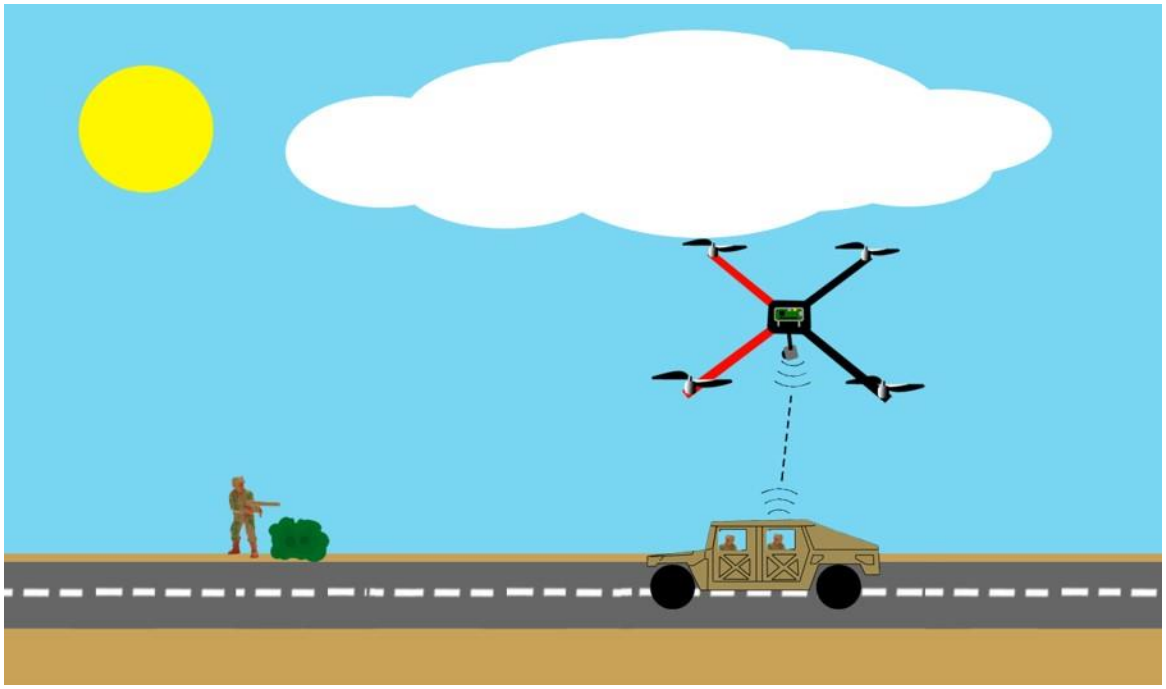
EAGLEYE

Team 08, 2013-2014



Drew Brandsen
Robert Hoff
Spencer Olson
Jared Zoodsma

Engineering 339/340 Senior Design Project
Calvin College
May 15, 2014

EXECUTIVE SUMMARY

Security is paramount for presidential motorcades, military convoys, and other safety critical transport. Visibility is often impaired for those inside the vehicles, making it difficult to see all possible threats ahead, behind, and to the sides. Lack of visibility creates a significant danger from insurgents, rocket propelled grenades (RPGs), improvised explosive devices (IEDs), and other threats. Moreover, travel routes can span hundreds of miles. Covering these routes with explosive detectors, bomb-sniffing dogs, and law enforcement is very costly and requires extra labor. Improving visibility for individuals in the vehicles can help mitigate these external risks

The solution our team of four electrical and computer engineers has chosen consists of two subsystems: one to fly above the vehicle and a second installed inside the vehicle. We considered various options for the first subsystem and chose a quadcopter because of its high expandability, maneuverability, and stability. The quadcopter is equipped with video capture equipment, as well as a global positioning system (GPS) receiver. Second, a base station located in the vehicle includes a computer, Wi-Fi router, GPS receiver, radio controlled (RC) transmitter, and RC interface module. The base station also contains a graphical user interface (GUI) that receives the video streamed from the quadcopter and provides a way for the user to interact with the system.

The design process entailed building a quadcopter and expanding it to fit the project specifications. Some of the key specifications were providing a video stream of the vehicle and its surroundings, being small enough to fit the entire system in the vehicle, and being able to track the vehicle autonomously. This involved adding both a camera and a Raspberry Pi development board to the quadcopter in order to capture and send the quadcopter's GPS location and video stream over Wi-Fi to the base station below. The base station computer, with the assistance of an Arduino microcontroller, used the GPS data in order to autonomously control the quadcopter via RC.

The Eagleye team showed the feasibility of this solution through first semester research and preliminary designs. Following this, the team designed a working prototype in the second semester. The team operated under a budget of $1455 in order to complete this work. The team hopes that the work done on this project will increase awareness of unmanned aerial vehicles (UAVs) and their beneficial applications.

TABLE OF CONTENTS

1 INTRODUCTION 1

1.1 CALVIN COLLEGE ENGINEERING DEPARTMENT 1
1.2 TEAM MEMBER BIOGRAPHIES 1
1.2.1 Jared Zoodsma 2
1.2.2 Robert Hoff 2
1.2.3 Drew Brandsen 2
1.2.4 Spencer Olson 2
1.3 REPORT FORMAT 2
1.4 DESIGN NORMS 3
1.4.1 Transparency 3
1.4.2 Justice 3
1.4.3 Trust 3
1.5 ACKNOWLEDGEMENTS 3

2 PROJECT REQUIREMENTS 5

2.1 PROBLEM STATEMENT 5
2.1.1 Solution 5
2.1.2 Target Customers 5
2.2 CUSTOMER REQUIREMENTS 5
2.2.1 Performance 5
2.2.2 Physical 9
2.2.3 Customer 9
2.3 DELIVERABLES 11

3 SYSTEM DESIGN 12

3.1 DECISION MATRIX 12
3.1.1 Possible Solutions 12
3.1.2 Decision Criteria and Justification 15
3.1.3 Justification 16
3.1.4 Best Solution 21
3.2 OVERALL SYSTEM BLOCK DIAGRAM 21
3.2.1 Hardware 21
3.2.2 Software 23
3.3 FEATURE MATRIX 25
3.3.1 Feature Descriptions 26
3.4 FEATURE RESULTS 29
3.4.1 Feature Descriptions 29

4 QUADCOPTER DESIGN 31

4.1 QUADCOPTER ARCHITECTURE 31
4.2 QUADCOPTER COMPONENTS 34
4.2.1 Flight Controller 35
4.2.2 GPS Module 39
4.2.3 Propellers 40
4.2.4 Motors 41
4.2.5 ESCs 42
4.2.6 Battery 43
4.2.7 Power Distribution System 46
4.2.8 Frame and Landing Gear 47
4.2.9 VGPU 49
4.2.10 Camera 57

5 BASE STATION DESIGN 59

5.1 BASE STATION ARCHITECTURE 59
5.2 BASE STATION COMPONENTS 62
5.2.1 Base Station Computer 62
5.2.2 RC Interface Module 69
5.2.3 RC Transmitter 82
5.2.4 GPS Module 83
5.2.5 Wireless Router 85

6 SYSTEM INTEGRATION AND TESTING 87

6.1 TELEMETRY STREAMING 87
6.1.1 Test: Full System Communication 87
6.1.2 Test: Communication with Live Telemetry Coordinates 87
6.1.3 Communication Latency 88
6.2 VIDEO STREAMING 89
6.2.1 Video Latency 89
6.2.2 Video Quality 90
6.3 GPS TRACKING 91
6.3.1 Stationary GPS Precision 91
6.3.2 Moving GPS Precision 92
6.4 RC CONTROL VIA RC INTERFACE MODULE 95
6.4.1 RC Pass Through 95
6.4.2 Pitch and Roll Recording 95
6.4.3 Steps 96
6.4.4 Autonomous Pitch and Roll Start 96
6.4.5 Autonomous Pitch Stop 97
6.5 TRACKING ALGORITHM 97
6.5.1 RC Response Test 97
6.5.2 Motor Response Test 98
6.5.3 North-Facing Flight Tracking 99
6.5.4 Orientation Tracking 100

7 PROJECT MANAGEMENT 103

7.1 DOCUMENTATION ORGANIZATION 103
7.1.1 Code 103
7.1.2 Design Journals 103
7.1.3 Manuals 103
7.1.4 Media 103
7.1.5 Miscellaneous Notes 103
7.1.6 Planning Documents 103
7.1.7 Status Reports 104
7.1.8 Team 08: Turn-in 104
7.1.9 Tests 104
7.2 TEAM ORGANIZATION 104
7.2.1 Technical Assignments (Design Areas) 104
7.2.2 Management Assignments 106
7.2.3 Working Guidelines 106
7.2.4 Safety Guidelines 107
7.2.5 Team Communication and Accountability 107
7.3 SCHEDULE AND WORK BREAKDOWN SCHEDULE 107
7.3.1 Hours Summary 107
7.3.2 Milestones 108
7.4 OPERATIONAL BUDGET 109
7.4.1 Project Costs 109

7.4.2 Summary *110*
7.4.3 Sources of Funding *110*
7.5 METHOD OF APPROACH 110
7.5.1 Design Methodology *110*
7.5.2 Research Techniques *110*

8 BUSINESS PLAN 111

8.1 OVERVIEW 111
8.2 VISION AND MISSION STATEMENT 111
8.3 VISION FOR THE COMPANY 111
8.4 VALUES AND PRINCIPLES ON WHICH THE BUSINESS STANDS 111
8.5 MARKETING STUDY 111
8.5.1 How Large is the Market? *111*
8.5.2 Is it Growing or Shrinking? *112*
8.5.3 Regulatory Restrictions *112*
8.5.4 Barriers to Entry and Exit *112*
8.5.5 Key Success Factors in the Industry *112*
8.6 COMPETITION 112
8.6.1 Existing Competitors *112*
8.6.2 Potential Competitors *113*
8.7 BUSINESS MODEL 114
8.7.1 Desired Image and Position in the Market *114*
8.7.2 Company Goals and Objectives *114*
8.7.3 SWOT Analysis *114*
8.7.4 Competitive Strategy *115*
8.8 FINANCIAL FORECASTS 116
8.8.1 Key Assumptions *116*
8.8.2 Financial Statements *117*
8.8.3 Break-Even Analysis *119*
8.8.4 Ratio Analysis *119*
8.9 SUMMARY 120

9 CONCLUSION 121

9.1 PROJECT RESULTS 121
9.1.1 Overall *121*
9.1.2 Final Costs *121*
9.1.3 Time Spent *121*
9.2 FUTURE WORK 121
9.3 SUMMARY 122

TABLE OF FIGURES

FIGURE 1: TEAM PHOTO - JARED ZOODSMA, DREW BRANDSEN, ROBERT HOFF, AND SPENCER OLSON. 1
FIGURE 2: 2-D HORIZONTAL DISTANCE REQUIREMENT FOR PROTOTYPE 1.0 6
FIGURE 3: QUADCOPTER OUTFITTED WITH CAMERAS 12
FIGURE 4: RC HELICOPTER 13
FIGURE 5: RC AIRPLANE 13
FIGURE 6: RC BLIMP 14
FIGURE 7: PANORAMIC BALL CAMERA 14
FIGURE 8: CONTROL OF A QUADCOPTER 18
FIGURE 9: PLATFORM AREA OF A QUADCOPTER 20
FIGURE 10: OVERALL SYSTEM HARDWARE BLOCK DIAGRAM 22
FIGURE 11: OVERALL SOFTWARE SYSTEM BLOCK DIAGRAM 24
FIGURE 12: QUADCOPTER PROTOTYPE 1.0 31
FIGURE 13: QUADCOPTER HARDWARE BLOCK DIAGRAM 32
FIGURE 14: QUADCOPTER SOFTWARE BLOCK DIAGRAM 33
FIGURE 15: CENTRAL QUADCOPTER COMPONENTS 34
FIGURE 16: COMPASS VALUES WITH HEADING-HOLD MODE 38
FIGURE 17: PROTOTYPE 1.0 MOTOR DIRECTION 42
FIGURE 18: PROTOTYPE 1.0 FLIGHT TIMES WITH VARIOUS BATTERY SIZES 45
FIGURE 19: BROKEN ARM 48
FIGURE 20: PROPELLER GUARDS WITH VIBRATION PROBLEMS (PROTOTYPE 0.2) 48
FIGURE 21: FINAL PROPELLER GUARD IMPLEMENTATION (PROTOTYPE 1.0) 49
FIGURE 22: CUSTOM VGPU LAYOUT 50
FIGURE 23: VGPU SOFTWARE BLOCK DIAGRAM 52
FIGURE 24: RASPBERRY PI DIGITAL I/O PINS 56
FIGURE 25: MULTIWII SERIAL PROTOCOL 56
FIGURE 26: MULTIWII PRO CONNECTION 57
FIGURE 27: FINAL BASE STATION IMPLEMENTATION 59
FIGURE 28: BASE STATION HARDWARE BLOCK DIAGRAM 60
FIGURE 29: BASE STATION SOFTWARE BLOCK DIAGRAM 61
FIGURE 30: BASE STATION COMPUTER SOFTWARE BLOCK DIAGRAM 63
FIGURE 31: FINAL GUI 65
FIGURE 32: RC INTERFACE MODULE HARDWARE BLOCK DIAGRAM 69
FIGURE 33: CUSTOM RC INTERFACE MODULE LAYOUT 71
FIGURE 34: FILTER BOARD CIRCUIT 74
FIGURE 35: RC INTERFACE MODULE OUTPUT DIAGRAM 75
FIGURE 36: FILTER BOARD LAYOUT 76
FIGURE 37: FILTER BOARD PHOTO 76
FIGURE 38: RC INTERFACE MODULE SOFTWARE BLOCK DIAGRAM 77
FIGURE 39: DISTANCE DEPENDENT SPEED METHOD 80
FIGURE 40: ORIENTATION TRACKING 81
FIGURE 41: RC INTERFACE MODULE EXTERNAL PHOTO 84
FIGURE 42: RC INTERFACE MODULE INTERNALS 84
FIGURE 43: VIDEO LATENCY RESULTS 90
FIGURE 44: VIDEO PROTOCOL QUALITY SCORES 91
FIGURE 45: MOVING GPS PRECISION TEST (22.35 M/S, 10 HZ) 93
FIGURE 46: MOVING GPS PRECISION TEST (16.9 M/S, 10 HZ) 94
FIGURE 47: ORIENTATION FLIGHT TEST 101
FIGURE 48: GRAPH OF HOURS WORKED ON PROJECT 108

TABLE OF TABLES

TABLE 1: PROJECT DELIVERABLES 11
TABLE 2: DECISION MATRIX 15
TABLE 3: SYSTEM HARDWARE BLOCK DIAGRAM INTERFACE DESCRIPTIONS 22
TABLE 4: SYSTEM SOFTWARE BLOCK DIAGRAM INTERFACE DESCRIPTIONS 24
TABLE 5: SYSTEM PROTOCOL DEFINITIONS 25
TABLE 6: FEATURE MATRIX 25
TABLE 7: FEATURE MATRIX IMPLEMENTATION 29
TABLE 8: QUADCOPTER HARDWARE BLOCK DIAGRAM INTERFACE DESCRIPTIONS 32
TABLE 9: QUADCOPTER SOFTWARE BLOCK DIAGRAM INTERFACE DESCRIPTIONS 34
TABLE 10: FLIGHT CONTROLLER ALTERNATIVES 35
TABLE 11: ALTITUDE-HOLD TESTS WITH CONTROL AND FOAM COVERS (METERS) 36
TABLE 12: BATTERY COMPARISONS 44
TABLE 13: BATTERY CAPACITIES AND FLIGHT TIMES 45
TABLE 14: COMPONENT POWER NEEDS 46
TABLE 15: VGPU SOFTWARE BLOCK DIAGRAM INTERFACE DESCRIPTIONS 52
TABLE 16: VGPU PROTOCOLS 52
TABLE 17: BASE STATION HARDWARE INTERFACE DESCRIPTIONS 60
TABLE 18: BASE STATION SOFTWARE BLOCK DIAGRAM INTERFACES 62
TABLE 19: BASE STATION SOFTWARE PROTOCOLS 62
TABLE 20: BASE STATION COMPUTER SOFTWARE BLOCK DIAGRAM INTERFACES 64
TABLE 21: GUI SOFTWARE PROTOCOLS 64
TABLE 22: RC INTERFACE MODULE HARDWARE BLOCK DIAGRAM INTERFACES 69
TABLE 23: RC INTERFACE MODULE DEVELOPMENT BOARD ALTERNATIVES 70
TABLE 24: RC INTERFACE MODULE SOFTWARE BLOCK DIAGRAM INTERFACE DESCRIPTIONS 78
TABLE 25: ANOTHER GPS STATIONARY RESULT (10 HZ) 92
TABLE 26: MOVING GPS PRECISION TEST (1 HZ) 93
TABLE 27: MOVING GPS PRECISION TEST (22.35 M/S, 10 HZ) 94
TABLE 28: MOVING GPS PRECISION TEST (16.9 M/S, 10 HZ) 94
TABLE 29: EXPECTED REACTIONS TO POSITION CHANGES 98
TABLE 30: DESIGN AREAS AND TEAM MEMBER ASSIGNMENTS 105
TABLE 31: SUMMARY OF HOURS 107
TABLE 32: TEAM 08: EAGLEYE BUDGET 109
TABLE 33: SUPPORTING CALCULATIONS 117
TABLE 34: INCOME STATEMENT 118
TABLE 35: CASH FLOW STATEMENT 118
TABLE 36: BREAK-EVEN ANALYSIS 119
TABLE 37: RATIO ANALYSIS 120

