a table of the specifications of the Arduino Uno as boasted by the official Arduino website.

The table provided here will be used as a reference to make sure that everything is working as it should. During the course of creating the program that will read in data from the accelerometer and gyroscope combination, we predict that the code will be inefficient on the first couple of iterations. Thus, knowing the processor speed and consequentially the amount of reaction time that the microcontroller should be operating at will give us a good indicator as to whether or not our code needs debugging. Similarly, if our code is up to par with the processing speed, then it will be an indicator that our accelerometer or gyroscope may not be functioning as they should be.

Microcontroller	ATmega328
Operating Voltage	5v
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40mA
DC Current for 3.3V Pin	50mA
Flash Memory	32KB (ATmega328) of which 0.5KB used by bootloader
SRAM	2KB (ATmega328)
EEPROM	1KB (ATmega328)
Clock Speed	16MHZ
Length	68.6mm
Width	53.4mm
Weight	25g

Figure 7.2E: Arduino microcontroller specifications for planning program boundaries

The size and weight of the Arduino will be helpful for us in determining how well the glove should work. It will give us predictions as to whether or not the glove will be a burden to the user or if the glove will work as hoped and become an integration into the virtual reality world we will provide.

7.2.3 USB Connector

We will be using the four-pin USB B 2.0 connector on this design. The USB 2.0 can handle an effective throughput signaling rate of 280 megabits per second. This should be more than enough to handle the data coming out of the accelerometer and gyroscope combination. The USB B has a lower insertion-removal cycle than either the micro or the mini USB, but since we are designing the glove to be more or less an extension of the host machine, the glove should not see a large amount of insertion-removal cycles. A standard USB will supply 5 volts, which will also be plenty enough power to handle the 3.6 volt maximum of both accelerometer and gyroscope.

7.2.4 Button Layout

Since we are designing a virtual reality where you control a quadcopter and, given enough time, try to shoot down your opponent's quadcopter, we will need a shoot button. Thus, we will be putting a button on the index finger part of the glove where it is easily accessible to the user's thumb.

The idea being that the user is making the classic childhood "finger is the barrel and thumb is the hammer" sign with their hand to mimic holding a gun. The user will hold their hand sideways in this position and when they want to fire at their opponent, they hold their finger out as the barrel and bring their thumb down on their finger as if it were the hammer of a gun striking gunpowder. This design is hopefully going to add to the lightheartedness of the design and possibly help with integration.

We may also add a few more buttons onto the printed circuit board, itself. One button may act as a kind of "kill switch." If something goes wrong, and the quadcopter starts to go haywire, the button will cause the glove to cease communication with the quadcopter and the copter will stop in midair and slowly descend. This button may be integrated as more of a switch so that the user can clearly identify the "on" or "off" position. This will be needed in a more physical way as opposed to visual because the user will be wearing the Oculus Rift, which cuts off visual contact with the world around them.

Another button to be added will depend on the integration of the accelerometer and gyroscope combination. We can add a button that will act in the same way a scale may tare weight. The button will establish a new base point where the x, y, and z axis will read at zero and there will be no sense of tilt. This button would be integrated for the comfort of the user.

As an end result, the glove only had the reset button which was used to timeout the connection between the glove and copter. When the button was pushed while controlling the copter, the copter would slowly power down until thrust reached zero.

7.2.5 Driver Software

Currently planned to try and integrate the received signals as simple keyboard inputs to mimic the same response a keyboard would give. The Arduino Uno comes with a driver creation software to make input into other devices simple and programmable. As buttons and functions are added we can convert them direct as keyboard commands which will allow for faster data flow and response time from the accelerometer and gyroscope as well as the physical buttons themselves. The glove on/off disconnect will be a manual switch for single interrupt and not be part of the driver. It will halt the data flow from being connected to the host machine but cutting off the power separately. All other software provided by the Arduino Uno is open source with many guides, references and tutorials. Further production with the Arduino software will be pending till the board is received and attached to the glove.

7.2.6 Print Circuit Board

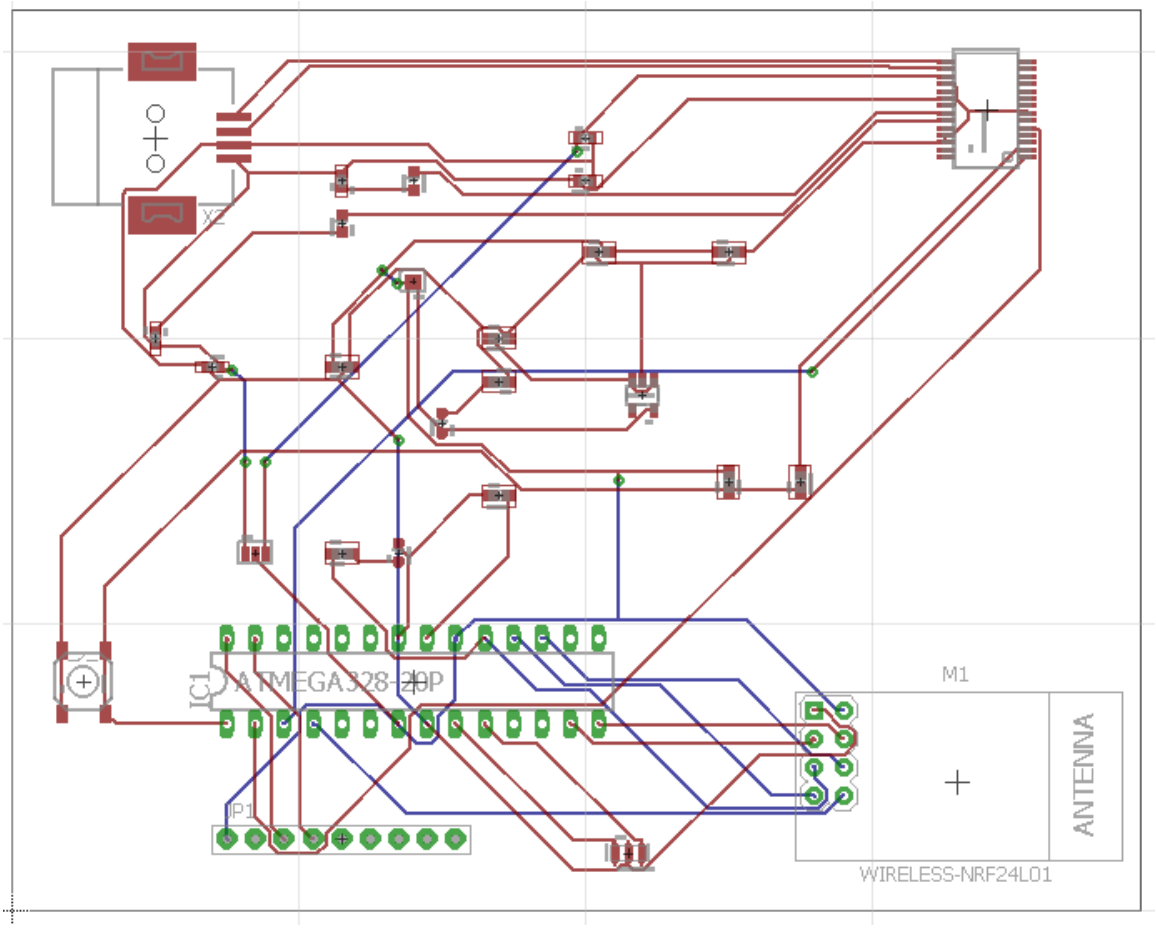


Figure 7.2F: Rough prototype of PCB using Eagle software.

Below in Figure 7.2F is an end result of the PCB that we will be using. The end result did not work for unknown reasons. The board needed to use an FTDI to communicate between the USB B and the ATMega328P. The FTDI used (FTDI FT232RL) was recognized and drivers downloaded by any computer connected. The ATMega328P, however, was unreachable and could not be communicated to by computer. Because the communication between the computer and the ATMega328P was integral in it working, the PCB was unusable.

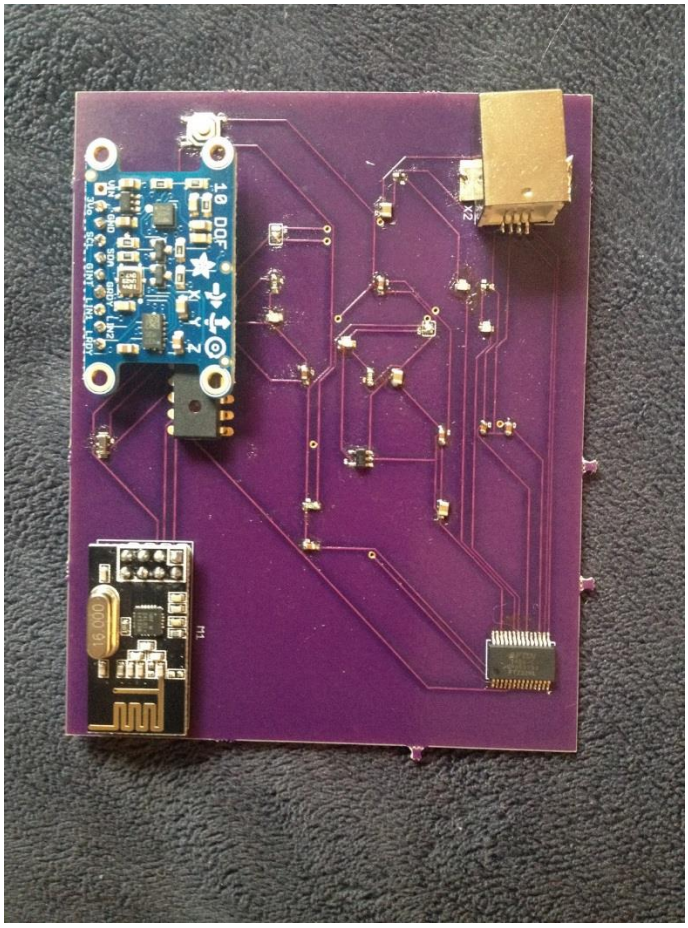


Figure 7.2G End resulting board created for the Glove

The general design of the PCB was to use basic commonalities within FTDI breakout boards in conjunction with basic commonalities within Arduino breakout boards which have the same pin layout and frequency rate as the ATMega328P microcontroller that was used on the project. This was then compared to the working Arduino Uno development environment to ensure that the result could work as a bare-bones version of a working Arduino Uno.

8. Host Machine Design

Below is the design aspects for the host machine. These include the Architecture and major components for the host machine. Currently the host machine is planned to be Matt's Personal computer as it has the needed power to run the controller and the Oculus Rift. The later plan though is to downsize to a Raspberry Pi Model B. Below will be listed the Raspberry Pi Model B planned upgrade. Further below in the specs section will be Matt's current personal computer as well as the Raspberry Pi's specs.

8.1 Host Machine Architecture

Our future host machine will be a Raspberry Pi Model B+ (RPi) this machine utilizes an ARM instruction. The Raspberry Pi has a couple of different alternative builds each with differences between them.