TABLE OF ACRONYMS

ABET: Accreditation Board for Engineering and Technology
API: Application programming interface
APM: ArduPilot Mega
BEC: Battery Eliminator Circuit
ESC: Electronic Speed Controller
FPS: Frames per Second
GPS: Global Positioning System
GUI: Graphical User Interface
HD: High Definition
IED: Improvised Explosive Device
I/O: Input/Output
IR: Infrared
Kv: Revolutions per Minute / Volt
MSP: MultiWii Serial Protocol
MW: MultiWii
NMEA: National Marine Electronics Association
PCB: Printed Circuit Board
PID: Proportional-Integral-Derivative
PPFS: Project Proposal and Feasibility Study
PPS: Picture Parameter Set
PWM: Pulse Width Modulation
OS: Operating System
Q&A: Question and Answer
RAM: Random Access Memory
RC: Radio Controlled
RPG: Rocket Propelled Grenade
SD: Standard Definition
SPS: Sequence Parameter Set
SSH: Secure Shell
SWOT: Internal Strengths and Weaknesses, External Opportunities and Threats.
TCP/IP: Transmission Control Protocol/Internet Protocol
U.S.: United States of America
UAV: Unmanned Aerial Vehicle
USB: Universal Serial Bus
VDC: Volts Direct Current
VGPU: Video and Graphics Processing Unit
WBS: Work Breakdown Structure

1 Introduction

The Calvin College Engineering Program concludes its four-year degree with a Senior Design Capstone project. This project spans the entire senior year, giving students the opportunity to use knowledge and skills accumulated throughout their academic careers. It also allows students to work on a project in their area of interest, from conception to final prototype. Teams of three or four students are formed around a common project idea. These teams may contain students from similar or different concentrations, as the project allows. The Project Proposal and Feasibility Study explored the feasibility of our chosen project, provided a structured plan for the design, and was a milestone for the project at the end of first semester. This Final Design Report discusses the design details of the project by displaying various decisions the team made throughout the design process. Lastly, it discusses future design work the team would like to do in the next prototype stage.

1.1 Calvin College Engineering Department

Calvin College's Engineering Program is ABET (Accreditation Board for Engineering and Technology) accredited and offers a Bachelor of Science in Engineering with concentrations in Chemical, Civil/Environmental, Electrical/Computer, and Mechanical engineering. The program also offers the option of an international designation with any of the concentrations mentioned above. The program "combines the foundation of a world-class liberal arts curriculum and adds the cutting-edge knowledge of technology in engineering." The program also "challenge[s] you to do all of [engineering] from a reformed Christian worldview."[1]

1.2 Team Member Biographies

The Eagleye team is composed of four senior engineering students in the Electrical & Computer concentration. The team learned about a safety application involving autonomous tracking from their faculty advisor, so they decided to pursue it due to their passion for autonomous vehicles and quadcopters. Figure 1 shows the team picture.

Figure 1: Team Photo - Jared Zoodsma, Drew Brandsen, Robert Hoff, and Spencer Olson.

Below is a brief biography of each team member:

1.2.1 Jared Zoodsma

Jared hails from Grandville, Michigan where he attended Calvin Christian High School. Jared served as the team financial manager and was in charge of researching and ordering components. In addition to his team role, Jared primarily focused on creating a graphical user interface. Work on the GUI included streaming video, sending GPS coordinates from the quadcopter to the Arduino, and providing a way for the user to interact with the system. Jared enjoys working on his car, following Michigan football, and playing sports such as soccer, ultimate, and golf. After graduation, Jared will begin his career working at GE Aviation in Grand Rapids, Michigan.

1.2.2 Robert Hoff

Robert is from Bloomington, Illinois and served as team marketing manager. Robert was in charge of team media including posters, website updates, and all other graphical design. Robert's technical design areas were the tracking algorithms and the RC control. The tracking algorithm compares GPS locations between the vehicle and quadcopter and autonomously controls the quadcopter over RC. Robert enjoys tinkering with electronics, learning about technology, and is an amateur photographer in his free time. Robert has accepted a job at Northrop Grumman in Rolling Meadows, Illinois.

1.2.3 Drew Brandsen

Drew calls the great lakeside city of Holland, Michigan home. He served as the team professional officer; assuring that all documents and materials were organized and of the utmost quality. On the technical side, Drew was responsible for identifying and integrating quadcopter hardware and software. Specifically, he created the software to interface between the quadcopter telemetry information and the GUI. Drew enjoys spending time with friends, sailing, and playing intramural sports, and after graduation will begin work with Northrop Grumman in June.

1.2.4 Spencer Olson

Acting as Project Manager, Spencer was in charge of keeping the team on track for deadlines set by the faculty adviser, as well as setting intermediate deadlines. He was also in charge of the video streaming and integration into the GUI, as well as the tracking algorithms and resulting flight controls from the base station. In his free time, Spencer enjoys playing golf, camping, running, and watching sports. Spencer will begin work this summer at Epic, in Madison, Wisconsin.

1.3 Report Format

This report details the design process of the project. Below are listed each of the chapters in the report, along with a brief description. References are available at the end of this document.

- **Chapter 2: Project Requirements** - Covers the requirements of the project, both in terms of a final product as well as the prototype that the team completed.

- **Chapter 3: System Design** - Evaluates different platforms for implementing a solution based on design criteria and project requirements. This chapter shows how the team laid out their design goals for the project.

- **Chapter 4: Quadcopter Design** - Discusses the quadcopter design and breaks down the requirements, alternatives, decision criteria, and final decision for each quadcopter component; hardware and software. This chapter also discusses implementation and tests for each component.

- **Chapter 5: Base Station Design** - Discusses the base station design. This chapter also breaks down each component for its requirements, alternatives, decision criteria, and final decision for both hardware and software. Each component also has implementation and test details.

- **Chapter 6: System Integration and Testing** - Discusses how the various subsystems were integrated, and how testing determined if each of those subsystems were performing their respective tasks. The team also completed testing on the entire system to ensure its performance met the requirements.

- **Chapter 7: Project Management** - This chapter explains the roles of each member of the project. In addition, it outlines project organization, budget, and timeline.

- **Chapter 8: Business Plan** - The business plan discusses a potential startup company and evaluates how this company could make a profit by selling this product.

- **Chapter 9: Conclusion** - The final chapter reviews the design work completed and concludes with potential future work for the project.

1.4 Design Norms

Design norms are virtues used to shape a design process in a way that aligns with our Biblical teachings at Calvin College. During the course of the project, the team took to heart three main design norms that shaped the project. Below are short descriptions of each of the design norms. The team attempted to integrate these design norms into all of our decisions. There are sometimes conflicts attempting to satisfy these design norms, however. For example, providing a high level of justice may result in lowered transparency. If this project was used by the United States (U.S.) military, the information would become classified and no longer available to the public. This creates tension within these norms that necessitated a balancing act by the team as work proceeded on the project.

1.4.1 Transparency

Transparency means that the design should be open about the communication and be understandable throughout the design process. Transparency came into play as video-taking UAVs are commonly highlighted in privacy concerns. The team took all actions necessary to prove that the video taken is for responsible uses and does not show disrespect toward another party.

1.4.2 Justice

The system should be a means of achieving justice, specifically by providing additional security for individuals who could be in life or death situations. The team made it a goal to protect those who protect and lead us.

1.4.3 Trust

The system must be dependable in all situations. This means that within the defined specifications, the device will function reliably and effectively

1.5 Acknowledgements

The team would like to take this opportunity to thank those who have helped make this project a success. First, the team would like to thank DornerWorks for sponsoring the project through a donation. This donation from DornerWorks was been a tremendous step in helping fund this project. In addition, the team would like to thank their faculty advisor, Professor Steve VanderLeest, who was influential in

providing guidance and direction to the project. Next, the group would like to thank the Calvin College Engineering program as a whole for supporting successful projects with finances and challenging students to strive toward stronger projects. The team would also like to thank Professor Yoon Kim for allowing them to run tests on his quadcopter and use parts of it for their project. Professor Kim also donated several quadcopter components, which helped the team stay under budget. Justin TeBrake is another individual who donated a net to the team and deserves gratitude. The team is also thankful to Andrew Jo for his significant contribution to the business plan. Lastly, the team would like to thank our families for their continued love and support in our education and personal development.

2 Project Requirements

This chapter identifies potential customers, defines customer requirements, and establishes a list of deliverables for the customer. Following the requirements, the next chapter will discuss in detail the team's solution to solving this problem.

2.1 Problem Statement

Security is paramount for presidential motorcades, military convoys, and other safety critical transports. Visibility is often impaired for those inside the vehicles, making it difficult to see all possible threats ahead, behind, and to the side. Lack of visibility creates a significant danger from insurgents, RPGs, IEDs, and other threats. Travel routes can span hundreds of miles and covering these routes with explosive detectors, bomb-sniffing dogs, or law enforcement is costly, but still does not guarantee complete safety. Improved visibility for individuals in the vehicles can help mitigate these external risks

2.1.1 Solution

The solution that our team proposes provides a system that shadows a motorcade or convoy from above in order to provide a bird's-eye view of their surroundings, enabling users to spot would-be attackers, IEDs and other threats. Providing this real time overhead vantage point will increase the situational awareness of those in the vehicles. This additional information provides one more level of security that ultimately leads to justice and safety for those who need protection.

2.1.2 Target Customers

Our solution targets organizations or individuals with a need for security when traveling by land. Our customers will include diplomats, the Secret Service and the U.S. Military to name a few. The team will design the system to bolster security for those with this need, particularly those who serve our country. The team considers it a just cause to protect our customer from malicious attack.

2.2 Customer Requirements

This section articulates customer-defined requirements by sub-sections such as performance, mechanical features, and customer use. The team also identified requirements used to build prototype 1.0, which was delivered on May 10, 2014. These requirements are defined generally here, and are applied more specifically for the chosen solution in chapters four and five.

2.2.1 Performance

These requirements define the performance of the system such as flight time and maneuverability.

2.2.1.1 Flight time

REQ 2.2.1.1.A: The airborne component of the system shall stay aloft for a minimum time of thirty minutes.

This time is viewed as the longest journey that our customer might take where they would expect the product to operate continuously without a recharge. Anything less and the customer would likely consider it difficult and cumbersome to use.

REQ 2.2.1.1.A.P1 Prototype 1.0 shall stay aloft for ten minutes.

Due to commercially available batteries for this application, one can achieve this standard time. Ten minutes also proves that with more time and money, a custom battery could be built to increase this flight time to thirty minutes.

2.2.1.2 Flight Performance

REQ 2.2.1.2.A The airborne component of the system shall move from one point in three-dimensional space to another in a straight line.

This is necessary in order to provide a high degree of maneuverability.

REQ 2.2.1.2.B The airborne component's airspeed shall be no less than 100 km/hr (62 MPH).

100 km/hr was determined to be the maximum speed at which vehicles would move through populated, urban areas where an attacker would be able to hide.

> REQ 2.2.1.2.B.P1 Prototype 1.0 shall have a top airspeed of no less than 15 km/hr (9.3 MPH).
>
> A speed of 15 km/hr will allow the team to produce a prototype that demonstrates the product's potential and provides a good platform for testing – without purchasing larger, more expensive motors to reach higher speeds.

REQ 2.2.1.2.C The solution shall have a 2-D tracking tolerance of 1.8 meters (6 ft) from a point horizontal from the source location to a point directly vertical of the destination location. This requirement is applied for both a moving and stationary vehicle.

This tolerance was selected to be half the width of a standard road lane size.[2] Thus, the system will stay within the same lane as the vehicle it is following. This is required so that the system keeps a centered view of the vehicle and its surroundings during flight.

> REQ 2.2.1.2.C.P1 The solution shall have a 2-D tracking tolerance of 7.2 meters (23.6 ft) from a point horizontal from the source location to a point directly vertical of the destination location.
>
> This is the width of a standard two-land road.[2] This was chosen for testing purposes to prove the system can track a moving vehicle on common traffic roadways. See Figure 2 for a visual depiction.

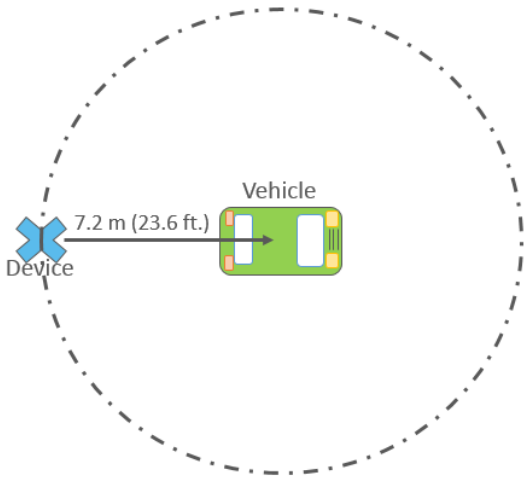


Figure 2: 2-D Horizontal Distance Requirement for prototype 1.0

REQ 2.2.1.2.D The system shall have an altitude tracking tolerance no greater than one meter (3.28 ft).

Since the camera will view from horizon to horizon (REQ 2.2.1.4.C), a one meter tolerance ensures accurate tracking and stable video.

REQ 2.2.1.2.E The system shall fly as far as 200 meters (656 ft) from the vehicle without losing communication.

200 meters is the common size of a city block in the United States and elsewhere. This distance would allow the system to investigate the next intersection as the vehicle enters a new block.[3]

> REQ 2.2.1.2.E.P1 The system shall fly as far as 100 meters (328 ft) from the vehicle without losing communication.
>
> 100 meters (328 ft) is considered a good starting place for prototype 1.0.

2.2.1.3 Environment

REQ 2.2.1.3.A The system shall operate fully in the military temperature range of -55 to 125 degrees Celsius (-67 to 257 degrees Fahrenheit).

Since the system is designed for military applications, the typical operating temperature for military grade equipment is expected.

> REQ 2.2.1.3.A.P1 Prototype 1.0 shall operate fully in a temperature range of 0 to 70 degrees Celsius (32 to 158 degrees Fahrenheit).
>
> In an effort to keep budget low, the team will use commercial grade parts. These parts are only qualified for the above temperature range.

REQ 2.2.1.3.B Humidity: the system shall fly at all humidity levels.

The humidity and condensation levels will not ground the device.

> REQ 2.2.1.3.B.P1 Humidity: the system shall fly at any temperature at which the ambient temperature is above the dew point.
>
> The humidity tolerated by the project will be any level at which there is no condensation. This means that the actual relative humidity will vary based on the temperature.

REQ 2.2.1.3.C The system shall fly in dusty environments.

This is particularly important, as military uses will expose the device to dust and small debris, which will limit visibility and be potentially destructive to system components.

> REQ 2.2.1.3.C.P1 Not required for prototype 1.0.
>
> This is similar to the humidity requirement, and though possible, is not required for the prototype. Testing the effects of dust, even on individual components such as motors, potentially adds costs to replace any broken components. The team is unable to afford these additional components for testing given the budget constraints for prototype 1.0.

REQ 2.2.1.3.D The system shall fly in smoky environments.

This is particularly important, as military uses will expose the device to smoke, which will obstruct the view of the vehicle on the ground.

REQ 2.2.1.3.D.P1 There is no smoke requirement for prototype 1.0

In order to view video in a smoky environment, a second infrared (IR) camera would be necessary, which is outside the budgetary constraints of prototype 1.0.

REQ 2.2.1.3.E The system shall fly in winds up to 105 km/hr (65 MPH).

The device will be used outdoors, requiring that it work in many conditions. 105 km/hr is the lowest bound for a tornado.[4] The system is expected to be useful in less than ideal weather conditions, but flying at tornado-like wind speeds is not expected.

REQ 2.2.1.3.E.P1 Prototype 1.0 shall withstand wind speeds up to 27 km/hr (17 MPH).

Prototype 1.0 shall be able to withstand the average wind speed at its production site: Grand Rapids, MI.[5]

REQ 2.2.1.3.F The system shall have the ability to avoid obstacles within the flight path.

This is essential as power lines, trees, and other obstacles will be inherent in the environment for which this system will be used. The system must avoid these obstacles without user intervention.

REQ 2.2.1.3.F.P1 There is no obstacle avoidance required for prototype 1.0

Due to cost and time constraints, obstacle avoidance is not explored in prototype 1.0. However, the team understands that this is essential for this system to meet its full utilization.

2.2.1.4 Video Performance

REQ 2.2.1.4.A The system shall stream video in 1080p quality.

Since the goal of the system is to provide a video stream for protection, this video shall be high definition in order to identify threats.

REQ 2.2.1.4.A.P1 Prototype 1.0 shall stream video at a quality of 240p.

240p quality is a good starting point quality for the video feed, which will show a proof of concept.

REQ 2.2.1.4.B The system shall capture video at 30 frames per second.

At 30 fps, there is a 33.3 ms delay between frames. This is about 1/6 the reaction time of a typical college-age student detecting a visual stimulus.[6] As such, this is not the bottleneck in reacting to threats and thus is sufficiently fast.

REQ 2.2.1.4.C The system shall have a 360 degree field of view in order to see threats from all directions; forward, behind, left, and right, and down.

This is essential, as the quadcopter must be able to see enough of the area in order to see all of the potential threats.

REQ 2.2.1.4.C.P1 There is no minimum field of view requirement for prototype 1.0.

So long as the system is able to send video, proof of concept is achieved. A more expensive camera could be purchased with a wider camera angle to reach the final requirement.

REQ 2.2.1.4.D The system shall have a video feed with a latency of under 810 ms (from the time the video is captured on the quadcopter, the video feed should display on the GUI within 810 ms seconds).

RPGs travel at about 300 m/s. The typical reaction time of a college-aged individual to detect a visual stimulus is 190 ms.[6] In this time, the rocket will have already traveled 57 meters. This does not factor in

a response by the individual or any latency in the video stream. As such, it would be nearly impossible to protect against an RPG fired at close range.

It takes an individual with a weapon about a second to jump around a corner and level the weapon. Since reaction time is 190 ms as mentioned above, that means that the maximum video latency is 810 ms (1000 ms jump – 190 ms reaction).

REQ 2.2.1.4.D.P1 The system shall have a video feed with a latency of under one second.

Since prototype 1.0 is a proof of concept, a video latency under one second will show the usefulness of a video stream.

2.2.2 Physical

These requirements deal with the mechanical size and durability of the solution.

2.2.2.1 Size

REQ 2.2.2.1.A The system shall fit into the back of a vehicle, taking up no more than a total space of 1.3 x 1.3 x 0.4 meters (4.25 x 4.25 x 1.3 feet)

The device must be small enough to fit in existing military vehicles or else the system will not be adopted. This conservative size ensures this is possible.

2.2.2.2 Weight

REQ 2.2.2.2.A The system shall weigh no more than 23 kg (51 lbs).

The device needs to be easily transported and be under the NIOSH lifting recommendation limit of 23 kg.[7]

2.2.2.3 Reparability

REQ 2.2.2.3.A The system shall have a modular design such that all components, with the exception of individual components on a printed circuit board (PCB), are replaced with only the need of a screwdriver, a needle nose pliers, and an Allen wrench.

Due to the rugged application, simple reparability is essential.

2.2.3 Customer

The following are requirements for how the system will interact with the customer.

2.2.3.1 Ease of Use

REQ 2.2.3.1.A The system as a whole shall autonomously track the intended vehicle.

Upon activating the system, it shall not require additional user interaction until the user is ready to deactivate the system. This allows the user to focus on the results of the system (the video stream from above) and not controlling the system. For security customers, this provides a level of justice in that this product should never decrease the situational awareness of the user.

REQ 2.2.3.1.B Training for the device shall be minimal, requiring no more than one eight hour day.

The device shall be intuitive to use and the training shall not require an engineering or technical background.

2.2.3.2 Safety

REQ 2.2.3.2.A A loss of communication shall result in the system landing and having flight temporarily disabled.

In the event that communication is lost, the system will perform an automatic landing that will minimize damage to the system and the chance of injury to others. It will also temporarily disable flight until the user is able to retrieve the system.

2.2.3.3 User-System Interaction

REQ 2.2.3.3.A The base station shall provide a video feed to the user within a GUI.

These statistics will provide valuable information to the user on the flight of the system, even when they cannot physically see the system themselves.

REQ 2.2.3.3.B The base station shall provide flight statistics including heading, altitude (in meters or in feet, selectable by user), and speed (in km/hr or MPH, selectable by user) on the GUI.

These statistics will provide valuable information to the user on the flight of the system even when they cannot physically see the system themselves.

> REQ 2.2.3.3.B.P1 The prototype 1.0 GUI shall display video.
>
> Since prototype 1.0 is simply a proof of concept, video feed will be the only requirement for the GUI. Other flight statistics are deemed as stretch goals for proof of concept purposes.

REQ 2.2.3.3.C The user shall be able to override the autopilot from the GUI.

This will allow the user to fly the system outside of the normal tracking parameters, and allow uses such as arriving at an intersection before the convoy or scanning a general area.

> REQ 2.2.3.3.C.PI Prototype 1.0 shall only need the ability to toggle autopilot on and off.
>
> Turning off autopilot will result in a deterministic action by the system, such as stationary hovering.

REQ 2.2.3.3.D The user shall be able to enable and disable flight when the system is on the ground and the throttle is turned down.

It is necessary for safety purposes to have a disarmed system that is still powered on. This will only be performed once the device is no longer airborne.

REQ 2.2.3.3.E The user shall be able to power on and off the device when it is on the ground and disarmed.

This important control must be available to the user.

2.3 Deliverables

The team provided a variety of deliverables throughout the project including reports, web materials, and prototypes. Table 1 shows a list of the major deliverables with this project.

Table 1: Project Deliverables

Deliverable	Explanation
	First Semester
Presentation 1	Initial presentation of the project providing an initial overview; five minutes with one minute of question and answer (Q&A).
Presentation 2	Second presentation of the project; including a project brief, feasibility, and current status; seven to nine minutes with one minute Q&A.
Team Website	Team website that has team overview, project details, progress, and documentation. Resides on Calvin Engineering webpage; updated regularly.
Team Poster	Team poster that describes the team and gives an overview of the project; updated regularly.
PPFS	Defines the project proposal and the feasibility of the project, includes requirements, budget, management, and status to name a few.
Prototype 0.1	Prototype completed by the end of first semester; includes a flying quadcopter (provided by Professor Yoon Kim), video streaming over Wi-Fi, some tracking and control research, and a basic GUI.
	Second Semester
Presentation 3	The third presentation of the project.
Prototype 0.2	Prototype completed by the end of interim; includes a flying quadcopter with the MutliWii (MW) flight controller.
Presentation 4	The final in-class presentation of the project
Design Notebooks	Design notebooks that are completed individually documenting the design process of each team member.
Prototype 1.0	Final prototype of the project; includes a complete quadcopter system, video streaming into a GUI, and accompanying base station for tracking.
Final Presentation	Final presentation of the project completed on Senior Design Night
Final Design Report	Final design report of the project; includes many portions of the PPFS as well as additional design details and other updates.

3 System Design

This chapter begins to evaluate different solutions to the problem outlined above. The chapter evaluates several design alternatives and concludes with a final solution. The team determined that a quadcopter with a base station was the best approach to solve the problem. High-level system details are established in this section. Component-level requirements for the quadcopter and base station are outlined in the following two chapters.

3.1 Decision Matrix

Before continuing on any further analysis of the problem, the team analyzed different solution platforms. The team evaluated five potential solutions that are detailed in the following sub-sections. This section concludes with a decision matrix that summarizes our decision along with justifications for the corresponding scores.

3.1.1 Possible Solutions

The five possible solutions are detailed below, followed by the different decision criteria used to evaluate the solution.

3.1.1.1 Quadcopter

Quadcopters are increasing in popularity for applications ranging from search and rescue to aerial photography. Quadcopters use four separate arms, each with a rotating propeller in order to achieve stable flight. Recent technological developments in microprocessors have made possible the complex, dynamic control systems necessary to balance a quadcopter. Given an easily expandable platform and a high maneuverability in the air, quadcopters have proven effective for many common UAV applications as is highlighted in an article in Popular Mechanics Magazine.[8] An example picture of a quadcopter outfitted with cameras is shown in Figure 3.[9]

Figure 3: Quadcopter Outfitted With Cameras[9]

3.1.1.2 RC Helicopter

RC helicopters (Figure 4) (either gas or electric powered) have been around for hobbyists for decades. Sizes can range from hand size to the size of a car. RC helicopters give the ability to hover in the air with relative stability (discussed later).

Figure 4: RC Helicopter[10]

3.1.1.3 RC Airplane

Like RC helicopters, RC airplanes (Figure 5) are not new to the hobby world. A downside to RC airplanes is the fact that they cannot hover in one place.

Figure 5: RC Airplane[11]

3.1.1.4 Balloon / Blimp

Another solution to the problem would be to use a balloon or blimp to stream the video, an example is seen in Figure 6. The device would be either towed by a rope or controlled by RC. It is likely that this type of solution would have a small motor for maneuverability purposes. A factor to consider is the size (1 m^{12} to 4.3 m^{13}) of this type of device, making it a very easy target for enemies.

Figure 6: RC Blimp[12]

3.1.1.5 Panoramic Ball Camera

A new consumer product exists where a user throws a ball in the air and at the peak altitude, it takes several pictures in each direction.[14] The pictures are then loaded onto a computer for viewing and stitched together to form a seamless overhead image of the surroundings. The panoramic ball camera is shown in Figure 7 below. However, this solution was quickly abandoned as it became apparent this was not practical. Some of the reasons it was deemed impractical include the fact that constant video would not be provided, users would be exposed when the ball needed to be thrown, and throwing and retrieving a ball from a moving vehicle is not feasible.

Figure 7: Panoramic Ball Camera[14]

3.1.2 Decision Criteria and Justification

Each of these solutions were compared against eight criteria that the team deemed important to the problem. Table 2 displays the results of this decision matrix below. Criteria weights were chosen between one and ten, with ten being the highest weighted. Weight values can be repeated. Scores, on the other hand, are ranked between one and four with four being the best score. Scores are not repeated on any category.

Table 2: Decision Matrix

Categories:	*Weight 1-10*	**Quadcopter**	**RC Helicopter**	**RC Airplane**	**Balloon / Blimp**
Battery / Fuel / Time in Air	*5*	2	1	3	4
Durability / Replacement Parts	*6*	4	3	2	1
Wind Resistance	*7*	4	3	2	1
Cost	*4*	1	3	2	4
Ease of Use	*4*	3	2	1	4
Maneuverability	*9*	4	3	2	1
Expandable Platform	*8*	4	3	1	2
Safety in Hostile Environment	*10*	4	3	2	1
	Total:	**186**	**145**	**99**	**100**

3.1.2.1 Battery / Fuel / Time in Air

How well the system stays in the air and for how long. Time in the air was a middle ground at a five due to the customizability of each system. The team did not feel this was a major differentiating characteristic because battery life and fuel capacity can be easily expanded in most systems.

3.1.2.2 Durability / Replacement Parts

How much abuse each system can take and how easy it is to replace broken parts. Durability and replacement parts were moderately important at a six because the solution needs to be able to hold up over time. Replacement parts deal with how easy it is to change parts (i.e. unplug and replace, tape, etc.). In addition, this accounts for the variety of parts that could be replaced. For example, a system that only has 30 parts, but only three different variety of parts, will score better than a system with 15 parts that are all unique.

3.1.2.3 Wind Resistance

How well the system is able to counteract the effects of light to moderate winds, including wind gusts. Wind resistance was rated moderately high at a seven due to the conditions in which the system will be used. Often times there will be opposing or cross winds that must not affect the stability of the system.

3.1.2.4 Cost

Expense to design, build, implement, and maintain. Cost was weighted at a four because when it comes to safety, compromising is not an option. The best parts must be used in order to mitigate failure whenever possible.

3.1.2.5 Ease of Use

Ease of use was weighted at only a four because it was assumed that the user will have at least some training in implementation, and users will likely be knowledgeable in technical areas. In addition, nearly all of the solutions will get easier as the amount of design and development increases.

3.1.2.6 Maneuverability

This category includes acceleration, flexibility, hovering, agility, and obstacle avoidance. This category was weighted at nine out of ten because of the nature of the vehicle it will be following. In many situations, the convoy to follow will start and stop quickly as well as turn corners and change directions suddenly. To ensure the vehicle remains in the video frame the solution must be highly maneuverable.

3.1.2.7 Expendable Platform

How easy it is to add cameras, sensors, and tracking hardware to the system. This was weighted an eight out of ten because of the project's need to carry multiple components such as communication devices and cameras.

3.1.2.8 Safety

How dangerous is it for the user to interact with the system in a hostile environment. Interactions the user could be expected to perform are as follows: launch the device, retrieve the device, and view the video. This was weighted a ten out of ten because the primary goal of the project is to implement a solution that leads to increased safety.

3.1.3 Justification

The following sections describe how each solution was scored for each decision criterion.

3.1.3.1 Battery / Time in Air

- **Quadcopter** - Average flight times for quadcopters with a 2200 mAh battery are about ten minutes.[15]
- **RC Helicopter** - Average flight times for a RC helicopter with a 2200 mAh battery are around seven minutes.[16]
- **RC Airplane -** Average flight times for a RC airplane with a 2200 mAh battery are around 20 minutes.[17]

- **Balloon / Blimp -** Balloons have a very long flight time when compared to the other solutions as it is only limited by the time it takes to drain the battery from accessories being powered.
- **Conclusion -** Based on the following results, the RC helicopter came in last with the lowest score, followed next by the quadcopter, and then the RC airplane. The best score was awarded to the balloon/blimp for its ability to stay potentially airborne for days.

3.1.3.2 Durability / Replacement Parts

- **Quadcopter -** Since a quadcopter uses several of the same components (i.e. four motors, four arms, four electronic stability controllers (ESCs) etc.) this makes it easy to swap out broken components. Similar internal electrical components would be used for a quadcopter, RC helicopter, and a RC airplane. Therefore, these components did not affect durability ratings. Quadcopters received the best score.
- **RC Helicopter -** Similar to a quadcopter, except all components are unique. Therefore, "replacement parts" receives a slightly lower score.
- **RC Airplane -** As discussed in the maneuverability section, RC airplanes have a low maneuverability. For this reason, it is more likely that a significant crash could happen. Therefore, it scored lower than the quadcopter and RC helicopter.
- **Balloon / Blimp -** Balloons received the lowest score since a small tear in the balloon would render the device useless. Repairs to a torn balloon would not be simple. It would require patching the hole and refilling the air. In order to refill with air, a user would be required to carry suitable gasses in a compressed state.
- **Conclusion -** Looking at all of these options, the quadcopter took the highest score, followed next by the RC helicopter. The RC airplane scored second to last with its higher probability for crashes, followed by the balloon/blimp because of its large and vulnerable balloon.

3.1.3.3 Wind Resistance

- **Quadcopter -** Due to quadcopters' robust control systems, they are able to adjust to changes in their environment by rapidly adjusting rotor speed without needing to turn the whole system. Quadcopter control systems are different from other air vehicles because they do not have a forward and backward direction, allowing them to move in all directions at any point.
- **RC Helicopter -** A RC helicopter would not tolerate gusts as well as a quadcopter due to lower maneuverability (See Maneuverability section below).
- **RC Airplane -** RC airplanes, though quite resistant to steady winds, are not able to quickly adapt to changing winds as quadcopters can, due to the lack of dynamic balancing available.
- **Balloon / Blimp -** A balloon has little motor control and is completely vulnerable to the slightest change in wind. Therefore, the blimp received the lowest score in this category.
- **Conclusion -** The quadcopter came in first due to its small and dense nature. Closely behind were the RC helicopter and the RC airplane. Lastly, again because of its size and limited maneuverability, the blimp/balloon came in last.

3.1.3.4 Cost

- **Quadcopter -** A stable quadcopter kit is about $600.[18]
- **RC Helicopter -** A RC helicopter with a good flight controller is around $400.[19]

- **RC Airplane -** A RC airplane option that includes the features needed for this project including GPS and flight planning are roughly $600.[20]
- **Balloon / Blimp -** The cost for a blimp that would cover the specifications is about close to $150.[21]
- **Conclusion -** The costs are lowest for the blimp due to its non-complex nature. Next is the RC helicopter. Following that is the RC airplane, and lastly the quadcopter.