Feature	Model A	Model B	Model A+	Model B+	Compute Model
BRCM2835 SoC	Yes	Yes	Yes	Yes	Yes
STANDARD SoC Speed	700Mhz	700Mhz	700Mhz	700Mhz	700Mhz
RAM	256MB	512MB	256MB	512MB	512MB
Storage	Full SC	Full SD	Micro SD	MicroSD	4GB eMMC
Ethernet 10/100	No	Yes	No	Yes	No
HDMI output port	Yes	Yes	Yes	Yes	Yes
Composite video output	Yes	Yes	On 3.5mm jack	On 3.5mm jack	Yes
Number of USB 2.0 ports	1	2	1	4	1
Expansion header	26	26	40	40	N/A
Number of available GPIO	17	17	26	26	48
3.5mm audio jack	Yes	Yes	Audio/Video	Audio/Video	N/A
Number of camera interface ports (CSI-2)	1	1	1	1	2
Number of LCD display interface ports (DSI)	1	1	1	1	2
Power (bare, approximate, 5v)	300mA, 1.5W	700mA, 3.5W	N/A	650mA, 3W	N/A
Size	85x56x15mm	85x56x17mm	65x56x12mm	85x56x17mm	62x30x3mm

Table 8.1A: Comparison between different models of Raspberry Pi(s).

The following table will be used, as the description says, as a comparison between the different models of Raspberry Pis. The different models, of course, will come at different prices with different advantages. Thus, this table will provide us with different options should the occasion arise that we have a problem. These problems will hopefully be very apparent since we will be working with both a Raspberry Pi as the host machine and a Raspberry Pi as the on-Quadcopter-computer. The hope being that we will be able to tell whether the host machine Pi or the quadcopter Pi is the one that is malfunctioning or needs an upgrade.

Additionally, the information given by the table shows the limitations of the Raspberry Pi and all of its models. Should we come to a problem that cannot be fixed by buying a model with more storage or more GPIO, then we will have to come to the conclusion that a larger, more capable host machine will have to be acquired.

8.2 Host Machine Major Components

For testing purposes and to prevent bottlenecking, we will use Matt's personal computer as a host machine. We will do this while we try to downsize the host machine to the Raspberry Pi. Matt has confirmed that the Oculus Rift does work on his computer which will allow us to begin coding for the Oculus Rift and the glove controller. The specifications of Matt's personal computer are below.

8.2.1 Motherboard

Personal motherboard choices can vary by the users and no particular standard is set. So long is it can meet the minimum requirements for the project then the motherboard will be alright to work with the OR.

MSI Z87-G45 Gaming Republic

- Supports 4th Gen Intel® Core™ / Pentium® / Celeron® processors for LGA 1150 socket
- Supports DDR3-3000(OC) Memory
- USB 3.0 + SATA 6Gb/s
- Audio Boost: Reward Your Ears with True Quality
- Killer Ethernet: Kill Your Lag
- Military Class 4: Top Quality & Stability
- OC Genie 4: Overclock in 1 Second
- Click BIOS 4: Easily Fine-tune Your System
- PCI Express Gen 3: World's 1st PCI Express Gen 3 Motherboard Brand
- Multi-GPU: NVIDIA SLI & AMD CrossFire Support
- Sound Blaster Cinema: Realistic Surround Sound Experience
- Gaming Device Port: Optimized with Triple Gold-plating for High Polling Rate Gaming Devices
- Total Fan Control: Optimize All Fan Speed As You Wish
- Lucid Virtu MVP 2.0: Uncompromised Game Response Performance

- Fast Boot: Quickly Boot Up & Enter OS in A Few Seconds

8.2.2 Processor

The processor that is running is listed in the table and part list below. It should be anything above 3.0 GHz. Any of the new Intel i5 or i7 series will work fine for the MSI board.

The planned processor for the Raspberry Pi: 900MHz quad-core ARM Cortex-A7 CPU

Intel(R) Core(™) i7-4770k CPU

# of Cores	4
# of Threads	8
Processor Base Frequency	3.5 GHz
Max Turbo Frequency	3.9 GHz
TDP	84 W
Max Memory Size (dependent on memory type)	32 GB
Memory Types	DDR3 and DDR3L 1333/1600 at 1.5V
Max # of Memory Channels	2
Max Memory Bandwidth	25.6 GB/s
ECC Memory Supported	No

Figure 8.2A Intel i7-4770k CPU specs

8.2.3 Random Access Memory

Corsair Vengeance 8Gb (2x4 GB) 240-Pin DDR3 SDRAM
- DDR3 1600
- Timing 9-9-9-24
- Cas Latency 9
- Voltage 1.5V

Planned RAM for the Raspberry Pi1GB LPDDR2 SDRAM

8.2.4 Storage

Samsung SSD 840 PRO Series ATA - 128 GB solid state

Planned Storage for the Raspberry Pi MicroSD card socket with 16gb MicroSD card

8.2.5 Graphics Card

EVGA 04G-P4-2972-KR GeForce GTX 970

- 1664 CUDA Cores
- 1050MHz Base clock, and 1178MHz Boost Clock
- 4GB GDDR5
- 224.3GBps Memory Bandwidth
- 7010MHz Memory Clock
- 145 Watts Power Consumption under load
- DL-DVI-I, HDMI and 3 Display Ports
- Supports up to 4 monitors
- SLI ready (2 Way, 3 Way, 4 Way SLI)

8.2.6 Power Supply

Corsair 800Watt Power supply
+5V @ 2A via microUSB socket
8.2.7 Final Host Machine

Ultimately we decided we would sacrifice the Oculus' functionality in order to make a more portable system that we could expand upon later. Our final host machine was the Raspberry Pi 2 Model B utilizing the Oculus Rift as a head-mounted display.

9. System Integration and Testing

The following sections list the plans to integrate and test the components with their individual parts. Parts include video streaming to the Oculus Rift from the camera, the Remote control from the glove and the glove's functions, the controllers' impact on the Oculus Rift. Tests will be done frequently as parts are matched together and individual sections are completed. Main testing will be held by each group member on their particular group set but when integration comes together different portions will overlap and the group will reform to complete the portions of testing.

9.1 Video Streaming

The video will be streamed using as minimal third-party modules as possible in order to increase portability. The Raspivid Modules is built into the Raspbian OS, this will be used to receive camera data from the CSI connection on the board. This video will be sent over the network through UDP protocol, this is due to the Netcat module allow the Raspberry Pi to write to a network stream to a particular IP address. The Host machine Raspberry Pi will equally listen on port 2222 by default via Netcat for any network data. Once data is received it is sent to the Mplayer module which displays the picture to the display, in this case the Oculus rift.

Raspberry Pi	QuadCopter Camera	Host Machine
Ping Router / Other Pi	Yes/No	Yes/No
Raspivid Module	Yes/No	N/A
Netcat Module	Yes/No	Yes/No
Mplayer Module	N/A	Yes/No

Table 9.1A Component Test List

Testing the overall portions of the camera and its software configurations will be primary and after signals are received for the camera, integration with the Oculus will become the follow up priority

Our initial testing plan will likely consist of getting our networking to feed a single raspberry Pi + Raspberry Pi Camera module to our host machine. Once we can properly receive that data we can begin displaying it in our Oculus Rift. There are 3 possible setups for our streaming module with which we can proceed, in the order that we will likely be experimenting with. Their advantages and disadvantages will be detailed below.

Single Raspberry Pi + Wide Angle Lens Raspberry Pi Camera

Having a single Pi + Camera setup will give us the lightest weight as well as the smallest power cost when taking into account the Pi + Batteries involved with powering each component. This option is also the most affordable. There are some immediate concerns with this setup however, the Oculus Requires Stereoscopic video feed to deliver a full 3D view. With this setup we would only have a single reference point for our video feed and would have to then sample and cut apart our feed as necessary before we transmit and output to our Oculus.

This could cause some delay concerns and might not be entirely feasible depending on the processing power of the Raspberry Pi module we decide to use. The supported Camera module however comes with built in encoding so this concern may not prove to be as much of an issue as originally estimated. The camera module itself comes with command line instructions for easy testing.

The Command raspivid -o vid.h264 allows us to save a 5 second video file to our given path, in the natural encoding setting of the camera. Further arguments can be specified such as the resolution sizes, the time length the fps settings we desire allowing for a large amount of customization to fit our needs. Complete documentation of the commands and arguments are available to us on the developer's site for Raspberry Pi and its associated modules.

To solve the issue of Stereoscopic input needed we could split the feed and duplicate it on a software level, as well as introduce any filters to simulate the angle difference of a human eye. A diagram outlining an example of how this would be implemented is below

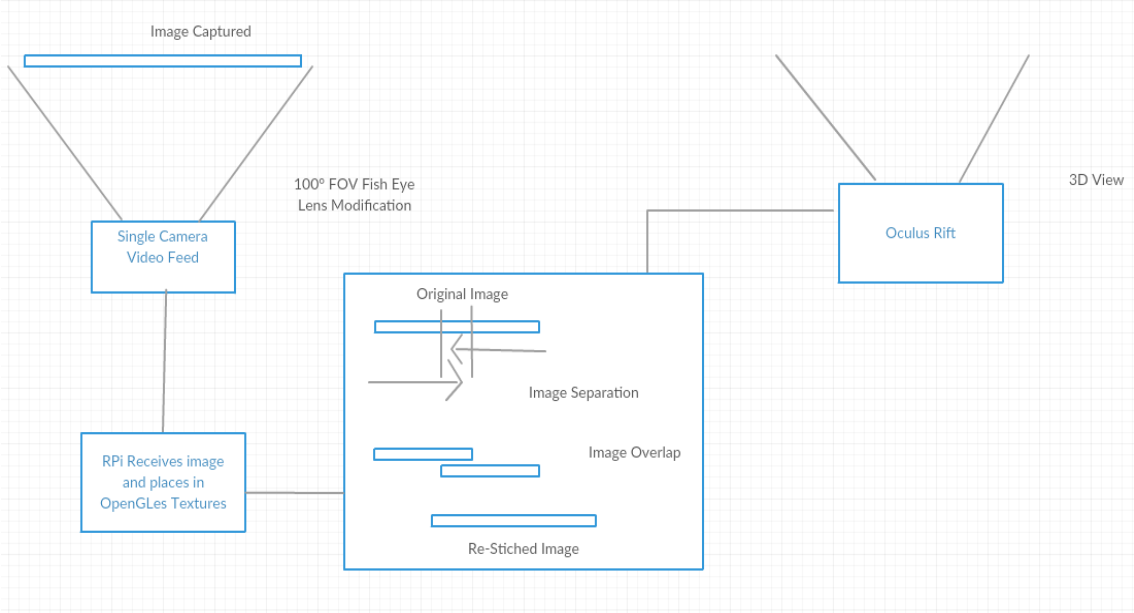


Figure 9.1B: Proposed Image Sticking Solution for single camera feed

We would have to test the delays of this implementation, as well as if the quality obtained is worth the delays of just simply feeding in a single wide angle lens view. The natural distortion of the lens might be enough to not require extra detailing to be done post process. This will be our first line of testing, as it will be required if we wish to continue onto our second alternative anyway.

Two Raspberry Pi + Two Raspberry Pi camera

If we decide that the quality of the video feed is insufficient with the single Camera setup, or that the distortion makes viewing uncomfortable. We could then consider lowering the individual resolution settings on our initial Raspberry Pi camera, (to help with data management) and incorporating a second Pi + Camera system. By doing this we can manually position the cameras to achieve the field of vision we desire. An immediate concern is that we would now require double the bandwidth we were already using for our single implementation and would need to now network the two Pi's together, through a client-server system, to make sure our video feed is synced together before we export it out through our RTP connection into our host machine.

There are worries about un-syncing when the wifi connection we utilize becomes unstable, we want to avoid the possibility of the feed splitting, if we wish to resolve this by instead letting our host machine handle the load we can implement a similar system by simply having each RPi function as its own server and stream their video feeds individually.

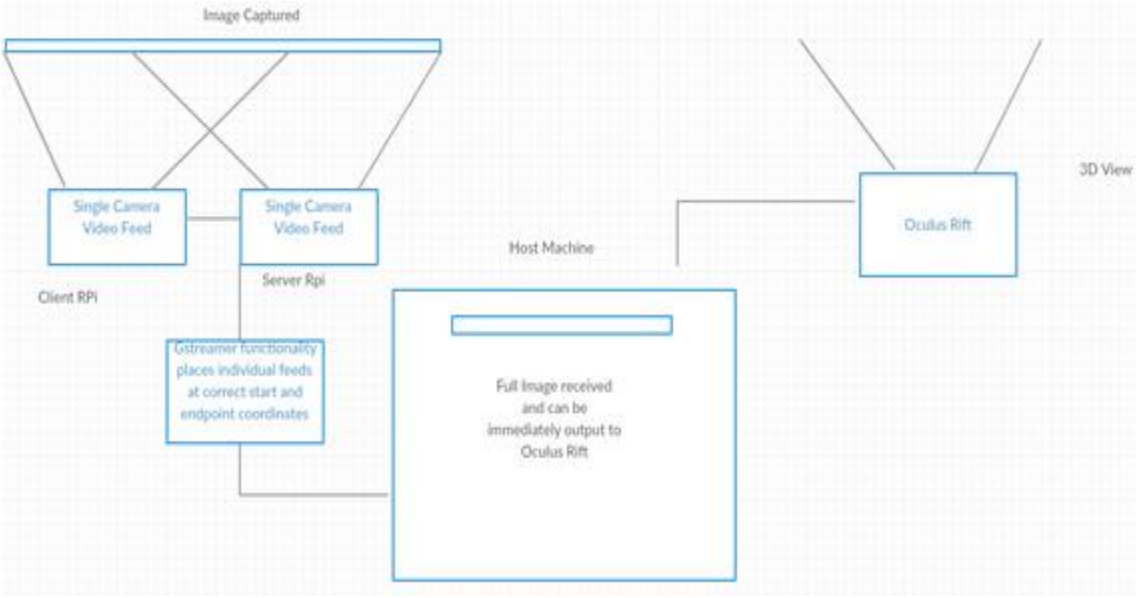


Table 9.2C : Proposed implementation of two Pi system

Because Raspivid does have built in functionality for interfacing two boards, which can be accomplished by setting our viewing port to the same ID on each board, we could possibly save ourselves the trouble of networking the two streams to each other, and simply sending this input to Gstreamer. By separating the ports on our host machine that are receiving each individual camera (eye), GStreamers videomixer functionality will allow us to stitch together our two individual video feeds into a single visible stream on our host machine.

Single Raspberry Pi : Compute Module + Two Raspberry Pi Camera

The final method is a sort of last resort. If we find that the networking delay between the two raspberry pi's are too large and cause notable unsyncs we have to pursue a stronger board, the Raspberry Pi:Compute. This module and its associated IO board have space to connect two Raspberry Pi Camera Modules. The cost of this kit on its own however totals $100+ and we would like to avoid having to resort to this option. By having onboard support for our two cameras and stronger processing power, we don't have to worry about any networking delays and can bundle our video feed before sending it to be more easily decoded on our host machine. Having to only power a single board will also alleviate some of our payload concerns for batteries. Once we have learned how to stream the double Pi+camera setup, modifying our code to implement the Compute model is trivial.

Further details as we begin testing each setup will be documented at a future draft, including vitals such as any delays, quality and frame rates achieved, battery life and overall personal comfort level of individual group members.

9.2 RC Control via Glove

There were a few design changes in the implementation of the Glove, as such we had to go through extensive testing to see if we could still manage to get a similar range of control as we had originally planned for. Because we had moved to a model involving a pair of

Arduino Uno boards the amount of control we had between the signals being sent out and how they are received was much higher than we originally anticipated.

The accelerometer connected to the transmitter Arduino Uno module is sending out packets to the receiver module located on the quadcopter with a set delay of 250ms between packets (This is an artificial delay we implemented ourselves to help with consistency of controls). The raw data of the accelerometer is scaled into a range of values between 0~250 with 125 being our neutral position, if comparing to the common control scheme of quadcopters using potentiometers on a large controller.

The receiver takes in these values and scales them back into a range of 1000ms~2000ms values (these signify the width of the pulse we desire to send to the motors), the flight controller helps us with the individual control of the motors by separating their functionality into 4 channels of movement : Throttle, Rudder, Aileron, and Elevator so we simply send these signals into our desired channel.

Because of complications in the sensitivity of the sensors themselves, and general desire to keep an intuitive control scheme, at the moment we are limited to 2 Axis's of movement for a single hand controller. At the moment the Elevator (forward/backward) and Rudder (turn left/right) are the only channels we control with the hand controller. The height of the copter is set manually through the serial monitor before beginning control with the glove. Future implementations will likely include a second glove to gain access to the remaining 2 channels and have a fully controllable quadcopter through our glove modules.

9.3 RC Control Monitoring via Oculus Rift

The Oculus Rift's functionality with the Glove controller will be designated to the Camera's functions including looking up, down, left, and right. when the Quadcopter is twisted left and right to spin horizontally in either direction the camera will remain stationary. There were a few design parameters around this, the most important being the complete separation of controller and Oculus.

Matt had huge worries concerning this as an avid gamer, when camera functions are bound together with other functions like movement it can hinder gameplay heavily and be extremely disorienting. Camera head orientation should act as a turret style mechanic with limited motion of the body and camera swivel limit. However this is not to say that it will be easy to get used to on the first try for new users. This will force the user to react more to the need of using the glove to turn to associate the difference between camera and orientation.

As was mentioned in an earlier section when we were discussing the features and game engines available to us our graphical overlays are primarily for the user's benefit. To help the user better control the device we planned to add in information tracking to be viewed on the Oculus Rift through the game engine. Little details such as the current orientation of the glove or the strength of the sensors current readings can be useful information to the user so they can better adjust how they move their hand. By having access to those readings

the user is then able to more accurately adapt to the controls and pilot our Copter more efficiently than if they were to dive in without any experience.

10. Project Management

This section will describe the full content and detail to our planning and testing that will take place from now through senior design II. Subsections include Documentation, Team Organization and Time planning, Schedule and Work Breakdown, Budget, and Method of Approach. This is essentially an initial outline of what will take place as soon as the semester begins and construction and design can commence.

10.1 Documentation and Organization

Our team followed a development model similar to that of the Scrum, agile software development life cycle. Each week during development we will have a meeting to status, discuss current tasks, and plan ahead. These meetings, are documented discussing the topics, concerns, and successes from the week prior.

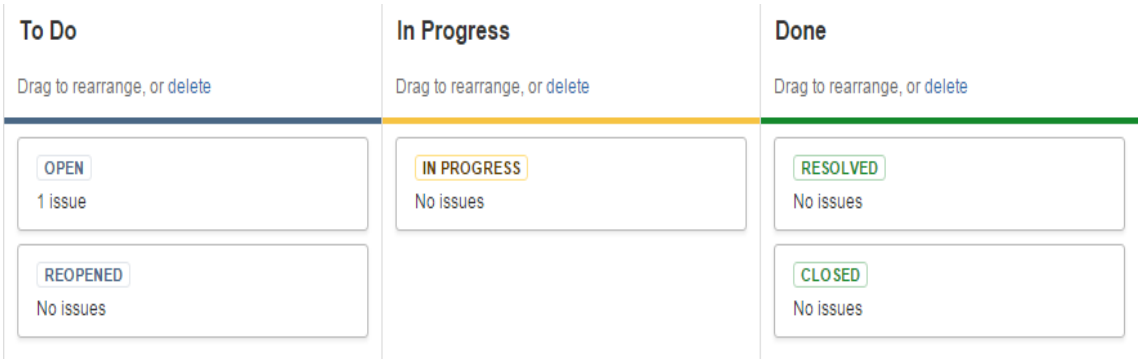


Figure 10.1A Scrum Agile Swimlanes

Furthermore a simplified workflow is in place as a procedural use method in order to keep ourselves on track as we progress. The stages are simplified so that as issues are discovered they can be picked up by a team member, be worked through, then resolved at completion. Usually this method is used with sprints to better sort through time constraints and plan faster. Currently, Atlassian has a planning tool for Scrum with Agile and Jira. Matt has in depth experience with its uses from his internship and will be implementing an offline version of the Agile process on a post-it board. This will help the team to track problems faster and prioritize work to be accomplished.

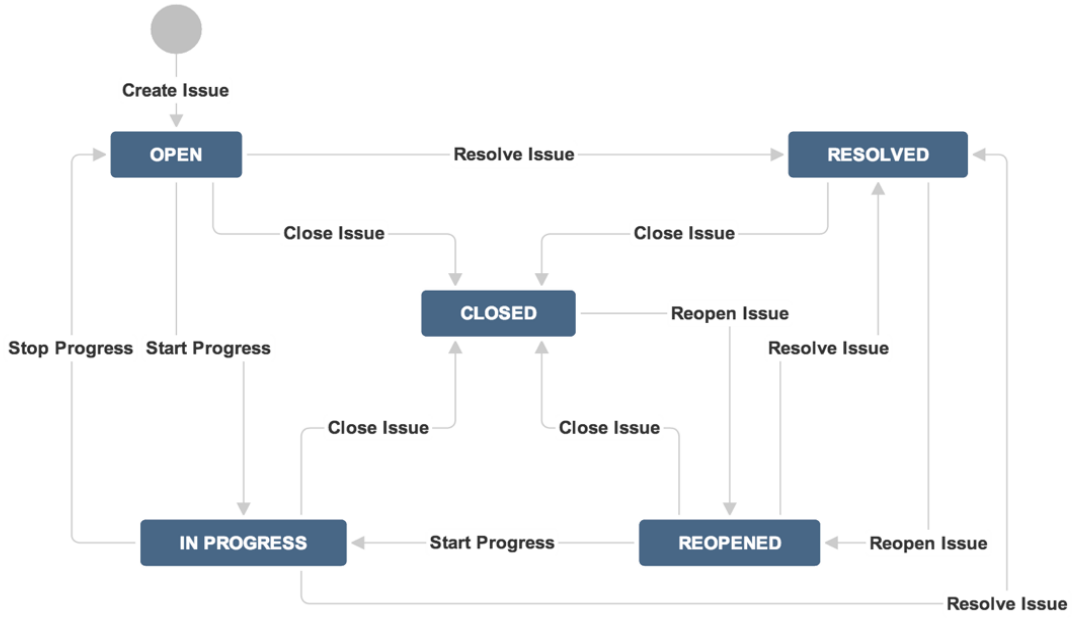


Figure 10.1B Workflow Diagram for Issues

From Figure 10.1B, have the most simplistic version that Atlassian provides for a project workflow. When an issue is created it is set into open. An issue has a few major pieces to it. Firstly, a title and summary of what the problem or task at hand is. Next a summary description of what is involved. JIRA allows for further customization with components such as version affected and expected completion dates. The parts we will focus on is date of issue/ticket creation, hours worked on the issue, expected hours to complete the issue, and who it will be assigned to to work on the task. With these descriptors we can progress issues along at a quick pace.

When a task is picked up by a group member they can move it to In Progress, Resolved, and Closed. If a discussion simply solves the problem or a mistake is discovered early on it can be resolved quickly and the issue can be closed. Closed issues can be reopened if a future problem is discovered as well. As an example if we clear up a section of code and the oculus works correctly we can close the issue. But if a function makes a call back to that section and it works incorrectly again we main need to reopen the earlier issue and look to see what other problems there could have been involved.