3.1.3.5 Ease of Use

- **Quadcopter -** Quadcopters include self-balancing software that keeps them flying without user input. With easy to implement antennae technology, they can be controlled from inside an armored vehicle and even programmed to fly specific routes without user input.
- **RC Helicopter -** RC helicopters are similar to quadcopters in that they can be self-balancing as well as include pre-programmed routes.
- **RC Airplane -** While they include pre-determined route flying, it is hard to quickly change directions in RC airplanes due to their need to maintain airspeed in order to stay aloft. Another downside is that these require some sort of takeoff distance or a hand thrown start that exposes the user.
- **Balloon / Blimp -** Once the balloon is attached or takes off, it would require little effort to land or replace batteries. Therefore, the balloon received the best score
- **Conclusion -** Looking at all of the options, it was easy to see that the blimp is the easiest of the systems to use, followed by the quadcopter. Next were the RC helicopter and the RC airplane, which requires the most attention of the powered solutions.

3.1.3.6 Maneuverability

- **Quadcopter -** Since quadcopters by design do not have a front facing direction, they are capable of moving in any direction from any orientation. This allows them to change direction and speed very quickly. A quadcopter's ability to pitch and roll is modeled by the difference in thrusts of two propellers multiplied by the length of an arm, as shown in Figure 8 and the two equations below. These two factors give the quadcopter great maneuverability.[22]

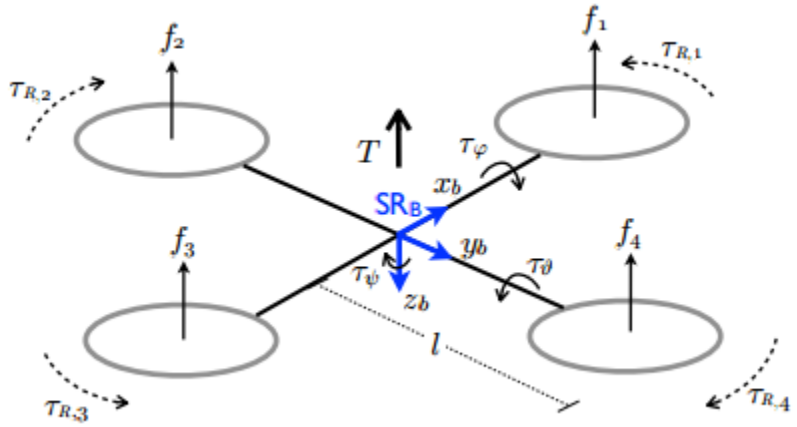


Figure 8: Control of a Quadcopter [22]

$$\tau_{\varphi} = l(f_2 - f_4) \; and \; \tau_{\vartheta} = l(f_1 - f_3)$$

- **RC Helicopter -** While RC helicopters do have a front and back direction, they are still capable of moving in any three dimensions at a given time. Since helicopters only have two propellers (one for thrust and one for stability). RC helicopter's directional flight is determined by the tilting of a swash plate, which in turn tilts the primary rotor. Therefore, the directional force available is modeled by $F * \cos(x)$ where F is the force of the primary propeller and x is the angle of the swash plate from the perpendicular.[23]
- **RC Airplane -** RC airplanes are not capable of stopping, hovering, or taking off with no speed. They are also unable to change directions quickly side to side. While the RC airplane could go into a holding pattern, this would likely be unhelpful since constant video from the same perspective would not be provided. A similar video perspective is desired to provide the user with a simplistic design. If the video is constantly changing perspective, the user will need to reorient him/herself by finding north, south, east or west every time he/she views the video. In addition, RC airplanes were not deemed maneuverable enough to fly at low altitudes in urban areas. Such flight patterns would almost certainly lead to a collision between the device and the surroundings. Since RC airplanes only have one propeller, they can only travel forward. RC airplanes cannot move side to side without moving forward and cannot move backwards at all.
- **Balloon / Blimp -** Balloons would likely be tethered to the vehicle, but in addition, they would likely have small motors to allow some movement. This does not allow for great maneuverability. Small movements could be performed, but they would be by no means quick.
- **Conclusion -** The quadcopter excelled in this category due to the fact that it can move in any direction in 3-D space immediately, without turning first or building up speed. Next was the RC Helicopter, with its only limitation being that it must turn before executing some maneuvers. The RC airplane came in third because it requires speed to move, and any increase in altitude cannot be done instantly but instead requires routing. Following these, the balloon scored worst of the solutions because it is slow and its large size makes it cumbersome.

3.1.3.7 Expandable Platform

- **Quadcopter -** Quadcopters are designed with expansion in mind. The flat and symmetrical frame makes it easy to incorporate more hardware onto the design. They also have self-balancing abilities that allow for minor frame imbalance, which enables hardware to be added without the concern for perfect balance. In addition, quadcopter arms are estimated to be 0.3 m long. With half of the space estimated as usable for expansion (to prevent interference with propellers), the platform size is 0.045 m^2. See Figure 9 for a visualization.

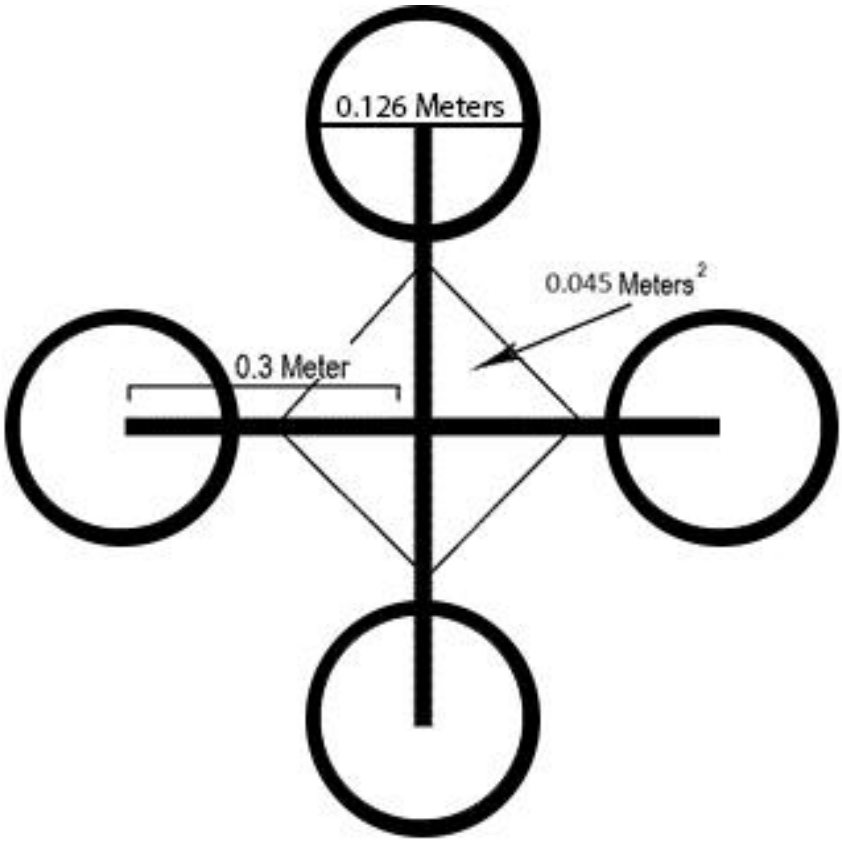


Figure 9: Platform Area of a Quadcopter

- **RC Helicopter -** RC helicopters have room for cameras and GPS modules, but not much else. Due to the hull, the only room to add additional components would be on the skids along the bottom of the RC helicopter. The skid area is estimated to be 15.25 cm by 10.16 cm giving an area of 0.015 m^2, just 33% of the estimated area available for a quadcopter. In addition to this, the volume available for an RC helicopter is limited by the landing gear on the bottom and the rotor on the top, while quadcopters can outfitted with much taller components without conflicting with the rotor and landing gear.
- **RC Airplane -** RC airplanes are built inside a certain predefined frame. This makes it difficult to add additional hardware, as there often is a space issue. Unlike RC helicopters, RC airplanes do not have skid like platforms to mount extra hardware on. Balance is also more of a concern, as RC airplanes do not have any way of dynamically counteracting balance. This would be problematic because symmetrical weight needs to be added to balance the RC airplane, potentially leading to weight being added solely for balance.
- **Balloon / Blimp -** The balloon/blimp scored quite low in this category, not because there is not space to add accessories, but because the lift is heavily affected by small weight increases. The whole system may need to be enlarged in order to handle the weight of additional accessories.
- **Conclusion -** The quadcopter scored best in this category because it is made with expansion in mind. The RC helicopter was second because it is similar in nature to the quadcopter, except that the hull limits space. A RC airplane is quite space limiting because of its specifically designed fuselage. Last are blimps, which are heavily affected by weight.

3.1.3.8 Safety

- **Quadcopter -** Quadcopters can be remotely controlled by a RC transmitter inside a vehicle. They can be programmed to take off and land from the top of a vehicle or a platform on the back to ensure that no operators need to exit the vehicle.
- **RC Helicopter -** RC helicopters have similar safety characteristics to quadcopters.
- **RC Airplane -** RC airplanes either require a takeoff runway, or close user interaction (for a hand thrown start). These options extend the time required for takeoff and put the user in more danger by requiring them to be outside the vehicle. Any time that the operator is outside of the vehicle presents a safety risk.
- **Balloon / Blimp -** RC balloons or blimps would not require an operator to exit the vehicle and expose himself or herself. However, being an air-holding vessel, these options would be vulnerable to punctures from surrounding objects, such as trees, buildings, and bridges.
- **Conclusion -** Quadcopters and RC helicopters are the best in this situation due to their vertical launches and programmable flights. However, quadcopters edged out because of increase takeoff and landing maneuverability. The balloon/blimp scored next again because of its near-vertical takeoffs. RC airplanes, because of their long takeoff and landing requirements, scored the lowest.

3.1.4 Best Solution

As one can see from the Table 2 above, the quadcopter scored the highest, and the team believes that it solves the problem most efficiently due to its maneuverability, expandable platform, and safety.

3.1.4.1 Recognized Weaknesses

While the solution chosen for the problem provides many positive attributes, it is not without its weaknesses. As is a problem with all RC vehicles, quadcopters have to constantly tradeoff between abilities and battery life. As soon as more features are added, the weight gain and processing power needed take a sizeable amount of battery life from the overall flight time. In addition to this problem, quadcopters are particularly vulnerable to crash damage. This could have been a major problem, especially during initial testing phases that will require iterative safety designs.

3.2 Overall System Block Diagram

The overall system block diagrams are in the following two sections. The first section looks at the hardware aspects of the solution and how the two subsystems interact, while the second section focuses on the software aspects of the solution and how the software on the two separate subsystems communicate.

3.2.1 Hardware

The overall system hardware block diagram is seen in Figure 10 and shows the two subsystems of the solution.

One part is the base station, which is located inside the vehicle. The base station includes two major subsystems, the controls and communication system, and the base station computer. The controls and communication system provides the network over which video streaming can occur, as well as the ability to send controls from the base station computer to the quadcopter. The base station computer provides an interface for the user to interact with the system. Refer to Chapter 5 for the details of the base station.

The second part of this solution is the quadcopter. This consists of a flight controller, a video graphics processing unit (VGPU), a camera, GPS module, motors, and a power distribution system. The GPS module receives data from the outside world while the motors provide thrust. The flight controller and VGPU send/receive data and controls to/from the base station. All of the interface connections are described in Table 3 below. Refer to Chapter 4 for details of the quadcopter.

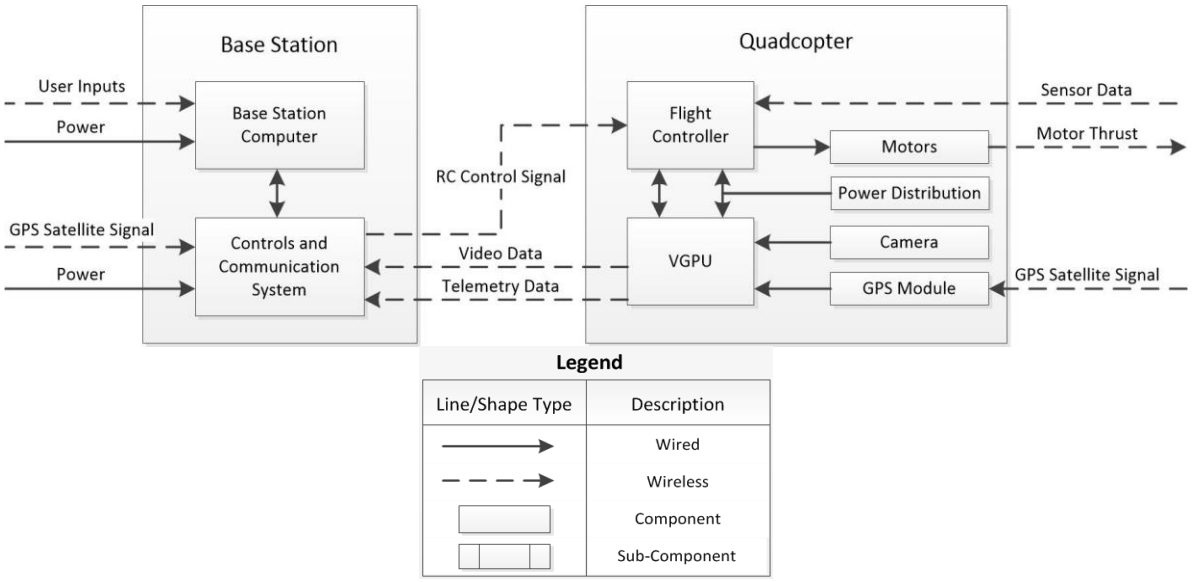


Figure 10: Overall System Hardware Block Diagram

Table 3: System Hardware Block Diagram Interface Descriptions

Signal Name	Description	Protocol
User Inputs	The user inputs include the ability to turn the quadcopter on/off, toggle on/off the autopilot, toggle on/off the video stream, and arm/disarm the system per REQ 2.2.3.3.C, REQ 2.2.3.3.D, REQ 2.2.3.3.E. All inputs are done through the GUI on the base station computer, using switches on the controller, or by physically plugging in the battery on the quadcopter.	GUI Buttons, Controller Switches, Battery Plug
Power	The power for the base station is provided through the battery of the car where the base station is mounted. The car powers both the base station computer and the controls and communication system.	15 V Car Outlet
GPS Satellite Signal	The controls and communication system and the GPS module on the quadcopter receive a GPS signal from global orbiting satellites.	NMEA 0183[24]
RC Control Signal	A control signal to control the quadcopter is sent from the controls and communication system to the flight controller on the quadcopter. This signal is a frequency-hopping spread spectrum signal sent over a 2.4 GHz frequency.[25]	2.4 GHz

Telemetry Data	The telemetry data is sent over an 802.11n Wi-Fi connection using transmission control protocol over internet protocol (TCP/IP). This carries the GPS coordinates to the base station computer as well as other telemetry data such as the quadcopter altitude, heading, and RC inputs. The GPS position is refreshed at 10 Hz and data is sent each time new GPS data is received. See section 4.2.9.2.2 for details and necessary bandwidth calculations.	802.11n/ TCP/IP
Video Data	The video data is sent in an H.264 compressed format over an 802.11n Wi-Fi connection. The latency is determined by REQ 2.2.1.4.D and the data is sent over UDP/IP using GStreamer. Video data is sent on a different port than the telemetry data to prevent interference. See section 4.2.9.2.1 for bandwidth calculations.	H.264/ GStreamer
Motor Thrust	The motors provide thrust to the outside world in order to lift the quadcopter.	N/A
Sensor Data	The flight controller has various sensors that read data of the environment and send it to the flight controller for analysis and flight.	Environment Data*

**The environment data received by the flight controller sensors varies for each sensor. The barometer measures the atmospheric pressure, the magnetometer measures the earth's magnetic field, the accelerometer measures the acceleration of the quadcopter, and the gyroscope measures the orientation of the quadcopter.*

3.2.2 Software

The overall software block diagram is seen in Figure 11. Once again, there are two main subsystems, software running on the quadcopter, and software running on the base station. On the quadcopter, the VGPU is a Raspberry Pi that is running Raspbian operating system (OS). On the OS are the two main functional software groups, video streaming and telemetry streaming. The telemetry streaming gets its data from the GPS module and the MW flight controller (on which the flight control software resides). The video streaming gets data from the camera that is connected to the Raspberry Pi. Both video and telemetry streaming interface with the base station computer over a Wi-Fi network. For video streaming, the choice of using GStreamer over UDP/IP was made because the latency was much lower than the RTSP stream over TCP/IP and the FFmpeg stream over UDP/IP. See section 4.2.9.2.1 for a full description of the video software decision. For telemetry streaming, due to the critical nature of all the information arriving intact, TCP/IP was chosen as opposed to UDP/IP. This will guarantee that the data will get from the quadcopter to the base station when a path is available. This is critical, as the telemetry data is used in the autonomous tracking algorithms. See section 4.2.9.2.2 for a full description of telemetry streaming. On the base station computer is a GUI running on Windows 7 OS. This GUI receives both the video stream and telemetry stream. Once the GUI receives telemetry information, it is parsed and displayed to the screen for the user as well as sent to the RC interface module. From there, the information is parsed, autopilot commands are calculated, and RC commands are sent to the RC transmitter. All of the interface connections are described in Table 4 below.