10.1.1 Code

The code base was kept and maintained using a Git repository for source control. We further maintained the code base by having weekly forks clearly marked e.g REPO_July_13. We felt that a weekly cycle would be a good strategy in the event of any malfunctions, giving us the ability to restore an earlier working place.Each member of the team contributing to the codebase creates their own working repository and would commit when they felt a contribution has been made to their project or current task.

Formal Guidelines:

- Use 1-2 lines of whitespace in between segments of code
- Each function/method definition will have a comment explaining the purpose and the return value
- Variable names will be clear to the scope of the software layer
- Proper naming capitalization is Camel-Casing as such “exampleCase”
    - Underscores may be utilized when
        - Using an API whose naming conventions use such
        - Clarity of the variable, rare occasion
- Strong attempts to make the code as modular as possible must be made.
- Any debug messages not in a developer’s personal branch must refer to the function/class/application that it is derived from, as such “Debug: Example Message - sendPacket : packet.c”
- Git commits will contain the following information:
    - General Description of the changes
    - Issue Number (if applicable)
    - Reason for Change
    - Author’s Name
- Each source file will contain a change log that notes the author, date, and short description of the changes or introductions
- Any function calls to an open source library or API should include a comment to give a high level idea, this is to increase readability
- Prior to git commits the author should pull a clean trunk branch, and do a diff and merge where applicable prior to the commit
- The repository will contain a weekly branch marked as such, and developers should strive to be on the most current branch as frequently as possible
- These guidelines apply only to the trunk build, each developer can code as they see fit in their branches so long as the code has been formatted prior to submission

10.1.2 Design Journals

Design journals should contain as much information as possible with regards to design, including research or reference to, previous versions or plans regardless of functionality, and benchmarks/tradeoffs. Primarily this includes keeping old information that may have changed, this should be documented with a change log attached at the end of each file containing a short description of the change.

If sections have completely been removed or changed, the author making the changes should effectively duplicate and rename that design document with the date and then can edit the original working copy. This allows us to never have to try to redesign a previous idea anything if one design does not work as we like.

10.1.3 Manuals

For each main subsystem we will be creating a manual explaining the basic scope and operation of that subsystem. There will also be a manual for the total product, this will contain more in-depth information about the usage of the product. The general layout for the manual will be as such:

- Introduction
- Acronym and symbology listings
- Basic Use
- Connectivity in the System
- System Limitations
- Advanced Use (Where applicable)
- Technical Information
- FAQ's
- Legal Limitations and Regulations - Main Manual
- Parts List for Replacements - Main Manual

10.1.4 Media

Our media is stored in three primary locations

- Code : Git Repository
- Documentation : Google Drive
- Master Copy : External Portable Hard Drive

All media is subject to version management, and should not contain any discrepancies with the data covered. Meaning that one media platform should not host any information that is crucial without backing up the data to the master drive. All developers are required to keep a personal copy of any of their work on a storage method of their choice. In the event all three of our primary methods fail, each member will have a backup of all their information, so we can rebuild the project.

When it came to choosing our media storage method all of the group members had different approaches to how the media should be handled. We initially thought that we would all use Google Drive to keep our media since it has the integration with all of Google's other office tools. However we quickly realized that would be a poor choice for the code base, and something like SVN or Git would be better suited towards that use case. Our decision to go with Git over SVN was primarily because Git is a trending technology right now and we felt it would be good experience to gain.

Additionally since some of group members live further away than an easy drive, we believed that a decentralized method of version control would work better than SVN's set up. Our group has had experience with SVN in the past, and it offers a nice function set while being easy to learn. While security is not a concern within this project we felt that having a secure copy of data would be a good practice to implement.

Storage Method Analysis		
Method	Pros and Cons	Use Case Bold means used
Git Repository	Pro: - In-demand knowledge - Access via Github - Branching allows for fast prototyping - Smaller repositories than SVN - Distributed Source Control - Faster than SVN Cons: - Learning curve - Clone entire repository	- **Source Code**
Subversion (SVN)	Pros: - Better UI than git - Marks versions better - Check out subtrees - Easier to use than git Cons: - Most of Git's Pros	- Documentation - Source Code
Shared Drive - Google Drive	Pros: - Everyone has access - Collaborative tools via Google's suite - Mobile Access - Familiarity Cons: - Online - No automatic version marking	- **Documentation** - Source Code
Physical Media	Pros: - Offline Access - Security - Independent of companies' uptime Cons: - Physical - Expensive depending format	- **Documentation** - **Source Code**

Figure 10.1C Media Software Pros and Cons

Since data is hosted on our Google accounts which historically can contain valuable personal data we would like to have a backup in the event that a malicious user gains access to our account, with malicious intent or by a group member leaving their google account signed in on a public machine. While we felt this scenario is extremely unlikely we would also rather be prepared for such a catastrophic event. Reassuringly Google Drive does have methods in place to allow for file recovery, but we do not have the same automatic features within Git.

10.1.5 Miscellaneous Notes

Group members are expected to upload any notes to the Drive under their own personal file. These notes should be sorted on at least two main levels, the author followed by either the subsystem or the task the notes align with. There is no formal guidelines for the format of the notes within due to allowing members to take their notes in a style that is best suited to their work style.

As the project progresses, this style may be changed to better alleviate space and placement for files. Files may be arranged as things pertain to Documentation, Programming, Circuit Maintenance and Construction, etc. Currently a high level is in place for initial design documents.

10.1.6 Planning Documents

Any planning document should be clear and concise for the component they are planning. These documents will be uploaded and kept within the drive as well. The planning documents will benefit from being modular, as it will allow us to focus on components at a smaller scale while working our way "up" to the higher level component pieces. Ex. planning the camera-raspberry pi connection will come prior to planning game engine overlay to the video feed from said camera. Additionally the planning documents should contain references to where the original data and/or resources exists, so that other members can quickly gain similar knowledge. The goal is to be able to have a member from the opposite sub-team be able to understand components outside of their scope, should it be for additional help or understanding the system better.

10.1.7 Status Reports

Within our weekly meetings we document our progress through a few criteria.

- Functional Progress - Progress of any subsystem having functionality
- Developmental Progress - Status regarding a development task
- Next Week's Target - A decision of where we want to be next meeting
- Blocking Factors - List any developmental pieces that need to be completed in order to progress
- Purchase Needs
- General 4 Week Schedule
- Concerns

These status reports are then saved into a folder specifically for them within our shared Google Drive. This allows anyone to go back to trace changes and tasks decided for each week. The focus is to be able to rollback any changes easily after a failure.

10.1.8 Tests

Once we have the quadcopter, it will be tested as it has come pre-packaged. Each of us will test it, which will give four test results that should coincide with the quadcopter's ultimate capabilities. We will use the same table for the given RC controller as for the custom made Glove. This will be a more general testing for the glove to see how well it performs in a leisurely environment. Before and after data will help with adjusting the glove and knowing limitations with the controller and quadcopter. The table following this next explanation paragraph will be filled out during testing.

We will rate how well the quadcopter and RC controller respond to each other in "Responsiveness". This will help us determine if we can continue with the quadcopter's built-in RC receiver or if we have to remove it and either install another WiFi receiver to read the glove's controls or use the WiFi receiver that is sending and receiving video data. A rating of 1 will be that the quadcopter does not respond well at all and a rating of 5 will be that the quadcopter responds perfectly.

We will also rate how well the quadcopter, itself, will be able to stay still given the correct conditions in "Stability". The "Stability" section is important since the Oculus Rift is prone to cause nausea in those not used to it, so if the quadcopter is not steady then the user will possibly be even more prone to nausea. A rating of 1 will be that the quadcopter continues to list even after on-board settings are adjusted and a rating of 5 will be that the quadcopter was able to stay perfectly still. The motor and power distribution will be edited if need be. The "Battery time" section is not something that we can rate, but it is data that needs to be recorded and this table is a fine chance to record it.

Rating 1-5 (if applicable)	Gus	Matthew	Gunnar	Craig
Responsiveness	4	5	4	4
Stability	1	1	2	1
Battery time	4	5	4	5

Table 10.1D Responsiveness check table

The stability is a HUGE factor to flying with the quadcopter. An excessive amount of balance needs to be applied when adding the components as the slightest weight increased cause the copter to drift in a particular direction.

The Oculus Rift will have to be tested on its initial PC platform before it is tested on the Raspberry Pi. Once again, each of us group members will test the Oculus on its initial

platform to make sure its base settings and parameters are performing correctly. We will use the same table for the PC platform as for the Raspberry Pi platform. This will give us data for comparison in deciding whether to move forward with the Raspberry Pi option. The table following this next explanation paragraph will be filled out during testing.

We will rate the video quality of the Oculus rift in the "Video quality" section of the table. This will be more for comparing between the Oculus Rift being run on a PC and it being run on the Raspberry Pi. A rating of 1 will be extremely poor video quality and a rating of 5 will be extremely good video quality. The "Brightness" category will, of course, be for us to rate the brightness of the Oculus Rift's screen. This will be on a scale of 1 for too dim and 5 for too bright. This data will be recorded in the case that nausea is a result of the screen being too bright or too dim. The "Brightness" category will also give an indicator as to the average customer and how we can contour the product to fit every need.

The "Motion controls" category will measure how responsive the motion controls of the Oculus Rift are. The motion controls in question being the property of the Oculus Rift to turn the "camera" of the virtual reality world in response to a person turning their head. Between four people, it should give a dependable indicator as to whether the individual feels that the camera moves out of sync with the individual's head motion.

A rating of 1 will indicate that the motion controls are too slow and a rating of 5 will indicate that the controls are too fast. Once again, we have a category that cannot be rated, but it is data that will be useful and needed. This will give a better idea to how long the customer should be exposed to the Oculus Rift. Given our possible goal of quadcopter laser tag in an arcade-like environment, we can have the games regulated to last for a certain time.

Rating 1-5 (if applicable)	Gus	Matthew	Gunnar	Craig
Video quality	3	3	4	3
Brightness	3	3	3	3
Motion controls	2	1	1	2
Time to nausea	3	3	3	3

Table 10.1E Quality Check table

The unfortunate result of this test was that the quality of the pi's stream to each other to the Oculus suffers a massive delay. This is mostly due to the write time on the Pi's. A more suitable system may have been to switch to the Beaglebone. But that requires time and money. Both of which we were greatly limited on. In future trials we may push for better parts to see if that balances out the technicalities that were suffered during the test. The long of the short is you won't fly it with the Oculus at its current frame rate and transfer rate.

We now have to test how well the Oculus Rift communicates with the Raspberry Pi's camera attachment via WiFi. Once again, each of us group members will test the Oculus communicating with the camera to make sure that it is performing correctly. We will use the same table for the PC platform as for the Raspberry Pi platform. This will give us data for comparison in deciding whether to move forward with the Raspberry Pi option. The table following this next explanation paragraph will be filled out during testing.

We will rate the video quality of the image coming in from the camera in the "Video quality" category. This will give us insight as to how well the host machine and camera are communicating. Given the data previously recorded of how well the video quality is on the platform it was made for, we can compare whether the camera over WiFi is working properly. A rating of 1 will indicate that the video quality is poor and a rating of 5 will indicate that the video quality is good. Another category that we will be rating is the "Video speed" of the camera over WiFi. This meaning that we will be watching for any lag that happens to the video while streaming.

This will be another indicator as to whether or not the camera is communicating properly with the host machine over WiFi. A rating of 1 will mean that there is a lot of lag and thus communication is poor and a rating of 5 will mean that there is no lag at all and communication is good. We will also rate the responsiveness of head movement in the "Responsiveness" category. Since we will be controlling the area of sight for the Oculus Rift, will will also have control over the responsiveness of head movement and thus virtual reality camera speed.

This can be a large contributing factor to nausea and thus must be considered carefully. A rating of 1 will mean that the responsiveness is extremely poor and a rating of 5 will mean that the responsiveness is extremely good. Again, we have a category that cannot be rated, but it is data that will be useful and needed. This will give a better idea to how long the customer should be exposed to the Oculus Rift. Since this will be a more accurate representation of what the customer will be viewing, our possible goal of quadcopter laser tag in an arcade-like environment will be better simulated and a time limit will be better represented.

Rating 1-5 (if applicable)	Gus	Matthew	Gunnar	Craig
Video quality	3	4	3	4
Video speed	3	3	4	3
Responsiveness	1	1	1	1
Time to nausea	1	2	1	2

Table 10.1F Video Quality Check

Again the quality of the video meets what we require however the latency of it is just too much for it to be viable.

Glove Control implementation required a handful of testing procedures to make sure things were working on their own. Initial planning stages involved connecting the Receiver/Transceiver pair and sending simple serial messages between them (that were hardcoded) without any artificial delays, to keep track of how strong the link is. We began moving them further away from each other to determine the rate of loss. We counted every 25 sets of packets and determined the rate loss by taking the ratio of packets that failed to arrive.

Distance	Rate of Loss
10 Meters	0%
30 Meters	1%
50 Meters	25%
100 Meters	40%

Table 10.1G Rate of Packet Loss

From the above chart it is visible that at around 100 Meters the chances of a packet being corrupt or failing to send are almost 50%, as such this is the limit of range we should ideally be able to fly. Because the # of packets in a second are in such a high quantity, even at 40% rate loss, there are few issues because the sheer amount of duplicates will likely replace the missing packet. The only time this becomes an issue is when in the manual control mode from the serial monitor rather than the glove controller itself, as there are notable delays between packets sent because they are limited to the speed of the user's typing capabilities themselves.

Once the connection between the two Arduinos were established there was little need to test anything else between them, as the amount of data and type we were planning to send in the first place was simulated in this first test. Once the code was all implemented and instead of dummy variables we began sending the actual output from the sensors we were able to move onto the final point of our testing. The Quadcopters responsiveness to our controls itself.

Located on the Quadcopter Flight Controller (The KK2.15) is an LCD screen with a menu and interface that makes debugging very easy. Through utilizing the Receiver Test functionality, which allows us to visually confirm the values of the signals being received by the flight controller and their respective commands they would send (Right/Left, Forward/Back for example). It was easy to turn on our Control system and visually confirm that our hand movements corresponded to the respective signals we desired.

10.2 Team Organization

As was mentioned in earlier sections the team is organized into two groups: hardware and software. Each group will remain separate on their portions of the project till testing and integration is required to bring parts together. This section entails the teams work on technical assignments, management assignments, working guidelines, safety guidelines and communication and accountability.

10.2.1 Technical Assignment Design Areas

The group was broken into two main groups, hardware and software. This will reflect the two separate main objectives that will come together in the end. The first objective being to make a glove that reads the inputs from the gyroscope and accelerometer combination and gives this data to a host machine that can then translate with quadcopter and all of its attachments. The second objective being to create a workable virtual reality environment seen through the Oculus Rift that picks up data from the Raspberry Pi's camera attached to the quadcopter and successfully updates that data in real time.

Hardware:

- Craig - Circuit Design and power systems
    - Safety Captain
- Matt - Circuit Design and integration
    - Documentation Lead

Software:

- Gus - Flight Controller and Raspberry Pi applications
    - Configuration Manager
- Gunnar - Raspberry Pi and Networking applications and Oculus Integration
    - System Administrator

10.2.2 Management Assignments

Management assignments will be more focused on keeping the teams up to date on each other's portions either day to day or week to week as things move along. Matt has taken to this role to keep things in check as the project moves forward and see that timing is running smoothly for each portion. Planning and focus for the designs is imperative to keeping the flow of the project moving and even as far as completing it earlier if possible. At this stage, most of the time limits planned are hypothetical in nature and due to change as challenges are faced or tasks are completed.

10.2.3 Working Guidelines

When working with any of the development tools and pieces they should be handled with care as to make sure no damage comes to the piece. Members are also expected to communicate to make sure no one is using any particular piece without the others knowing

it is use. This is to ensure there aren't any experimental components that could be problematic and damage any piece or person.

As more information on the project is created and updated it is important to keep a set of guidelines for keeping data manageable and secure. below is a list of software security guidelines

1. Do not share login information with anyone outside of the group.
2. Actively maintain a password of at least 12 characters in length.
    a. Password length is the quickest way to secure a password
    b. If possible make logging in a remote process with an encrypted key
3. Keep a local copy of the files being worked
4. Ensure logging out of project accounts when leaving a public computer.
5. Do not divulge technical inner workings to non-group members when avoidable. Specifically:
    a. Networking information
    b. Input (wary of buffer overflows)
6. When using email do not open attachments you do not expect.
    a. Never email files to a user, always upload through our media choice
        i. A malicious user could pretend to be a group member sending source code to another.
7. Drone's wifi network should be hidden as to minimize interest in the network.
    a. The network will be whitelisted so as to further deter malicious users
    b. Ensure network is WPA2
8. Remove any default password, again abiding by Rule # 2.
9. The root accounts must be disabled upon SSH.
10. Only the software team will have the root password for Linux machines.
11. Critical pieces of software should be backed up on physical media often.
12. Each system will generate logs of users, access times, and commands ran
13. Keep software as up to date as possible, specifically the operating system(s).
14. Keep software on the machine to the bare minimum, so as to not introduce vulnerabilities.
15. If a software package is being introduced to the system, run a virus scan on the package.
16. All of these guidelines apply to any computer the group members use, including smartphones and tablets.

10.2.4 Safety Guidelines

The safety guidelines for the group will be dependent on the portions of work. These rules and guidelines pertain more to the hardware team than the software but none the less should be followed should either team cross them.

To start, handling the quadcopter will be the largest factor of safety concern.

1. Always carry the quadcopter from the base. Carrying from the arms may cause damage to the arms and the propellers may go off accidentally.
2. Make sure to have the RC unit turned off before turning on the quadcopter. This will avoid the possibility of propellers hitting the hands trying to set up the quadcopter.
3. Conversely, make sure the RC unit is turned off before attempting to retrieve and turn off the quadcopter. The same reasoning applies here.
4. When testing in indoor environments, make sure to keep aware of ceiling height, people around, and objects in the space.
5. When testing in outdoor environments, make sure to keep aware of plants, wild animals, pedestrians, and weather conditions.
6. Do not fly in strong winds or rainy conditions. Strong winds can carry the quadcopter more than is within permittable bounds and water can damage important components.

We will also have to establish rules for handling electronics. This may be more for the safety of the electronics than for the safety of the individual but the rules are essential to both.

1. Make sure to ground yourself by touching something metal that is attached to the ground. This will release excess charges.
2. Make sure disconnect components from all power sources before working on them. Shock to the device or shock to yourself could ensue.
3. Keep your wires clearly color coded. This will help in not connecting something to a power source that shouldn't be.
4. Make sure to use wires with supple coating. Crossed wires can cause shortages that can damage equipment.
5. If you are unsure on where to place a wire, do not guess. Get a second opinion.

10.2.5 Team Communication and Accountability

Our group uses various forms of communication depending on the topic and severity of the topic. Given that a topic needs to be discussed immediately, a mass text is sent to each group member and a time is made where each can get to a computer with Skype accessibility. If the matter is not as urgent, then email is sent instead of text.

<u>Meetings:</u>
- Face to Face
- Skype

<u>General Group Questions:</u>
- Group SMS Message Chat
- Email Group

<u>Software Questions/Bug Reporting:</u>
- Email and Google Spreadsheet for Bug Tracking

- Detailed Git Commit Messages (for software status updates)

Hardware Questions/Problem Reporting:

- Email and Google Spreadsheet for Issue Tracking
- Google Spreadsheet for Development tracking (mirroring git commits)

Each member also holds accountability over particular design components. These design components will become more apparent aslog sheets are filled out and members who spent the most time on projects will most probably be the ones taking responsibility for the component's failure or success.

10.3 Schedule and Work Breakdown Schedule

First figuring out what problems need to be prioritized then determine a detailed workline and expectation timeline.

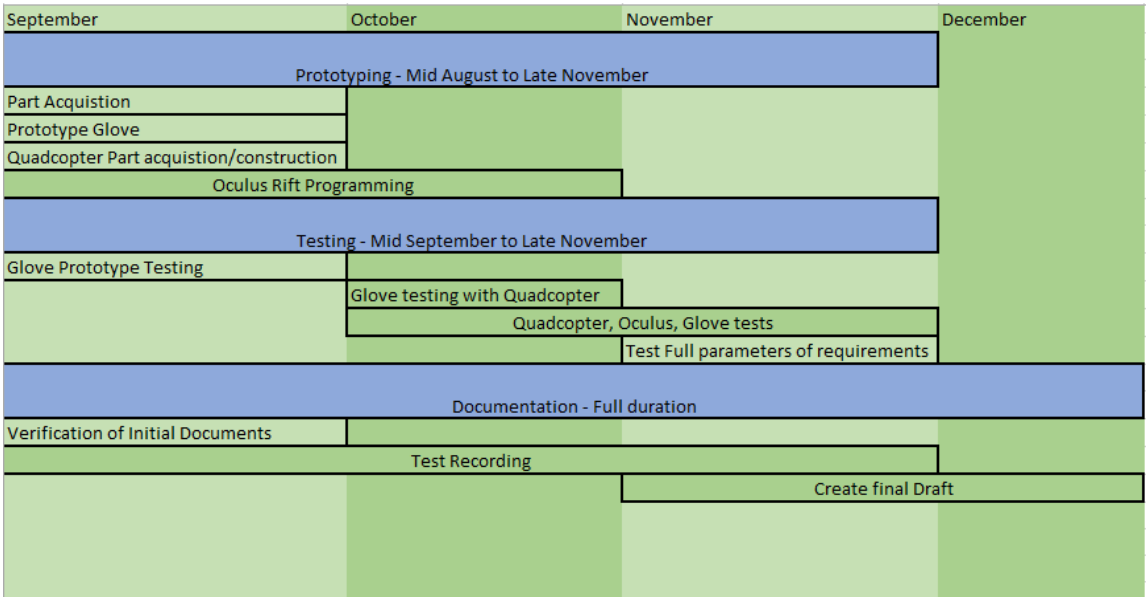


Figure 10.3A: Schedule Plan

10.3.1 Hours Summary

Total hours expected to accomplish everything. Senior Design 2 begins August 24th and ends December 7th. Due dates for specifics will be maintained at a later time. If 15 hours are dedicated per week for 15 weeks at a minimum then 225 hours will be available to complete the tasks.

Task	Total time Expected(hours)
Initial Programming For Oculus Rift	25
Initial Programming For Flight Controller	30
Glove Construction	15
Glove Testing	25
Quadcopter and Glove Testing	20
Oculus with Quadcopter Camera Testing	20
Oculus, Quadcopter, Glove Testing	30
Documentation	15
Total Hours	180

Table 10.3B: Hours time summary

With the listed hours we will have around 45 extra hours of which we will be able to dedicate to potential optional additions should the key requirements be met. As we will be running an Agile style build process, we will be able to confirm testing timelines and hours spent throughout the build process so that this value can be adjusted.