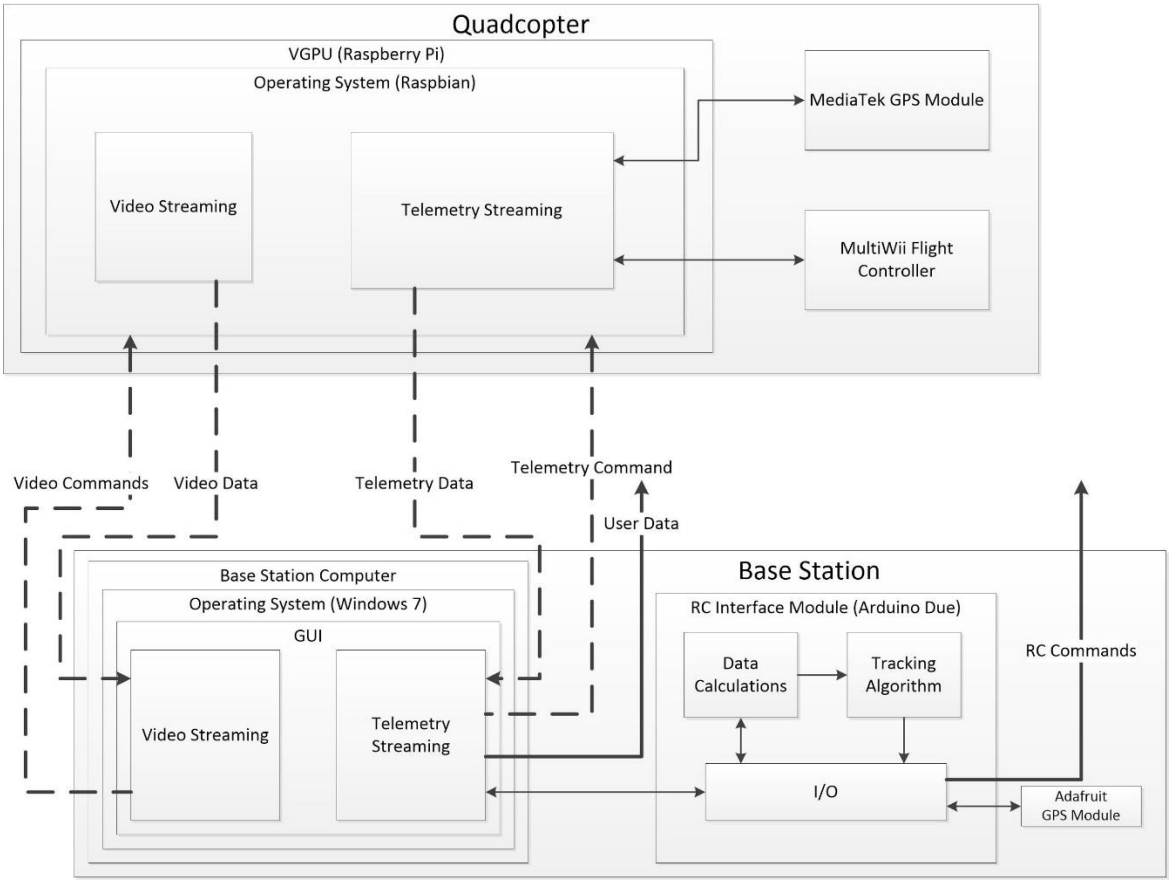


Figure 11: Overall Software System Block Diagram

Table 4: System Software Block Diagram Interface Descriptions

Signal	Description	Type
Video Commands	These video commands are secure shell (SSH) commands ("./gstreamer.sh") to initialize a script to setup the camera or commands ("kill $(pidof raspivid)") to terminate the camera process.	SSH
Video Data	The video data is a GStreamer stream sent from the VGPU to the base station computer via UDP/IP as described in section 4.2.9.2.1	GStreamer
Telemetry Command	A telemetry command is sent from the GUI via a button click. This command initializes a python script on the Raspberry Pi. This script gathers telemetry information from the flight controller.	SSH
Telemetry Data	The telemetry data string can contain several instances of quadcopter telemetry information. The quadcopter sends information via TCP/IP and the GUI collects all information available on the TCP/IP read buffer and stores it into the telemetry data string. This string can be several lines long.	String See Below (Table 5)

User Data	User data is the collection of flight statistics printed to the GUI. This data serves no functional purpose, but only provides the user with up to date information.	String See Below (Table 5)
RC Commands	RC commands are voltage levels sent to the RC transmitter to be sent on to the quadcopter as defined in section 5.2.2.	Analog Voltage

Table 5: System Protocol Definitions

Protocol	Description
Telemetry Data	The telemetry data is sent as a string. This string starts with 'A' to identify the beginning of a new data packet, contains a 'W' to signify the end of the standard data, and a 'Z' to signify the end of the packet. The standard data is the data sent to the quadcopter and contains the fields: latitude, longitude, altitude, heading, and timestamp. Between the 'W' and the 'Z' is various debugging data. It contains the following data values; number of satellites, RC pitch, RC roll, RC yaw, and RC throttle.
User Data	The user data contains information about flight statistics of which the user may be interested. This information consists of latitude and longitude difference between the quadcopter and base station, quadcopter altitude, quadcopter heading, amount of time that has passed since a message was sent by the quadcopter and received by the base station, number of quadcopter satellite GPS fixes, and quadcopter RC pitch, RC roll, RC yaw, and RC throttle values.

3.3 Feature Matrix

During the start of the first semester, the team identified different levels of features and their implementation importance in the system. The feature matrix is below in Table 6. The following section goes into more detail for each of these features.

Table 6: Feature Matrix

Legend
Essential
Core Features
Basic Features
Stretch Features
Dream Features

Eagleye					
Mechanical	**Control**	**Communication**	**Power System**	**Sensors**	**Video**
Strong Frame	Motor Control	RC for control	Battery	Hovering Sensors	Take pictures
Propeller Gaurds	Full Motion Control	Wi-Fi for Video/GPS	Easily Accessible	GPS	Stream SD Video
Robust Landing Gear	Autonomous Following via GPS	Redundant Communication	Base Battery Charging		Steam Video to GUI
Crash Protection	Full control via GUI		Battery Monitoring via GUI	Altimeter	Stream HD Video
Reach a speed of 50 km/hr (31 MPH)	Altitude Control		Extra-Range Battery	Obstacle Avoidence	
	Automatic Takeoff and Landing	Emergency Comm-Loss Landing	Emergency Low-Battery Landing	Speedometer	Record Video while Streaming
Rainproof	Autonomous Following via IR	Total Wi-Fi System	Laser Charging	Computer Vision for IR Beacon	Horizon to Horizon Field of View

** Graph Format modeled after SWIM-R (2012-2013)*

3.3.1 Feature Descriptions

This section attempts to provide some clarity and definition for the features shown in the feature matrix above.

3.3.1.1 Mechanical

Mechanical features are items that deal with the mechanical structure of the quadcopter. These features are defined below.

- **Strong Frame** - The strong frame will prevent the arms from flexing and give the quadcopter more stability. This is essential because a flimsy frame will make it difficult to control the quadcopter as well as give the user a sense of distrust. The strong frame will also provide a greater level of durability for crashes.
- **Propeller Guards -** Propeller guards will prevent the propellers from being damaged by near-by objects. This is not essential for flight, but it is a core feature to prevent damage during testing and to provide better safety to the users during takeoff and landing.
- **Robust Landing Gear -** Robust landing gear will help prevent takeoff and landing damage from occurring. This is a basic feature since normal landing gear will perform adequately, but robust landing gear will aid in testing the automatic takeoff and landing and increase system durability.
- **Crash Protection** - Adding additional protection to the quadcopter in the event of a crash would be very useful to protect the electrical and mechanical components of the quadcopter. However, it is not essential to creating the system.
- **Reach an Air Speed of 50 km/hr -** Quadcopters are known for their agility, but not their speed. Having a quadcopter reach 50 km/hr (30 MPH) will help it keep pace with the convoy in most situations.[26] For the purposes of this project, the team will strive for 15 km/hr (9.3 MPH) and document how achieving 50km/hr would be done given more sufficient resources.
- **Rainproof -** A rainproof quadcopter will be one more step in creating a robust system that can operate in all environments. This can be done by covering the electronics in a case or using commercial waterproofing techniques like NeverWet.[27]

3.3.1.2 Control

Features that affect the movement, navigation, or overall control of the quadcopter are described here.

- **Motor Control** - Basic control of the motors on the quadcopter is essential to balancing and lifting the quadcopter.
- **Full Motion Control** - Full motion control is the ability to move the quadcopter any 3-D direction, which will result in a more agile system.
- **Autonomous Following via GPS -** Once full motion control has been established, autonomous GPS navigation is the next step in the control process. This is a core feature to the project in order to make the project track autonomously.
- **Full control via GUI -** It would be desirable to combine many of the quadcopter controls in the GUI. Control of the quadcopter is a basic feature that would be added to the GUI.
- **Altitude Control -** Altitude control would give the team the ability to tell the quadcopter to fly at a set height. This is a basic step to move towards automation.

- **Automatic Takeoff and Landing -** Automatic takeoff and landing would be a big step towards full automation. Without this feature, a user could manually control take off and switch to automated flight once it is in the air.
- **Autonomous Following via IR -** This redundant form of tracking will help the quadcopter stay in contact with the base station in the event that the GPS signal is weak or lost.

3.3.1.3 Communication

This section describes features that could be used to provide means of communication between the quadcopter and the base station.

- **RC for Control -** Since RC is the current protocol used for most quadcopters, it is essential to have in order to achieve basic flight. In addition to basic manual control, RC could be used for automated controls.
- **Wi-Fi for Video/GPS -** Wi-Fi will be the primary way for the onboard computer to send information to the ground. A wireless network will be set up to send GPS and video signals for processing on the control computer.
- **Redundant Communication** - Redundant communication protects against the event RC communication is lost. This is the next major development step after the full system is working with basic communication.
- **Emergency Communication-Loss Landing** - In the case that all communication fails with the vehicle, a landing command will be given after a timeout expires to prevent a crash.
- **Total Wi-Fi System -** In this system, Wi-Fi would be used in all modes of communication (video, GPS, and controls). This could also be used as another redundant form of control.

3.3.1.4 Power System

This section describes the components, which deal with powering the quadcopter and base station.

- **Battery** - Using a battery to power the quadcopter is essential for basic flight.
- **Easily Accessible -** A battery will need to be replaced and recharged often. Therefore, an accessible battery is essential.
- **Base Battery Charging -** Integrating the battery charger into the base station will help simplify the number of parts in the system and increase convenience. It will also make the system more portable if a battery is able to charge while in a moving vehicle.
- **Battery Monitoring via GUI -** Battery monitoring is a basic feature to prevent damaging the battery and making it available on the GUI makes the system easier to use.
- **Extra-Range Battery -** Because of the high performance nature of the vehicle and all the processes running on board, power consumption will be relatively high, yielding a need for a high capacity battery.
- **Emergency Low-Battery Landing -** A command will be needed to tell the quadcopter to land if the battery is under a certain percentage. With sensitive equipment on board, a controlled landing is better than an uncontrolled landing.
- **Laser Charging -** Perhaps a science fiction feature, but this could offer nearly unlimited flight time. Because of the difficulty, it is a dream feature.

3.3.1.5 Sensors

The sensor section describes various sensors that could be used to collect system information

- **Hovering Sensors -** Sensors on the quadcopter are essential to keep it hovering autonomously. These include accelerometers, gyroscopes, a compass, and possibly an optical flow camera.[28]
- **GPS -** GPS is used to monitor the location of the quadcopter and the computer on the ground. The GPS signals from both are compared and used to calculate the necessary movement of the quadcopter.
- **Altimeter -** An altimeter will be necessary to implement altitude control. Since an altimeter of some sort is necessary for altitude control, this will need to be a core feature as well. This can be done by using a barometer, sonar sensors, or GPS.[29]
- **Obstacle Avoidance -** Obstacle avoidance is another major design step that would be needed in order to avoid trees, bridges, light posts, and other things that could impede the flight of the quadcopter. This could be done by using infrared proximity sensors, or an optical flow camera.
- **Speedometer -** A graphical speedometer would make it possible to monitor the speed of the quadcopter on the GUI. This is not necessary to meet the requirements, but would be useful to users. The speed would be calculated in software using GPS data.
- **Computer Vision for IR Beacon -** One way of determining movement is to have an IR beacon on the car and use computer vision to register the beacon and follow it. Similar projects have used IR sensors from popular gaming consoles to do tracking of a vehicle.[30]

3.3.1.6 Video

This section defines items relating to the video, which is sent from the quadcopter to the base station.

- **Ability to Take Pictures –** At a minimum, the system should provide still pictures of the area from the view of the quadcopter.
- **Stream SD (Standard Definition) Video -** Core to the system is streaming video provided from the quadcopter, which allows real-time view of the area surrounding the convoy.
- **Stream Video to GUI -** The streaming video should be integrated into the GUI in order to simplify and minimize the number of components in the system.
- **Stream HD (High Definition) Video -** It will be important to have high quality video from the quadcopter in order to pick out threats from above. However, it is not essential to the quadcopter system and is therefore not an essential feature.
- **Record Video while Streaming -** Being able to record the video while also streaming would give the user playback at another time. This is not crucial for the system and is therefore a stretch feature.
- **Horizon-to-Horizon Field of View -** Several cameras could be used to see all around the vehicle on the ground. Another option would be to create a simple controllable gimbal to move the camera's view. This component will be beyond the scope of this project but would be needed in a production model.

3.4 Feature Results

Table 7 shows the results of what the team has completed or partially completed. A description of what was accomplished for each feature is discussed below.

Table 7: Feature Matrix Implementation

Legend
Completed Feature
Partially Complete
Incomplete

Eagleye					
Mechanical	**Control**	**Communication**	**Power System**	**Sensors**	**Video**
Strong Frame	Motor Control	RC for control	Battery	Hovering Sensors	Take pictures
Propeller Gaurds	Full Motion Control	Wi-Fi for Video/GPS	Easily Accessible	GPS	Stream SD Video
Robust Landing Gear	Autonomous Following via GPS	Redundant Communication	Base Battery Charging		Steam Video to GUI
Crash Protection	Full control via GUI		Battery Monitoring via GUI	Altimeter	Stream HD Video
Reach a speed of 50 km/hr (31 MPH)	Altitude Control		Extra-Range Battery	Obstacle Avoidence	
	Automatic Takeoff and Landing	Emergency Comm-Loss Landing	Emergency Low-Battery Landing	Speedometer	Record Video while Streaming
Rainproof	Autonomous Following via IR	Total Wi-Fi System	Laser Charging	Computer Vision for IR Beacon	Horizon to Horizon Field of View

** Graph Format modeled after SWIM-R (2012-2013)*

3.4.1 Feature Descriptions

This section discusses what features were completed, as well as what work was done on the partially completed features.

3.4.1.1 Mechanical

All essential and core features were implemented in this section. Crash protection, which was a basic feature, was partially implemented and its progress is described below

- **Crash Protection -** In addition to propeller guards, the team added shock absorbing foam pads to the quadcopter legs to mitigate the damage from rough landings. In addition, the team added a protective case to the flight controller to help protect it in the event the quadcopter flips over. The team would like to add additional protection to the quadcopter for the event of a flip in order to fully implement this feature.

3.4.1.2 Control

Once again, the team achieved all essential and core features. One of the basic features was implemented, but full control via GUI was not because of time constraints. The stretch goal of automatic takeoff and landing was begun, however, and it is discussed below.

- **Automatic Takeoff and Landing -** For safety purposes, the team implemented a feedback free automatic landing sequence. This autonomous landing function slowly lowers thrust until motors are turned off. The team used a feedback free method so this function could be called when communication is lost. In addition to the open loop landing function, the team would like to add a take-off and another landing function with altitude feedback enabled to fully implement this feature.

3.4.1.3 Communication

Both of the essential features were completed for communication, but one of the core features was only partially finished. Redundant communication was begun, but not finished due to time constraints and necessary design work for other components. However, the team did finish the stretch

feature of emergency communication-loss landing. The total Wi-Fi system was also under development, and is discussed below.

- **Redundant Communication -** Wi-Fi is the primary medium through which data is sent; in addition, the team began initial research on XBee as a redundant form of communication between the base station computer and the quadcopter. The XBee modules available at Calvin were not sufficient for low latency telemetry streaming because they only sent data at a rate of 1-2 Hz. Therefore, research and development was paused to focus on more critical design areas. The team would like to take the research completed and implement XBee to finish this feature.
- **Total Wi-Fi System -** Video and GPS information is sent from the quadcopter to the base station via Wi-Fi; however, the team would like to send quadcopter controls over Wi-Fi as well to fully implement this feature and give the system another layer of redundant communication.