10.3.2 Milestones

Milestones for this project will be key points on completion readiness. Having an adequate amount of milestones will allow the team to have a better idea of what is to be expected as deadlines come closer. Our milestones are not in any particular order, however we do have milestones that aid in the progress of larger ones. Primarily we break the tasks down into small steps that are easy to implement, but add value to the main milestone.

General Milestone Outline:

- Order the parts
- Set up our main work lab space
- Configure git
- Configure defect tracking systems on Google Drive
- Set up our Jira environment
- Raspberry Pi
    - Set up operating system
    - Apply security fixes
    - Create user accounts for each group member
    - Create a logging system
- Create a cloned image of the raspberry pi with our final setup
    - Allows for easy restoration

- Prototype a basic networking test tool
- Receive camera data on RPi alone
- Send data between arduino and host machine
    - Should be data similar to the controller (analog/digital)
- Send camera data through RTSP
    - Oculus Rift
    - Overlay created
    - Camera Visuals updated to the Oculus
    - Feed
- Glove Controller
    - Construction Completion
    - Signal received by host machine
        - Can be tested with network test tool
    - Signal carried to Quadcopter
        - Can be tested with network test tool

Optional Milestone Outline

- Laser Integration with Quadcopter
    - Game controller software
        - Create basic point tabulation
        - Create a basic game lobby system
        - Create distributed networking broadcasting of score
        - Create a postgame sync
- Downsize Oculus Host Machine To Raspberry Pi
    - Includes configuration of the Oculus
    - Make this run on a battery matching the quadcopter's life

10.4 Operational Planned Budget

Currently we are projecting our budget based upon receiving no additional funding. Given the opportunity that we receive funding from corporate sponsorship, we may look into expanding our budget for either stronger parts or more functionality. The hope would be that the sponsor donates a set amount of money so that we can divide the money accordingly and possibly find a better deal for parts than would be available at the time of donation.

Additionally, if we receive just money, then we can enhance what we currently have, which may be more efficient than starting from scratch. Monetary donations also decrease risk of having to give the donated items back in their original conditions. The possibility of having to power the Raspberry Pi on the quadcopter by the quadcopter's battery is an example of altering a quadcopter past its original condition.

10.4.1 Project Cost

This final pricing table will be filled in as the parts are purchases. If a sponsor donates the parts, not currency, we have indicated as such. This is to give a clear representation of what

was purchased from our own funding, as to give the sponsors their credit. Monetary donors will be listed following the chart, mentioning their donation amount unless donor notes elsewise.

Funding Breakdown		
	Funding Amount	Expected Amount
Group Members	1000	1000
Part Donations (estimate)	200	250
Corporate Sponsors	0	0

Table 10.4A Funding Estimations

Funds were in acceptable bounds for the project.

Parts	**Estimated Prices**
Copter Components (Propellers, Hull, Motors)	Donated, Possible ~$100 upgrade
Flight Controller	$60
Gyroscope, Accelerometer, Sensors	$100
Oculus Rift	$350
Quadcopter Battery	$50
Raspberry Pi Battery	$20
RC transmitter/receiver	$20
MicroControllers/Dev Board	$100
Camera	$25
Wifi Transmitter/Receiver	$50
Wifi Range Extender	$25
PCB	$50
Total	**~$850**

Table 10.4B Part Cost list(to be updated as parts arrive)

10.4.2 QuadCopter components

In the case that the HPQ1 is sufficient for our needs, the Raspberry Pi, camera, and WiFi transmitter together will be about $90. This being $40, $30, and $20, respectively. Replacement propellers will cost all of $5 for a cheap set of spares while a new RC controller can cost as much as $70 depending on our eventual needs as they arise.

If it comes to the point that we need to buy a new quadcopter, then a new Traxxas 6608 has recently had a price cut making it a mere $100. This would lead to a few new expenses of solder, wire, and electrical components to re-circuit the power distribution to include the Raspberry Pi and its attachments.

Quadcopter Component list:

- HPQ1 Quadcopter
- 4 Rotor Blades
- 1 Rotor Concept TX7 RC Controller
- 1 Raspberry Pi
- 1 Camera
- 1 WiFi Adapter
- 1 Camera Mount

10.4.3 Oculus Rift DevKit 2

All parts and components of the Oculus Rift DevKit 2 came as a bundled package. Replacement parts and components (should they be required) will be updated on the funds tables as they are acquired. Below is the current list of components that come with the Oculus Rift.

Component list:

- Oculus Rift DevKit 2 goggles
- Oculus Rift Camera/Mount
    - USB and Mic Connector
- Power cord
- Spare Monocul glasses
- Carrying Case

10.4.4 Controller Components:

The end product will be the cheapest to produce. The glove, itself, will cost at most $10 while the Arduino Uno will cost about $25 and the accelerometer and gyroscope break ot board will cost about $20. So the prototype will cost about $60 to produce including a breadboard and wires. However, the accelerometer and gyroscope are about $4 each while the ATMega328 microcontroller as about $2. So the final product will be approximately $30 including PCB board printing and other possible components.

Component list:

- Glove(left and right)
- Glove mount backing
- USB Connector
- PCB with mounting
- Accelerometer
- Gyroscope

10.4.5 Project Cost Summary

So far as it stands, the following graph is a representation of how we have spent our money on this project. Because the HPQ1 was donated to us, it is the largest monetarily translated donation we have at a retail price of $900. The Oculus Rift was previously owned and sold at a reduced price. Thus, there was about $50 donated to our project and around $300 spent. Therefore, including donations, about $1250 have been spent on the project so far and we have about $900 left in our personal budget to spend.

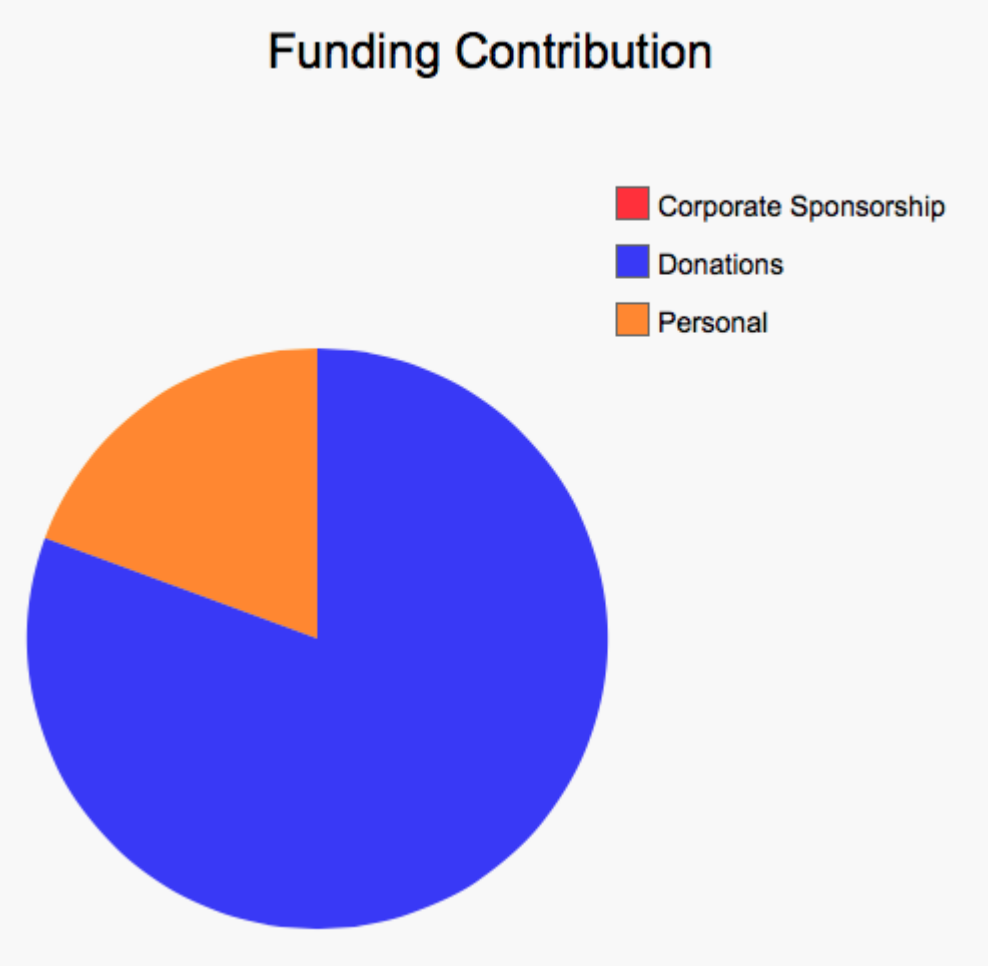


Figure 10.4C: Current budget pie graph(to be updated with more in the future)

10.4.6 Sources of Funding

The primary source of funding is being contributed by the group members. The group has received some generous donations towards the Oculus Rift and the quadcopter thus far. We as a group feel that our project has a strong appeal towards many of the companies surrounding UCF and we are actively pursuing corporate sponsorship. Specifically we are talking with companies such as AVT Simulation, Bohemia Interactive, Hard Knocks, and Inter-coastal Electronics Inc. We feel that our project directly relates to those in the simulation and gaming worlds and will actively seek more partners until we have sufficient funding.

10.5 Method of Approach

We plan to use the Agile process for problem and issue solving as mentioned in early sections. Using this process we can layout the design and assign tasks as needed based on levels of importance. This approach is easier to document as well as each change is monitored by the group as a whole. Testing, listing bugs, fixing said bugs, and retesting is the simplest process for the Agile Scrum cycle. Furthermore, in the sections to follow will be Design Methodology and research techniques used so far

10.5.1 Design Methodology

The main design approach followed an agile style modeled closest to the Scrum cycle. This consisted of our team designating tasks and try to complete that task by the next Scrum Sprint. We felt that this would be the best approach with a system that has a number of subcomponents.

That we would be able to effectively target our needs and be able to build a working product from the ground up focusing on the small pieces as we built the big picture. Often times this resulting in getting a small baseline version completed within a week. The following week would result in elaboration on the version to make a robust and feature/piece of the subsystem.

10.5.2 Research Techniques

Software Team:

For the software team a lot of the research was time spent reading forums for the particular technology, seeing if anyone had any experience doing such tasks, specifically looking for any errors or obstacles that they faced. As our project contains elements that are widely researched for use in a variety of hobbies and crafts we were able to find multiple similar projects with either wildly different or similar implementations. This gave the team a strong sense of the limitations of the technologies, while gaining insight. However the primary method of research was reading a lot of the technologies documentation.

Grasping a basic understanding of the API's used at a high level. Most of our research was conducted using online resources. Because a majority of software and information regarding it is Open Source the software team had nearly full access to the inner workings of any software they decided was worth looking into. Any forums or development pages, as well as documentation for possible libraries and functions are being kept separately on our shared Google Drive's folder for easy access should we find the need to go back and look through these sources.

Hardware team:

The main things the hardware team has to focus on are the Glove, quadcopter components and host machine, though the portion of the host machine will be mainly handled by the software team once it's components are gathered. The main research technique for finding the glove setup was first seeing what other remote controllers used for signal processing and data transmission. Many projects and current devices use microcontrollers so the next step was determining what controllers were of a reduced size so that they could fit on the back of a user's hand and remain a close distance to the host machine.

The arduino uno fit this description perfectly. Light and simple to program, we can use the main power of the arduino model to establish as stepping stone for the full components of the glove. An accelerometer and gyroscope were needed to calibrate the actual use of the soon to be created PCB. Google has been the main starting source for searching for documents and hints as well as Remote Control forums for controller creation.

Lastly the Quadcopter's research mainly led to Traxxas as they are a major leader in Remote control vehicles. Googling anything RC related usually pulls up Traxxas' main page. Key points of searching through options though was battery longevity and weight of the quadcopter.

We figured buying a prebuilt platform for the Quadcopter would prove to be most efficient as it gives us easy things to replace and more time to focus on other hardware such as the glove or the host machine. Communicating with the software team to pull together the needed camera choice for the Oculus was also vital to the task.

11. Conclusion

The major key points of our project have been outlined. A majority of our time in the Summer 2015 semester was spent performing extensive research into the finer details of how we plan to create each component of our project. We went through some complications, and quickly realized that communication was going to be key to ensure we would meet our goals.

While we had some initial communication issues and misunderstandings of responsibilities we feel that we were able to work on these problems as they arose in an organized and calm manner. Through time, group members became more comfortable with each other and this facilitated communication as we were able to more uniformly combine our efforts.

The team focused their efforts on accomplishing personal milestones as much as possible, as well as assisting fellow members when they were falling behind, to avoid running into any major issues near the end of the semester. Through the research done, each group member has learned new material and techniques that they had not experienced before in the scope of their courses at UCF.

We have grown individually and as a group. Ultimately our goal has been to develop an interesting Virtual Reality experience for everyone to enjoy, and we feel that with our newly gained knowledge and support from fellow Alumni, Friends, and Family we believe we can make our designs a reality.

Now that our group has a stronger awareness of the scope of our project and the difficulties we will likely encounter as we begin building and prototyping, we plan to not slack, and continue building our current momentum to ensure we make strong progress towards not only having a functioning final build, but also to incorporate any extra features we originally wished to include in our project to make the most of our time left here at UCF.

Overall it has been a great learning experience so far, and through analyzing similar projects and others experiences with similar hardware we think we have a firm understanding of how we plan to move forward.

Further hardships came to fruition as the end of the projects deadlines came closer. Most notably were issues with the Oculus' delayed feed from the camera on the pi. Another problem was balancing out the quadcopter to fly. An even further problem was the copter motor mounts breaking at the last minute during the last demo flight. Luckily these problems and complications didn't ruin the entire project as a whole. That isn't to say they weren't noted though.

Every project comes with its own set of issues and it is our duty as engineers to work through those issues and solve the problems the best we can. We feel that is exactly what we managed to do in this project. We do hope to work in the future on the project as time permits and plan to keep up the usability of the boards and crafts we purchased. Each part has a use whether in the project or on future personal project. Waste not, want not.

11.1 Project Results

The project's total success relies the full integration of each setup coming together to complete the system. If the Oculus Rift works but the glove does not, then we are left with a quadcopter that needs to be controlled via a regular RC controller.

This will lose the objective of integration and complexity is added that the user now has to learn. If the glove works but the Oculus Rift does not, then nothing but a new controller is made and no steps towards new, exciting innovations have been made.

System Check	Pass/Fail
Quadcopter	Pass
Glove	Pass
Oculus Overlay	Pass(barely)
Quadcopter Integrated with Glove	Pass
Host Machine	Pass

Figure 11.1A Pass Fail check list

The above list is our verification that the parts worked out as we had hoped. They may have been slightly changed from what we had thought we might get a semester and a half ago but it has proven to be quite complicated. Nonetheless we believe the project has managed to pass all its designated parts and requirements.

11.1.1 Final Costs

These costs will be considered final upon project completion. It would be preferable to be able to plan the cost of this project down to the penny, but mistakes must be accounted for and so spaces must be made. Continuously updating this chart throughout our project will give us an indication as to how much we are spending.

When a pair of new gyroscopes and accelerometers cost $8, it is no big deal. When four pairs are immediately burnt on the first four attempts at creating a PCB, then research has to be done and people have to be consulted before moving on.

The parts that made the project the most expensive were the Copter kit and the Oculus itself. As development finished with the Oculus we began to realize there may have been one other cost effective idea to this and that was the use of Google Cardboard. This development may have allowed us to balance out the costs a bit better as the Cardboard is in an easier cost range of about $30 versus the Oculus' $300.

Parts	**Estimated Prices**	**Actual Costs**
Copter Components (Propellers, Hull, Motors)	Donated, Possible ~$100 upgrade	$120
Flight Controller	$60	$26
Gyroscope, Accelerometer, Sensors	$100	$40
Oculus Rift	$350	$220
Quadcopter Battery	$50	$30
Raspberry Pi Battery	$20	$30
RC transmitter/receiver	$20	$10
MicroControllers/Dev Board	$100	$120
Camera	$25	$30
Wifi Transmitter/Receiver	$50	$30
Wifi Range Extender	$25	$40
PCB	$50	$150
Total	**~$850**	**$846**

Table 11.1A: Total final Project cost

The above table is the Final Cost estimate that we came to with the project as a whole. Surprisingly we kept ourselves in budget and $4 short at that. In future perspectives of the project we may have been able to cut more corners but for this, we think we did pretty well.

11.1.2 Time Spent

The Table 11.1B will contain the total amounted hours spent on various tasks as a final amount. This will be used more for our benefit on how much time is being delegated to what aspects of the total project. This data will then lead us, to the problem's source if ever a problem arises. If no time is being spent on glove testing, then we will know that the reason the quadcopter is malfunctioning is because of the glove and not necessarily because of possible power distribution modifications.

Task	Total time (hours)
Oculus Linux Dev / Video Output Integration	35
Initial Programming For Flight Controller	20
Prototype Glove Construction	2
Prototype Glove Testing	5
Glove Construction	3
Glove Testing	1
Quadcopter and Glove Testing	10
Oculus with Quadcopter Camera Testing	1
Oculus, Quadcopter, Glove Testing	2
Documentation	20
Total Hours	99

Table 11.1B: Final Hours Totaled (will be updated on completion of project)

11.2 Moving Onwards

Even though we have the larger time frame of Fall 2015 to build and test our product we have to make sure we don't take that for granted and underestimate exactly how much work we will need to do to stay on track. We will never know if any emergencies can arise that hinder our ability to work, or even if we run into malfunctions on parts we never expected. We need to make sure any planning takes into account possible setbacks and to not treat our project goals with ideal environments in mind. As such early focused efforts will be the best way to ensure the rope around our necks has some wiggle room.

As the group now has our hands on an Oculus DK2 and HPQ1 Copter, we plan to make use of our early access and spend our time growing familiar with the products for the upcoming semester. The Copter itself we must first learn how to pilot, as well as seeing how far we can push the weight limits available to us, while still maintaining functionality, so we can include any necessary information in our user manuals and get a better understanding of what exactly our hand controller needs to be able to do to achieve as close to possible functionality of its original pre bundled RC controller.

We will have to thoroughly test the breakout board and Arduino Uno to make sure that the components we have chosen will work in the first place. If they work at the prototype stage but not at the printed circuit board stage, then we will have to look into the other gyroscope

and accelerometer combinations and see if the combined-into-one-unit electronics will work better for us in the long run.

We also will have to know whether we will need to pursue a different copter model. The sooner we can make these decisions the more time we will have available should the need arise to redesign larger components of our project. Luckily as each major aspect of our project is nearly an independent module, we have more room to adapt to critical issues and won't have to start entirely from scratch should any major changes be made.

Projected Long Term Timeline	
1 Year after Senior Design Completion	● Fix initial bottlenecks, and trim down the quadcopter design. While scaling the power and speed up or at least maintain. ● Have the host machine be portable. ● Create an easy to use GUI for every application within the project. ● Polish the project for commercial sales
2 Years	● Use our own quadcopter design if not already, by this point our drone should be rather small, suitable for indoor household use. ● Flex budget up and build smaller, lighter hardware. ● Switch / Add other VR headset support, HTC Vive, Sony's Project Morpheus ● Software expands capabilities based upon user feedback. Primarily increasing the autonomous nature of the drone.
5 Years	● Have a successful small company running entirely off of this product, expanding to either a portable personal version, or a custom designed flight arena. ● At this point the software should be in a mature state that mostly support work is happening.(Assuming project's scope never changes) ● Some really drastic changes could be implemented now due to the growth of the tech market.

Table 11.2A Projected Future Timeline

We plan to get our hands on some Raspberry Pi's early on and begin getting more hands on experience with the interfacing of the cameras and such before the semester begins. We will need to get a strong familiarity with the planned software involved and their

functionalities. Once we have the Pi's and cameras we can begin testing the limitations of their power for encoding and seeing what things we need to start cutting back on, whether we need to use lower resolutions than initially planned for example.

We also want to kickstart our progress into working on the most difficult component of our project , the Hand Controller. If we could enhance its capabilities to be used in other interfaces as a standalone project, that could be the sole greatest aspect of the project. Combining it with other virtual reality equipment could prove to be a hugely profitable endeavor.

Appendix and References

References

Consulted:

2.2

http://static.oculus.com/documents/health-and-safety-warnings.pdf

http://www.censusscope.org/us/s12/c95/chart_age.html

3.2

http://knowbeforeyoufly.org/for-recreational-users/

http://www.multiwii.com/

http://copter.ardupilot.com/

4.1

http://www.eecs.ucf.edu/seniordesign/fa2013sp2014/g06/uploads/2/8/7/8/28781263/senior_design_i_documentation.pdf

http://www.eecs.ucf.edu/seniordesign/fa2013sp2014/g07/

4.1.1

http://torkeldanielsson.se/live-twocamera-video-stream-from-raspberry-pi-to-oculus-rift

5.2

https://www.oculus.com/en-us/dk2/

5.3

https://support.oculus.com/hc/en-us/articles/201835987-Oculus-Rift-Development-Kit-2-FAQ

6.1-6.2.2

http://rotorconcept.com/Documents/aHPQ1%20RotorConcept%20Manual%20003.JPG

http://www.hovership.com/2012/12/06/beginner-quadcopter-kit-buying-guide/

http://www.hobbyking.com/hobbyking/store/__54299__Hobbyking_KK2_1_5_Multi_rotor_LCD_Flight_Control_Board_With_6050MPU_And_Atmel_644PA.html

https://www.raspberrypi.org/forums/viewtopic.php?f=37&t=35746

https://latrax.com/sites/default/files/6608-parts-list-140428.pdf

https://latrax.com/products/alias?t=specs

http://www.ebay.com/itm/like/281566912485?lpid=82&chn=ps

[1] http://blog.oscarliang.net/how-to-choose-motor-and-propeller-for-quadcopter/

http://thequadcopterguy.blogspot.com/p/choosing-your-parts_23.html

http://www.banggood.com/8045-3-Leaf-Propeller-ABS-CWCCW-For-Quadcopter-330-Frame-Kit-p-954478.html

6.2.3

http://www.eflightwiki.com/eflightwiki/index.php?title=Outrunner

*** [2] http://www.eflightwiki.com/eflightwiki/index.php?title=Rotor_Concept_HPQ1

6.2.4

http://www.rcgroups.com/forums/showthread.php?t=1822983

http://www.hobbyking.com/hobbyking/store/__9516__Turnigy_5000mAh_4S_30C_Lipo_Pack.html