3.4.1.4 Power System

The team finished both essential and the one core feature for the power system. They also began work on an extra-range battery, which was one of the basic features.

- **Extra-Range Battery -** The team secured several batteries of different sizes that provided different flight times. Outside of using larger batteries, the team did not experiment with multiple batteries in parallel to achieve longer flight times.

3.4.1.5 Sensors

All of the essential, core, and basic features for sensors were completed. The team also researched the possibility of a speedometer, and it is partially implemented as discussed below.

- **Speedometer -** The team began work on implementing velocity measures of the quadcopter using GPS location information. This is not a very precise measure of speed, but does give a rough estimate for motion. A full implementation would include air and ground speed sensors as well as a display in the GUI.

3.4.1.6 Video

The team completed the essential, core, and basic features for video. Research was completed on sourcing a camera with a horizon-to-horizon field of view, but it was not finished due to cost constraints.

- **Horizon-to-Horizon Field of View -** While there are cameras that provide a horizon-to-horizon field of view, they are expensive. For this reason, they were not pursued any further for prototype 1.0. This is a feature, however, that would be necessary for the production model.

4 Quadcopter Design

This chapter discusses the quadcopter design in detail by breaking the design down by components. Requirements, alternatives, decision criteria, decision, implementation details, and finally unit tests are given for each quadcopter component. The next chapter will discuss the design details of the base station. A picture of prototype 1.0 is seen in Figure 12 below.

Figure 12: Quadcopter Prototype 1.0

4.1 Quadcopter Architecture

A block diagram of the quadcopter architecture is shown in Figure 13 below. Table 8 accompanies the block diagram and lists all of the communication interfaces between the various subsystems. The flight controller receives power through the power distribution board and interfaces with the flight controller sensors as well as the four ESCs. The four ESCs drive each of the four motors. The other main component of the quadcopter subsystem is the VGPU, which interfaces with the camera, the Wi-Fi dongle, and the flight controller. A GPS module receives data from global orbiting satellites, interacts with the VGPU through its input/output (I/O) pins, and then the telemetry information is sent over Wi-Fi along with the video stream.

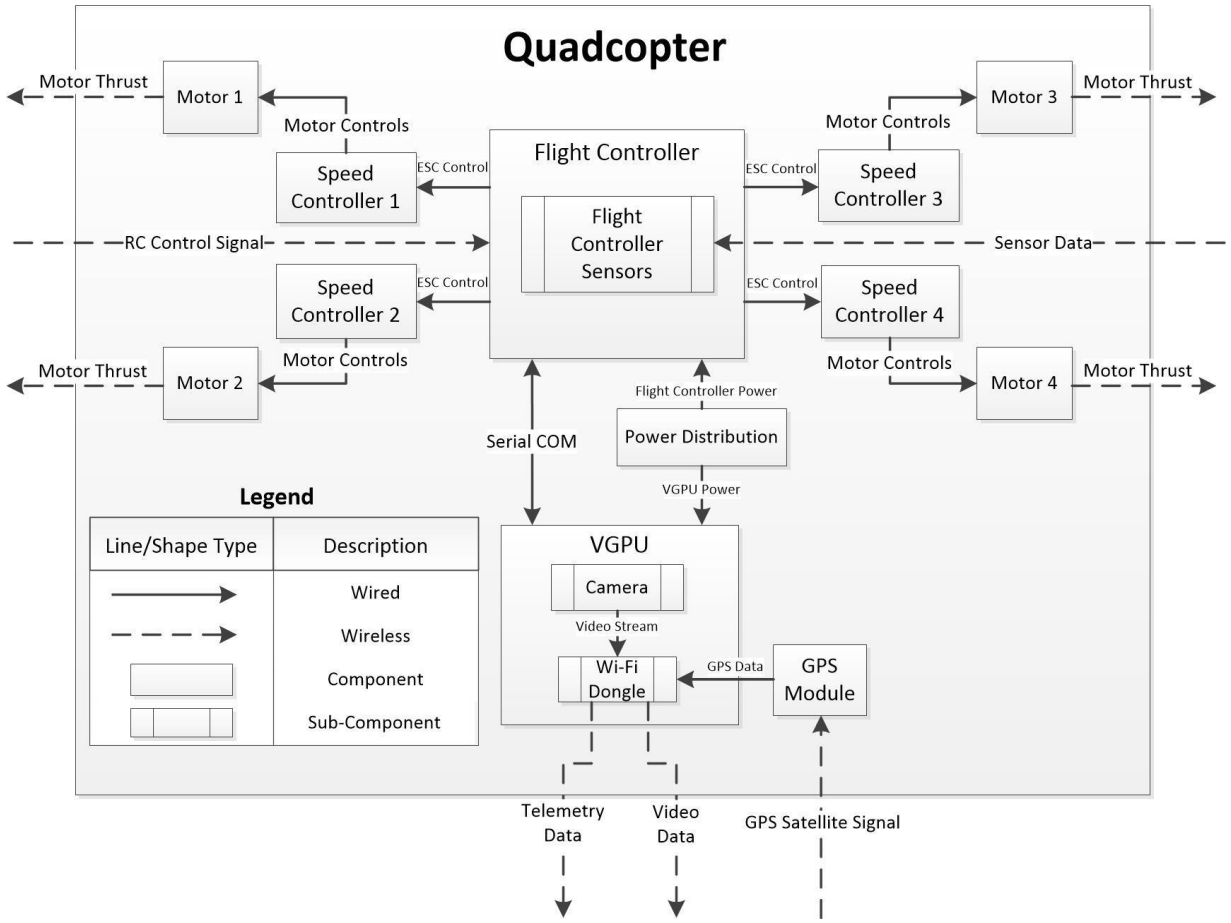


Figure 13: Quadcopter Hardware Block Diagram

Table 8: Quadcopter Hardware Block Diagram Interface Descriptions

Signal	Description	Protocol
Flight Controller Power	The power distribution system on the quadcopter provides a 5 VDC (volts direct current) power supply to the flight controller.	DC
ESC Control	The flight controller sends a pulse width modulated signal between one millisecond and two milliseconds to the ESC for each motor to control the movements of the quadcopter.[31]	PWM
Motor Controls	The ESCs send a three phase trapezoidal wave with varying frequencies to their respective motors to drive them.[32] The voltage ranges from 1.15 V to 1.45 V.[31]	Three-Phase Power
Serial COM	The VGPU and the flight controller communicate with each other over a serial connection in order to send telemetry data from the flight controller to the VGPU.	USB 2.0, MSP

VGPU Power	The power distribution system supplies a 5 VDC +/- 0.25 V power supply to the VGPU. The maximum current draw is 700 mA giving a maximum power consumption of 3.7 W.[33]	Micro USB
Video Stream	The camera sends a raw video feed to the VGPU for processing and preparation to be sent to the base station.	H.264 video[34]
GPS Data	GPS data is sent to the VGPU via its digital I/O pins. The voltage supplied to the GPS module is 5 V with a maximum current draw of about 48 mA resulting in power consumption of 0.24 W. Transmit and receive pins send the NMEA (National Marine Electronics Association) 0183 data at a 115,200 baud rate.[35]	NMEA 1083[24]

A block diagram of software communication is shown in Figure 14. You can see that the software resides on the VGPU, of which there is a video streaming block and telemetry streaming block, as well as the GPS module and MW flight controller. The telemetry streaming interfaces with these last two items to gather telemetry and GPS data and sends that over TCP/IP to the base station. The video streaming software gathers video from the VGPU camera and sends that over UDP/IP to the base station. Both video and telemetry streaming are controlled by SSH commands sent by the base station. All of the signals are described in Table 9 below.

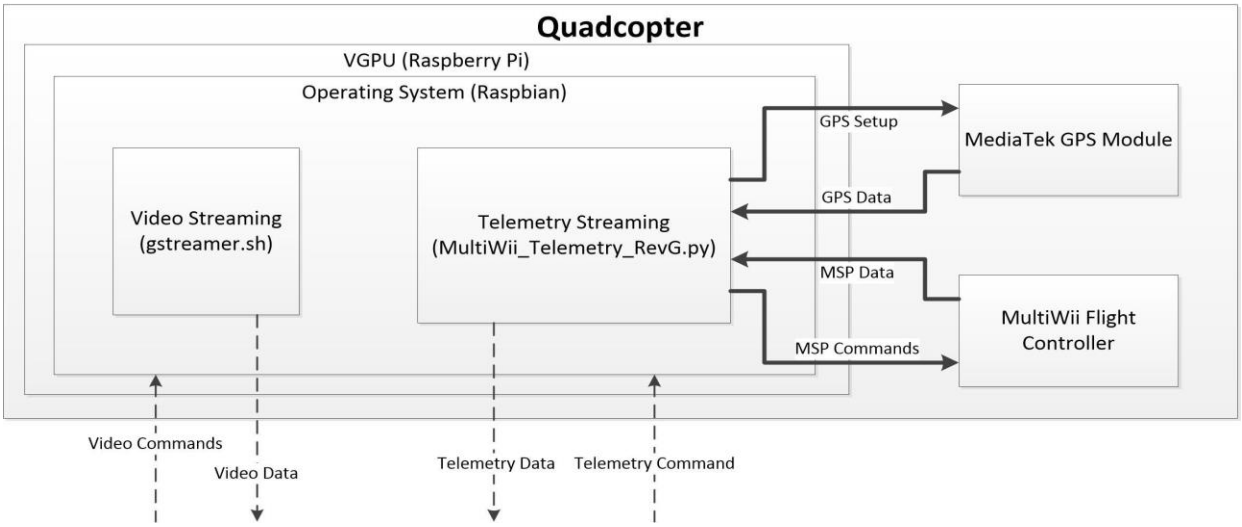


Figure 14: Quadcopter Software Block Diagram

Table 9: Quadcopter Software Block Diagram Interface Descriptions

Signal	Description	Type
GPS Setup	GPS setup is a command sent once to set the refresh rate of the MediaTek GPS module to 10 Hz and the baud rate to 115,200.	PMTK NMEA 1083 String[24]
GPS Data	The NMEA string is sent from the MediaTek GPS module to the Raspberry Pi at 10 Hz.	NMEA 1083 String[24]
MSP Commands	The Raspberry Pi needs to gather data from the MultiWii flight controller using the MultiWii Serial Protocol. Using this protocol the Raspberry Pi gets the heading, altitude, and RC input values of the quadcopter.	MSP String
MSP Data	The MultiWii flight controller sends the desired data back to the Raspberry Pi using the MultiWii Serial Protocol.	MSP String

4.2 Quadcopter Components

This section discusses the design process behind quadcopter hardware and software component choices. Figure 15 shows a picture of some of the components that are mounted at the center of the quadcopter.

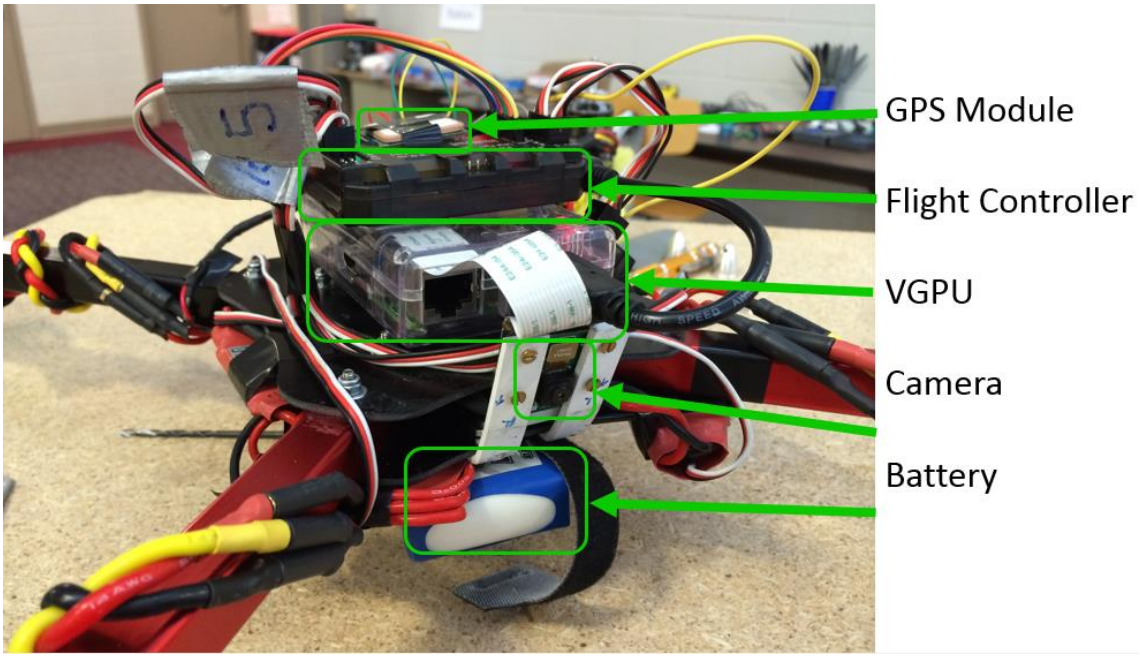


Figure 15: Central Quadcopter Components

4.2.1 Flight Controller

4.2.1.1 Hardware

A flight controller is used to interpret RC controls, provide telemetry data to the base station, as well as provide dynamic control feedback to keep the quadcopter stable during flight.

4.2.1.1.1 Requirements

The flight controller shall support GPS navigation. The flight controller shall use software that can be manipulated by the user. That is, the team shall have access to the flight controller code and modify it per project needs. In addition, it will aid in satisfying the following customer requirements: REQ 2.2.1.2.A, REQ 2.2.1.2.C.P1, REQ 2.2.1.2.D, REQ 2.2.1.2.E.P1, REQ 2.2.1.3.E.P1, REQ 2.2.1.4.C.P1, REQ 2.2.3.2.A, REQ 2.2.3.3.B.P1, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, REQ 2.2.2.2.A, and REQ 2.2.2.3.A.

4.2.1.1.2 Alternatives

The team investigated using AeroQuad 32, ArduPilot Mega (APM) 2.6, AutoQuad 6, MultiWii Lite, MultiWii Plus, MultiWii Pro, PX4FMU Autopilot, and UAVX-ARM32 flight controllers. The team did not consider all flight controllers because there so many different types, so these eight were selected with their features laid out in Table 10.

Table 10: Flight Controller Alternatives

Controller	Cost	Gyroscope	Barometer	Magnetometer	Waypoint Capabilities	Experience
AeroQuad32[36]	$149.95	MPU6000	MS5611	HMC5983-TR	Add-on	None
APM 2.6[37]	$159.99	MPU6000	MS5611	HMC5883L	Yes	Yes
AutoQuad 6[38]	$406.25	IDG500/ ISZ500	MP3H6115A	HMC6042/ HMC1041Z	Yes	None
MultiWii Lite[39]	$26.49	ITG3205	None	None	None	None
MultiWii Plus[40]	$27.77	ITG3205	BMP085	HMC5883L	None	Yes
MultiWii Pro[41]	$64.99	ITG3205	BMP085	HMC5883L	Yes	None
PX4FMU AutoPilot[42]	$149.00	MPU6000	MS5611	HMC5883L	Add-on	None
UAVX-ARM32[43]	$96.50	MPU6050	MS5611	HMC5883L	Add-on	None

4.2.1.1.3 Decision Criteria

The team evaluated the flight controllers based on their price, sensors included, waypoint capabilities, and previous experience with the controller. Weight difference was negligible (~5 g) so weight was not used as a decision criteria. The firmware that the flight controller runs is important, but in

order to test this the team would have needed to individually purchase and test all of the alternatives, which would have surpassed our budget and time constraints.

4.2.1.1.4 Decision

The team was able to use an APM 1.4 flight controller for prototype 0.1. This is an earlier version of the APM 2.6, and it gave the team experience using an APM flight controller. Over interim, the team had a chance to use the MW Plus flight controller. This controller was used for prototype 0.2, and it proved equally effective in providing a stable flight. Both flight controllers also provided similar flight modes such as altitude-hold, heading-hold, and various stability and GPS flight modes. MW also provided better accessibility to the source code for the flight control software, giving the team better access to make changes to that software. Because of the experience gained with MW and APM, the team ruled out the other flight controllers. This choice was made to mitigate start-up time if a different flight controller was chosen. Because the MW proved as effective as the APM, but was $130 less expensive, the team decided to go with the MW brand.

The team's final decision was not to go with the MW Plus, however. The team needed a flight controller that could interface with a GPS module, and the MW Plus could not do this. Because of this, the team chose the MW Pro flight controller. This still offered the familiarity with the MW software, as the same source code can go on either, but it offered additional serial ports that could interface with a GPS module as well as including more memory. Though the price of the Pro was about $30 greater than the Plus, it included a GPS module which would have cost at least that much. This made the MW Pro an easy choice since it could interface with a GPS module and cost about the same as the MW Plus when GPS cost was factored separately.

4.2.1.1.5 Implementation

The team purchased the MW Pro in February and installed it onto the quadcopter inside of a plastic case connecting it to the ESCs and the VGPU as shown in Figure 13 and Figure 15 above.

4.2.1.1.6 Unit Test

While the basic flight control of the MW was tested simply by PID (proportional-integral-derivative) tuning and flying the quadcopter, there were a number of sensors that needed to be tested since they were used for the tracking algorithms. The barometer and compass on-board the flight controller were unit tested to ensure optimal performance. These tests and are discussed below.

4.2.1.1.6.1 Barometer

The first sensor to test was the BMP085 (barometer).[41] This sensor uses atmospheric pressure to estimate the altitude of the quadcopter in meters. This sensor is needed to implement altitude-hold, a flight mode that attempts to keep the quadcopter at the same altitude. The team tested three methods to improve the accuracy and precision of the barometer; control (no action), adding a foam sponge cover, and adding a mesh cover. The mesh cover proved ineffective as it made the altitude-hold worse by inspection. The remaining two options were tested in a number of different ways. One test was flying the quadcopter up to two meters and seeing which method provided better altitude-hold. The results are seen in Table 11 below.

Table 11: Altitude-Hold Tests with Control and Foam Covers (meters)

	Control	Foam
Average	0.83	1.90
Median	0.78	1.94
Standard Dev.	0.45	0.31

Standard deviation was considered the most important metric while testing. While accurate values would be ideal, the barometer itself measures only absolute pressure, thus it was more important that the sensor's standard deviation was the smallest as this signified the best hold at a given altitude. As is shown from the results, a foam cover improved the barometer in both accuracy and precision, and thus was implemented on the flight controller. No alternative foam covers were explored due to time constraints for the prototype. The barometer tests are seen in Appendix A.5.1.

4.2.1.1.6.2 Compass

The other sensor that underwent unit tests on the flight controller was the HMC5883L (compass).[41] In order to track the vehicle correctly, one must know where the quadcopter is pointed so that it can turn to follow the vehicle. Because of this, unit tests were performed to ensure the accuracy and precision of the compass. It became quickly obvious that the compass was not operating correctly when the first flights were completed. Under motor power, the compass value would skew over 50 degrees on a 360-degree scale. After preliminary tests, and discussions with professors on campus as well as research online, the source of the interference was found to be magnetic fields emitted from the power distribution board. The best way to shield against this interference was simply to increase the distance between the compass (and the flight controller on which it is mounted) and the source of interference. Because of this change, the VGPU was mounted directly onto the quadcopter platform and the flight controller mounted on top of that. This final layout for prototype 1.0 is seen in Figure 15 above.

Once the quadcopter was reassembled, the compass was tested again to prove the changes. The skew resulting from motor power was under two degrees, much improved from the original case. The quadcopter was then flown in a heading-hold mode. The purpose of this flight mode is to keep the quadcopter's orientation throughout a flight. With this mode activated, compass values were recorded and the results are seen in Figure 16 below.

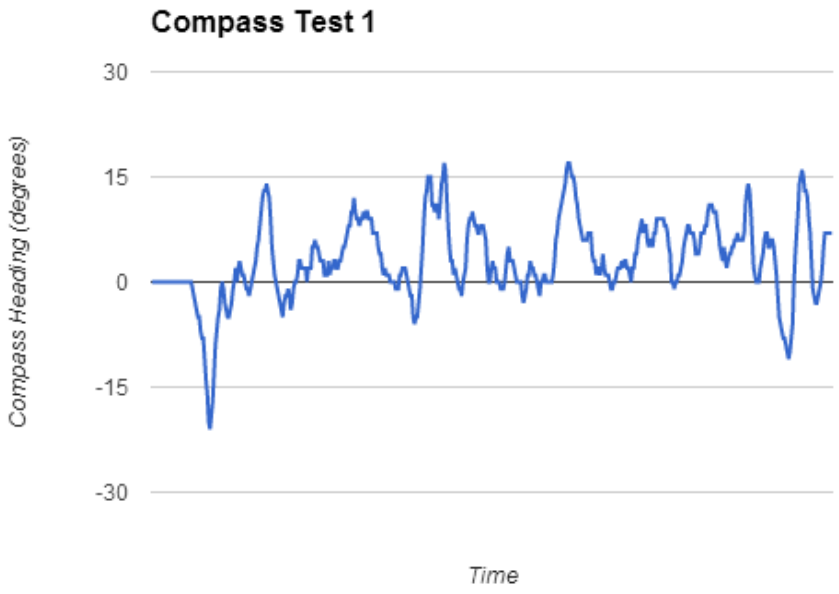


Figure 16: Compass Values with Heading-Hold Mode

As you can see from the figure above, the quadcopter was able to hold the quadcopter at a zero-degree heading fairly well. While the PID control of the heading-hold mode does have an effect on this test, the important thing to notice is that under power, and with varying throttle inputs, the quadcopter held orientation and continued to correct to zero degrees. A full list of the compass tests completed are seen in Appendix A.5.2.

4.2.1.2 Software

The flight controller software runs on the flight control board. It controls the quadcopter and handles all of the dynamic feedback control loops.

4.2.1.2.1 Requirements

The flight controller software shall satisfy the following customer requirements: REQ 2.2.1.2.A, REQ 2.2.1.2.D, REQ 2.2.1.2.E.P1, REQ 2.2.1.3.E.P1, REQ 2.2.3.1.B, REQ 2.2.3.2.A, and REQ 2.2.3.3.B.P1.

4.2.1.2.2 Alternatives

The software used on the flight controller is most directly related to the hardware itself. Most flight controllers come with their own custom flight software. The MW board has two main software options: MultiWii and MegaPirate. Both options offer similar flight modes and overall functionality

4.2.1.2.3 Decision Criteria

The highest weighted decision criterion was the experience that the team had of the software. Having open source software was another factor, as that allows for changes for the specific project.

4.2.1.2.4 Decision

The team chose to remain with the MultiWii software installed on the flight controller. This was used on prototype 0.2, and as such, the team had experience and was more familiar with it than MegaPirate. However, the team did upgrade from V2.2 to V2.3 when they purchased the MW Pro. V2.3 was the most recent, stable release, so it made sense to have that software package. Another nice feature of this new version is that the barometer data calibrates to zero each time the quadcopter is powered, making the altitude data relative to ground during flight.

4.2.1.2.5 Unit Test

There were no specific unit tests performed on the MultiWii software. Once loaded onto the flight controller, the quadcopter was PID tuned in order to maximize stability and flight response. The software functioned as expected and provided stable and responsive quadcopter flight.

4.2.2 GPS Module

The GPS module is responsible for obtaining quadcopter longitude and latitude data required for tracking. The telemetry streaming software sends this data down to the base station, which compares it to the vehicle's position.

4.2.2.1 Hardware

4.2.2.1.1 Requirements

The GPS module shall have the accuracy necessary to track the vehicle's position within 7.2 meters. The GPS shall refresh its location with a frequency of 10 Hz. It shall also be capable of interfacing over digital I/O pins. In addition, the GPS module will help satisfy the following customer requirements: REQ 2.2.1.2.C.P1, REQ 2.2.1.3.D.P1, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, REQ 2.2.2.2.A, and REQ 2.2.2.3.A.

4.2.2.1.2 Alternatives

Many GPS modules use digital I/O pins to transmit and receive data. Specifically the team considered an Adafruit GPS breakout board, which costs $39.95.[38] All alternatives considered use the NMEA 1083 protocol. The Adafruit GPS board operates at between 3.0-5.5 V and has a maximum current draw of 25 mA giving a typical power consumption of 152 mW at 5 V. The Adafruit board claims an accuracy of less than three meters with a maximum refresh rate of 10 Hz.[44] Another GPS board that was considered was the MTK 3329 GPS module. This module is part of a package deal that comes with the MW Pro flight control board. It is a GPS module with maximum current draw of 48 mA, which at 5 V results in a power consumption of 240 mW. This module also supports a refresh rate of up to 10 Hz.[45]

4.2.2.1.3 Decision Criteria

Cost, refresh rate, accuracy, weight, and power were all factors affecting in choosing the best alternative.

4.2.2.1.4 Decision

The team chose the MediaTek 3329 GPS module. Because the MediaTek GPS module came with the flight controller, it was an easy decision. Weight was not a significant issue since the difference in weight was not significant (~1 gram). It met the required accuracy and refresh rates, and was proven to interface with the MW Pro flight controller. The Adafruit GPS was also purchased but for the base station instead. This decision is discussed in section 5.2.4.

4.2.2.1.5 Unit Test

No formal unit tests were performed. In the initial setup of the GPS module, the latitude and longitude values were compared to that of a map and proved accurate. As seen in Chapter 6, the GPS module was used in tandem with full communication to test the system response and positional precision.

4.2.2.2 Software

There were only minor amounts of software tweaks that occurred for the GPS module. PTK NMEA messages are sent in the GPS setup command that increase the baud rate of the device to 115,200 and increase the refresh rate from factory default 1 Hz to maximum 10 Hz. This increase was done to improve the latency of the feedback in the tracking algorithms.

4.2.3 Propellers

4.2.3.1 Requirements

The propellers shall be large enough to provide adequate lift for the quadcopter, but small enough to fit on the chosen frame. Propellers are also specific to the direction of rotation, making it necessary to match propellers with motors. The propellers used will aid in satisfying the following customer requirements: REQ 2.2.1.1.A.P1, REQ 2.2.1.2.B.P1, REQ 2.2.1.3.E.P1, REQ 2.2.2.1.A, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, REQ 2.2.2.2.A, and REQ 2.2.2.3.A.

4.2.3.2 Alternatives

Carbon fiber or plastic propellers both fit project needs. There are propellers that range in length from 4 to 22 inches and with pitches ranging from 2 to 12 degrees.[46]

4.2.3.3 Decision Criteria

The choice between plastic or carbon fiber propellers depended on multiple factors. The blades needed to be robust enough to handle moderate collisions, balanced enough to limit vibrations, and have appropriate length and pitch values. Motors driving the propellers are rated for specific propeller sizes, so this was taken into effect as well. Larger propellers (and those with a higher pitch) can provide more lift because they move more air, but they also require more power. Cost was another factor, as multiple crashes were anticipated with a new quadcopter design.

4.2.3.4 Decision

The team choose to continue using plastic propellers. Carbon fiber propellers are more expensive and much stronger, making them even more dangerous if they were to contact an object or a person. Due to the nature of the project with many new components of hardware and software coming together, the team anticipated multiple crashes. Keeping this in mind, the team chose plastic propellers to help stay under budget while providing a safer product for the user. The team specifically selected APC propellers for their build quality. The length chosen was 25.4 cm (10 in) with a pitch of 4.5 degrees as this provided the best balance of lift without sacrificing stability.

4.2.3.5 Unit Test

Propellers were not tested in a specific way other than during test flights of the quadcopter. The team observed that propellers longer than 25.4 cm and with a pitch greater than 4.5 degrees made the quadcopter too sensitive to user input. However, the larger the propeller, the more lift the quadcopter experienced. As such, the team settled on 25.4 cm, 4.5 degree propellers because they have good lift while leaving the quadcopter stable.

4.2.4 Motors

Four motors drive the propellers and provide thrust for the quadcopter.

4.2.4.1 Requirements

The motors shall be powerful enough to spin the propellers, lift the quadcopter, and move the quadcopter at the required speed of 50 km/hr for the production model, and 15 km/hr for prototype 1.0. In addition, the motors will help aid in satisfying the following customer requirements: REQ 2.2.1.1.A.P1, REQ 2.2.1.2.B.P1, REQ 2.2.1.3.E.P1, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, REQ 2.2.2.2.A, and REQ 2.2.2.3.A.

4.2.4.2 Alternatives

On prototype 0.1, the motors were 850 Kv brushless motors with a 3.17 mm diameter shaft. The weight of each motor was 62 grams. The motors used on prototype 0.2 were 1200 Kv (revolutions per Minute / Volt) brushless motors with a 3 mm diameter shaft and a weight of 57.6 grams. The max current draw is 17A.[47]

4.2.4.3 Decision Criteria

The motors chosen for the final design depended on weight, power (Kv), current draw, and cost. The motors must have a maximum current draw lower than the ESC output rating. The shaft diameter is another factor as a thicker shaft makes for a more durable motor.

4.2.4.4 Decision

Ultimately, the decision was made to use the motors from prototype 0.2. Since the frame and power distribution system were the same between prototype 0.2 and prototype 1.0, these motors were already proven with these components. The team was very grateful because these motors were donated to the project by Professor Yoon Kim.

4.2.4.5 Implementation

Motors are mounted to the end of the quadcopter's four arms as shown in Figure 12. They are each connected to an ESC with three wires. The order of wiring simply affects the direction that the motor turns. As such, two motors (opposite each other) are connected to spin counter-clockwise and two connected to spin clockwise. See the quadcopter block diagram in Figure 13 and accompanying Table 8 for more information. Figure 17 below shows the direction of the motors on prototype 1.0

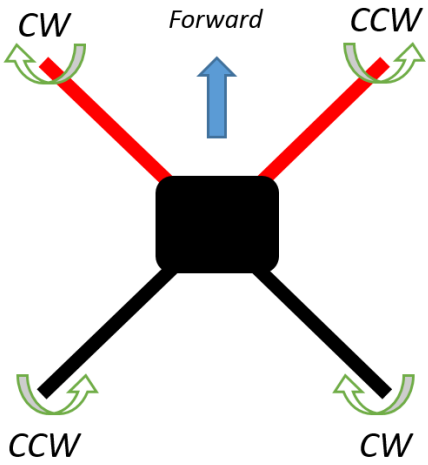


Figure 17: Prototype 1.0 Motor Direction

4.2.4.6 Unit Test

They motors were tested to lift the quadcopter and its components successfully (1.66 kg). Above that, the motors were able to lift an additional 0.9 kg. The prototype was able to fly over 15 km/hr meeting the prototype requirement, but the maximum speed was not tested out of concern for a crash that could occur during this test.

4.2.5 ESCs

ESCs regulate power, and thus speed, for each motor.

4.2.5.1 Requirements

The ESC's shall provide enough current to power the motors effectively. This means the ESC's must have a higher current rating than the motors they control. In addition, the ESC's will help satisfy the following customer requirements: REQ 2.2.1.1.A.P1, REQ 2.2.1.2.B.P1, REQ 2.2.1.3.E.P1, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, REQ 2.2.2.2.A, and REQ 2.2.2.3.A.

4.2.5.2 Alternatives

ESCs can be found with different current values and are made by several manufacturers. The team first 3DRobotics as a vendor because of their short lead-time on components. 3DRobotics only sells one ESC, a 20A ESC with a battery eliminator circuit (BEC) output of 5 V and 2 A for $25.99.[50] The ESCs used for prototype 0.2 were Turnigy Plush 25 A ESCs with a BEC output of 5 V and 2 A, and at a cost of $13.44, from HobbyKing.[48]

4.2.5.3 Decision Criteria

The major criteria for ESCs were current output, cost, compatibility with the selected motors, and lead-time. The ESCs also provide BEC capabilities, but this is discussed in the power distribution system in section 4.2.7. The ESCs also needed to be compatible with the voltage of the battery used in order to power the motors. The current output was a major criterion because the ESCs needed to provide at minimum the same current than the motor can draw (17 A for the motors selected). Cost was a major deciding factor for both device and shipping. Where the parts ship from was another factor, as it directly influences lead-time on parts.

4.2.5.4 Decision

While both ESCs provide the required current output, the team decided to use the Turnigy ESCs because they cost half of the price of the 3DRobotics ESCs. The disadvantage to these is their lead-time ranging from two to four weeks as they are from HobbyKing, but because of the supply from our interim class, this was not a major concern.

4.2.5.5 Implementation

Four ESCs were purchased, with each one being connected to a specific motor. These ESCs are powered from the power distribution board on the quadcopter and controlled from the flight controller.

4.2.5.6 Unit Test

The ESCs met the required current draw of the motors to which they were connected. Through a simple programming setup, the ESC's were matched with the throttle commands of the RC transmitter. The BEC output of each ESC was also verified with a digital multimeter to ensure it met the required 5 V output.

4.2.6 Battery

The battery provides stable voltage, high current power to all of the components on the quadcopter.

4.2.6.1 Requirements

The battery powering the quadcopter shall provide power for all on-board sensors and computers, as well as the ESCs and motors. The battery shall provide power for the equipment for a minimum of ten minutes without overheating for prototype 1.0. The battery shall also have protective circuitry to prevent overcharging and over-discharging which can cause batteries to catch fire and explode.