7.2.1
https://learn.adafruit.com/ahrs-for-adafruits-9-dof-10-dof-breakout

http://www.adafruit.com/products/1714

https://www.verical.com/pd/stmicroelectronics-sensor-misc-LSM303DLHC-369341?wm_ctID=250&wm_kwID=70581028&wm_mtID=1&gclid=Cj0KEQjwiN6sBRDK2vOO_vaRs5cBEiQAfsnJCZLeKmAioGAtkiolcsThdklcjPUMAkXi3z8e_sHmR2EaAk3a8P8HAQ

http://www.newark.com/stmicroelectronics/l3gd20h/3-axis-gyroscope-digital-lga-16/dp/78X5255?mckv=sPkmjxXGy_dc|pcrid|72465880475|plid||kword|l3gd20h|match|p&CMP=KNC-GUSA-SKU-MDC?gross_price=

7.2.1 block diagrams:
http://download.siliconexpert.com/pdfs/2013/12/18/13/33/10/666/st_/manual/215dm00027543.pdf

http://www.farnell.com/datasheets/1836728.pdf

7.2, 7.2.5

https://www.arduino.cc/en/Main/arduinoBoardUno

8
https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

http://www.alliedelec.com/raspberry-pi-raspberry-pi-2-model-b/70465426/

http://www.adafruit.com/product/1030

8.1

https://www.raspberrypi.org/documentation/hardware/raspberrypi/models/specs.md

8.2.1

http://www.newegg.com/Product/Product.aspx?Item=N82E16813130847&cm_re=z87-_-13-130-847-_-Product

8.2.2

http://ark.intel.com/products/75123/Intel-Core-i7-4770K-Processor-8M-Cache-up-to-3_90-GHz

8.2.4

http://www.newegg.com/Product/Product.aspx?Item=9SIA24G15S7606&cm_re=samsung_ssd_840_pro-_-20-147-192-_-Product

8.2.5
http://www.newegg.com/Product/Product.aspx?Item=N82E16814487075&cm_re=gtx_970-_-14-487-075-_-Product

10.1

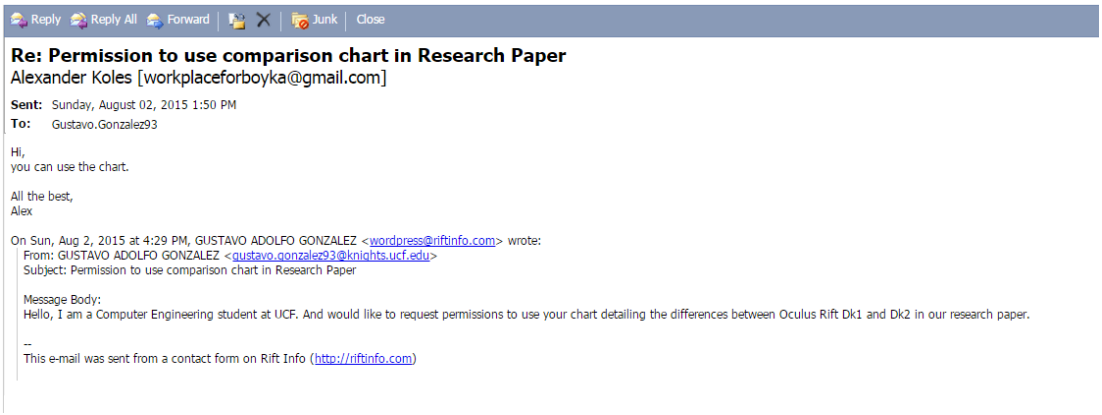
Designated Jira site PatientPoint Infrastructure Swimlane
http://jirappns.patientpoint.com:8080/

https://confluence.atlassian.com/download/attachments/185729618/jira_default_workflow.png?version=1&modificationDate=1378968996981&api=v2

10.1.4
https://git.wiki.kernel.org/index.php/GitSvnComparsion

Appendix

permissions for oculus tables 4.1.8a & 4.1.8b



Reply | Reply All | Forward | Junk | Close

Re: Permission to use comparison chart in Research Paper
Alexander Koles [workplaceforboyka@gmail.com]

Sent: Sunday, August 02, 2015 1:50 PM
To: Gustavo.Gonzalez93

Hi,
you can use the chart.

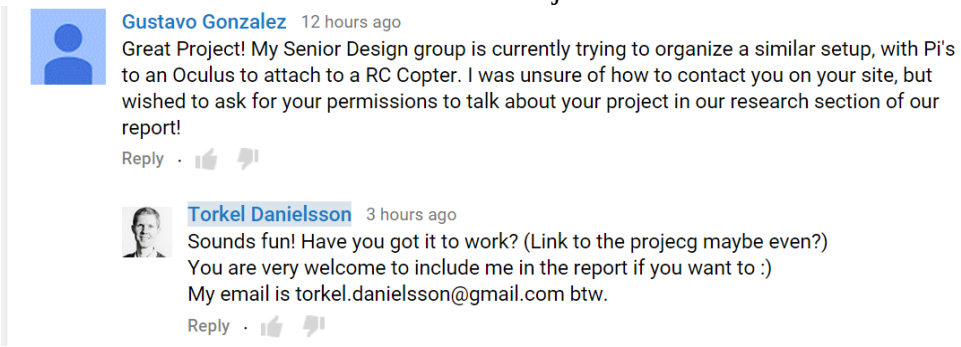
All the best,
Alex

On Sun, Aug 2, 2015 at 4:29 PM, GUSTAVO ADOLFO GONZALEZ <wordpress@riftinfo.com> wrote:
From: GUSTAVO ADOLFO GONZALEZ <gustavo.gonzalez93@knights.ucf.edu>
Subject: Permission to use comparison chart in Research Paper

Message Body:
Hello, I am a Computer Engineering student at UCF. And would like to request permissions to use your chart detailing the differences between Oculus Rift Dk1 and Dk2 in our research paper.

--
This e-mail was sent from a contact form on Rift Info (http://riftinfo.com)

Permissions for mention in 4.1 Related Projects



Gustavo Gonzalez 12 hours ago
Great Project! My Senior Design group is currently trying to organize a similar setup, with Pi's to an Oculus to attach to a RC Copter. I was unsure of how to contact you on your site, but wished to ask for your permissions to talk about your project in our research section of our report!
Reply ·

Torkel Danielsson 3 hours ago
Sounds fun! Have you got it to work? (Link to the projecg maybe even?)
You are very welcome to include me in the report if you want to :)
My email is torkel.danielsson@gmail.com btw.
Reply ·

Permission for DroneNet project

BF Brandon Frazer <Brandon.frazer@live.com>
Sun 8/2/2015 2:04 PM

Mark as unread

To: ☐ Joseph Howard <joeh.cobra@gmail.com>; ■ Matthew Grayford;
Cc: ☐ gusgodzilla@gmail.com; ☐ rtljr76@gmail.com; ☐ Ryanborden23@gmail.com;

Matthew,

Feel free to use any info that will help. Good luck, that sounds like a cool project!

Sent from my Verizon Wireless 4G LTE smartphone

-------- Original message --------
From: Joseph Howard <joeh.cobra@gmail.com>
Date: 08/02/2015 2:02 PM (GMT-05:00)
To: Matthew Grayford <dragonbolt1@knights.ucf.edu>
Cc: gusgodzilla@gmail.com, rtljr76@gmail.com, Brandon.frazer@live.com, Ryanborden23@gmail.com
Subject: Re: Senior Design mention request

Matthew,

Good to go dude! Good luck! That sounds like a cool project. If you have any questions about the Quad, let me know.

Joe

On Aug 2, 2015 10:24 AM, "Matthew Grayford" <dragonbolt1@knights.ucf.edu> wrote:

> Hey Guys,
>
> I was running through previous senior design projects and was wondering if I could mention yours in our senior design project. We're doing an integration of Quadcopters, a glove and the oculus rift for a new user experience. I wanted to mention your use of battery choice and extension for the quadcopter. If one of you could reply and let me know if thats fine I can add it to our paper.

AM Ali Mizan <ali.mizan@gmail.com>
Sun 8/2/2015 9:26 PM

Mark as unread

I'm fine with it. best of luck on your project!

←Reply ←←Reply all →Forward •••

MG Matthew Grayford
Sun 8/2/2015 1:21 PM
Sent Items

Mark as unread

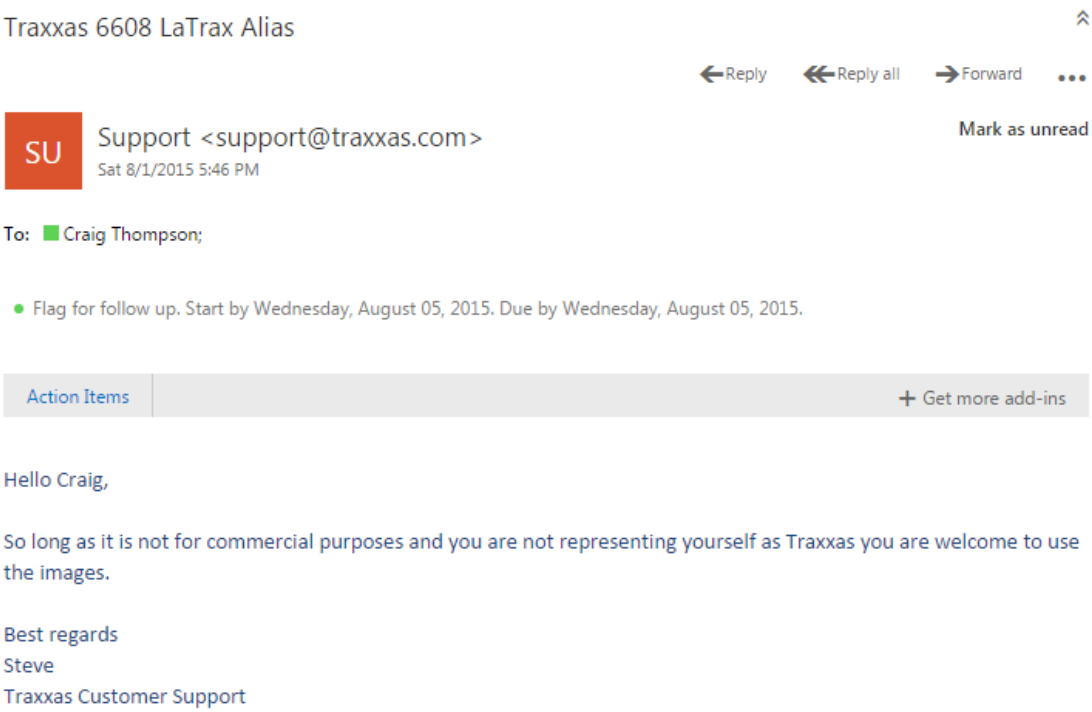
To: ☐ braintroili; ☐ lalis.rubiete; ☐ ali.mizan@gmail.com; ☐ Kirk Chan;
Cc: ☐ gusgodzilla@gmail.com;

Hey Guys,

I'm working with my team on my senior design project and was reading through the list of previous projects and found yours.
I wanted to ask if it was ok to mention your project within ours as a reference. More so for technicalities... as we're not copying or using any details from your project.

Thanks and hope to hear back within the next day.
Matt Grayford

Permission for High 6 project mentioned in section 4


Traxxas 6608 LaTrax Alias

Reply Reply all Forward

SU

Support <support@traxxas.com>

Sat 8/1/2015 5:46 PM

Mark as unread

To: Craig Thompson;

Flag for follow up. Start by Wednesday, August 05, 2015. Due by Wednesday, August 05, 2015.

Action Items

Get more add-ins

Hello Craig,

So long as it is not for commercial purposes and you are not representing yourself as Traxxas you are welcome to use the images.

Best regards
Steve
Traxxas Customer Support

Permission to use Traxxas 6608 Image