In addition, the battery helps satisfy the following customer requirements: REQ 2.2.1.1.A.P1, REQ 2.2.2.1.A, REQ 2.2.3.3.E, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, REQ 2.2.2.2.A, and REQ 2.2.2.3.A.

4.2.6.2 Alternatives

Lithium ion polymer (LiPo) batteries are the standard power source for quadcopters so the team limited their focus to them.[49] Lithium ion polymer batteries have power capacities ranging from 1000 mAh to 8000 mAh.[50]

4.2.6.3 Decision Criteria

Cost was a portion of the decision for the battery chosen, but voltage output, discharge rate, capacity, and weight were major criteria as well. The battery needed to operate at a voltage of 11.1 V and have a discharge rate of at least 30 C. Batteries with larger capacities weigh more, creating a need for more power, and eventually diminishing the advantages of having a high capacity battery. Table 12 shows the capacity, weight, cost, and charge time of several batteries that were considered. All of the listed batteries had the required voltage and discharge rates needed. Charge time was estimated with 5 A at 11.1 V

Table 12: Battery Comparisons

Capacity (mAh)	Cost	Weight (g)	Charge Time (min)
2450[51]	$ 19.60	218	29.4
3000[52]	$ 21.79	269	36.0
3600[53]	$ 24.70	321	43.2
5000[54]	$ 39.99	414	60.0
11,975*	$ 90.00*	1050*	143.7*
**Estimated battery statistics for production model*			

4.2.6.4 Prototype Decision

The prototype decision was to use originally use a 3600 mAh battery, which was made available to the team by Professor Yoon Kim. This battery was chosen because it was free to use and provided a baseline the team could run capacity tests on in order to make a more educated purchase in the future. The team later acquired a number of other batteries from Professor Yoon Kim ranging from 2450 mAh to 5000 mAh to enable extended flight tests with only the switch of a battery. With the 5000 mAh battery, the team was able to reach the prototype flight time requirement of ten minutes, so this battery was the final team decision.

4.2.6.5 Production Decision

The team would likely source a custom battery that better fits within the quadcopter packaging. Based on the described tests in the next section, average current draw for the system was 23.95 A. With a flight time of 30 minutes, the estimated battery size is calculated below:

$$23.95\ A * \left(1000 \frac{mA}{A}\right) * \left(\frac{30\ min}{60\ min/h}\right) = \ 11{,}975\ mAh$$

This is a very large battery based off the current system specifications. The average cost per mAh was calculated to be $0.0075. Based on this, a cost of about $90 was predicted for this battery. The average grams per mAh was found to be 0.0877 grams/mAh. With this, the weight of the battery was estimated at 1.05 kg. This data is seen in Table 12 above. Motor tests were performed with the 5000 mAh battery, and the motors could lift an additional 0.9 kg. This new battery would add an additional 636 g (0.636 kg) compared to the 5000 mAh battery. While this extra weight is within the lifting ability of the motors, the team would likely need to look for power consumption savings as well as reductions in weight in order to meet the flight time requirement of 30 minutes.

4.2.6.6 Unit Test

The team had access to several different battery sizes from which they could perform battery life tests. Prototype 1.0, which included the VGPU and camera accessories, had an average current draw of 23.95 A. With the largest 5000 mAh battery, prototype 1.0 was able to meet REQ 2.2.1.1.A.P1 for flight time. Figure 18 below shows the flight time with the five batteries available for prototype 1.0 and Table 13 lists the results including the production battery. Documentation on all the battery life tests is found in Appendix A.1.

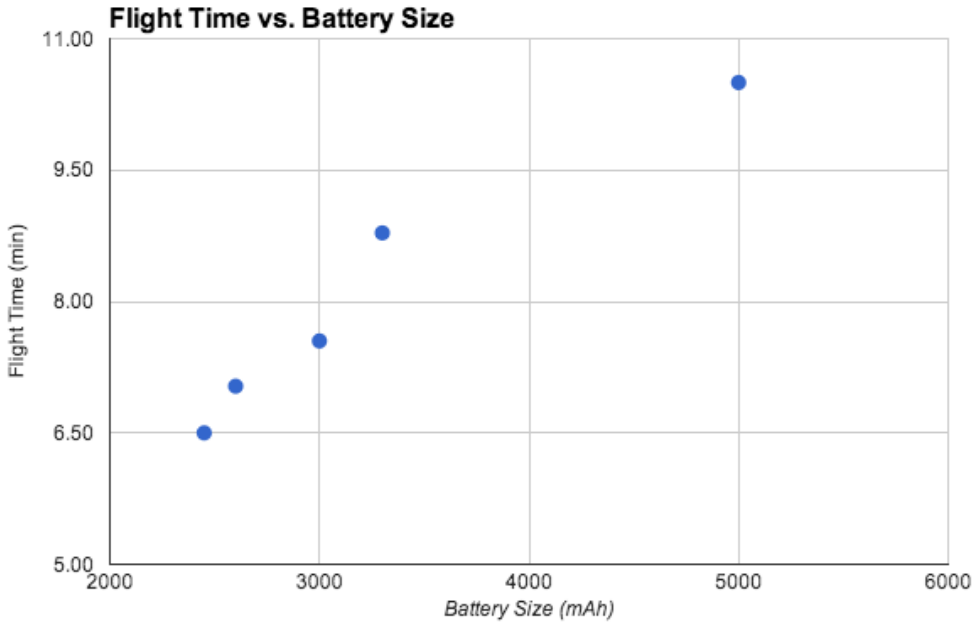


Figure 18: Prototype 1.0 Flight Times with various Battery Sizes

Table 13: Battery Capacities and Flight Times

Capacity (mAh)	Flight Time (min)
2450	6.5
3000	7.55
3300	8.78
5000	10.50
11,975*	30.00
**Estimated battery size for production model*	

The team completed calculations to determine how many batteries are needed to keep the production system aloft with quick battery changes every 30 minutes. Based on the charge time of 143.7 minutes and a flight time of 30 minutes, the total number of batteries needed is calculated below.

$$Batteries = \frac{143.7 \frac{min}{battery}}{30.0\ min} = 4.79 = 5\ Batteries$$

A total of five batteries are needed to keep the system aloft with only quick battery changes.

4.2.7 Power Distribution System

The multiple electrical components on the quadcopter are powered by a power distribution system that is connected to the battery.

4.2.7.1 Requirements

The power distribution system shall provide adequate and stable current and voltage that is required for all components on the quadcopter. See Table 14 for these specific requirements. In addition, the system uses 23.95A on average as described in Appendix A.1.

Table 14: Component Power Needs

	Flight Controller[41]	**RC Receiver[55]**	**VGPU[56]**	**ESCs[48]**
Voltage (V)	4.5-5.5	4.5-6.5	4.75 - 5.25	12
Max Current (A)	*Not Available*	*Not Available*	0.7	25

4.2.7.2 Alternatives

There were two main alternatives for this section: a custom printed circuit board or purchasing a power distribution board.

4.2.7.3 Decision Criteria

The decision was based on cost, size, weight, and ability to scale for more components. With the expandable nature of the project, the power distribution system needed to be able to handle extra components necessary for the project.

4.2.7.4 Decision

The team made the decision to first evaluate whether the existing board was able to power the VGPU along with the ESCs and flight controller. In order to add a level of redundancy to the power, the team chose to use a HobbyWing UBEC to power the VGPU, while using the BEC of the ESCs to power the flight controller. This later needed to be changed as three HobbyWing UBECs supplied between 5.25 and 5.3 volts to the VGPU, instead of the 5 volts specified online, making the product unusable as that was above the maximum 5.25 volts the VGPU can accept. Due to this malfunction, the team decided to use an extra Turnigy ESC's BEC to power the VGPU. Because the extra ESC was able to power the VGPU, the team chose to continue to use the existing board from prototype 0.2

4.2.7.5 Implementation

The board is mounted on the inner portion of the quadcopter and connected to all onboard equipment to supply power. The flight controller is powered through the original four ESC BECs while the VGPU is powered through the extra ESC added for its BEC capabilities. This reduces the load on the main four ESCs by up to 700 mA, as that is the maximum current draw of the VGPU.

4.2.7.6 Unit test

After replacing the malfunctioning UBECs with a Turnigy ESC, the power distribution system was able to power all quadcopter components successfully.

4.2.8 Frame and Landing Gear

The frame and landing gear provide a platform upon which all quadcopter components are mounted.

4.2.8.1 Requirements

The frame shall be strong enough to carry the components necessary to accomplish the project and be able to withstand the stress caused during landing. The weight of the quadcopter with all components is 1.66 kg. During a severe crash, it is estimated that the quadcopter would be falling at 2 $\mathrm{m/s}$ (and the landing gear would provide a deceleration distance of 4 cm due to the give in the landing gear and frame. Using $\Delta x = \frac{v_f + v_i}{2} * t$ (where v_f = final velocity and v_i = initial velocity) and solving for t gives $\frac{1}{25}$ s. The quadcopter changing from a velocity of 2 $\mathrm{m/s}$ to 0 $\mathrm{m/s}$ in $\frac{1}{25}$ s leads to an acceleration of $-50\ \mathrm{m/s^2}$. Provided this acceleration, and mass of 1.66 kg, the force that the arms would need to endure would be 83 N due to Newton's second law of motion: $F = ma$.

In addition, the frame shall help satisfy the following customer requirements: REQ 2.2.1.2.B.P1, REQ 2.2.1.3.E.P1, REQ 2.2.2.1.A, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, REQ 2.2.2.2.A, and REQ 2.2.2.3.A.

4.2.8.2 Alternatives

The team considered a designing and building a frame and landing gear in-house, or purchasing one. Alternatives included aluminum and carbon fiber, as well a variety of other materials that were not seriously considered due to low availability, weight, and cost. The team also considered the vendor to purchase from, as there are many frames available.

4.2.8.3 Decision Criteria

Price, strength, weight, rigidity, and expandability were all factors that determined the chosen frame. Expandability is defined as how well the quadcopter can expand to accommodate new components. Rigidity was a goal, as the initial implementation of propeller guards involved arms that extended beyond where the propellers could reach, and fishing wire was wound from each corner. This design ensured against collisions with flat surfaces as well as corners but had a major flaw. During an initial test, the fishing wire came loose and was caught in a propeller, bending the arm. The final decision needed to be one that a user can trust and depend upon. This follows one of our design norms of trust.

4.2.8.4 Decision

The team decided to purchase the frame because of the lack of a mechanical engineer on the team, but after deciding this, Professor Kim donated a frame to the project that was strong enough to withstand the 83 N as calculated above. The propeller guards were designed in-house as the donated frame did not include this. Prototype 0.1 helped with the decision as the team experienced problems that poor propeller guards and landing gear can cause. During initial stages of testing, the team flew the quadcopter for short periods in order to verify that it was stable and controllable. During these tests, the landing gear was able to handle small falls and minor hits without damaging the quadcopter. After major falls however, the landing gear began bending and it was obvious that the quadcopter needed a better means of protection (Figure 19). The propeller guards implemented in prototype 0.1 were arms that extend past where the propeller spins, ensuring safety in a collision with a flat surface. Fishing wire was

used, but due to the problems that it caused, it was not used in any later prototypes. The team also found it to be uncommon for the quadcopter to hit corners during flight, eliminating the need for a propeller protection between the ends of the arms.

Figure 19: Broken Arm

Prototype 0.2 had propeller guards similar to that of prototype 0.1 but without the fishing wire connecting the arms (Figure 20).

Figure 20: Propeller Guards with Vibration Problems (Prototype 0.2)

These were designed specifically to keep the quadcopter from flipping if it hit a wall at an angle, as the flat side would straighten the quadcopter. They also kept the quadcopter from landing on any of the sensitive components if it flipped during flight. The problem with these propeller guards was the

vibrations they caused. The extra aluminum used to keep it from flipping vibrated enough to loosen the motors from the frame arms. As such, the team removed the extra extensions for prototype 1.0. Figure 21 below shows the final propeller guard implementation.

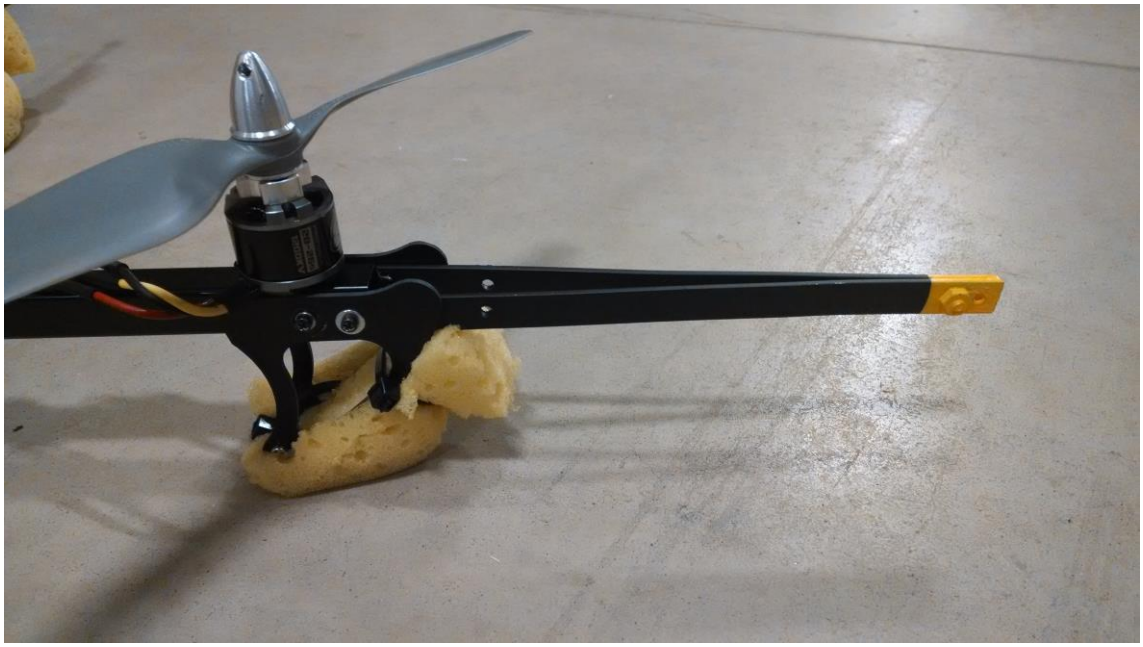


Figure 21: Final Propeller Guard Implementation (Prototype 1.0)

4.2.8.5 Implementation

The frame donated by Professor Kim was used as the base for mounting the quadcopter components. The landing gear consisted of foam attached to the bottom of the rounded feet that came with the frame and the final propeller guards were implemented as seen in Figure 21.

4.2.8.6 Unit Test

The frame was tested in each prototype as part of the quadcopter integration tests. The frame of prototype 0.1 did not pass the functional test as the arms were not sturdy and the propeller guards caused problems. Prototype 0.2's frame was sufficient, but the propeller guards caused vibrations leading to the final implementation that has both a sturdy frame and safe propeller guards that do not cause significant vibrations to shake the motors loose.

4.2.9 VGPU

4.2.9.1 Hardware

The VGPU is used to collect video and GPS data and sends that data from the quadcopter to the base station.

4.2.9.1.1 Requirements

The VGPU for this prototype shall be capable of sending 240p video and GPS coordinates to a computer. In the production device, 1080p video is necessary to provide adequate resolution for viewing surroundings. The device should also be capable of getting GPS coordinates from satellites and sending them to the base station. The VGPU must be able to communicate over Wi-Fi, whether through an

external attachment or built-in module. If an external device is used, a universal serial bus (USB) port is necessary to accommodate this hardware.

In addition, the VGPU helps satisfy the following customer requirements: REQ 2.2.1.4.A.PI, REQ 2.2.1.4.B, REQ 2.2.1.4.D.P1, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, REQ 2.2.2.2.A, and REQ 2.2.2.3.A.

4.2.9.1.2 Alternatives

Options include development boards such as an Arduino, a BeagleBone, or a Raspberry Pi. Development boards are ideal for this project because they limit the amount of board and integration design needed. These three were selected for comparison due to their popularity in the electronics industry, their relatively small sizes, and the design team's prior experiences. None of these modules includes Wi-Fi modules built in, due to the cost and size associated with boards in that category.

Raspberry Pi computers are available with up to 512 MB of random access memory (RAM) and 20 usable I/O ports.[56] BeagleBone development boards are available with 512 MB of RAM, as well as 92 I/O pins. Arduino development boards are available with between twenty and sixty I/O pins; however, they only provide small amounts of RAM (less than 1 MB).[57] Raspberry Pi computers can run a variety of operating systems with the most popular one being Raspbian (a Linux environment tailored toward the Raspberry Pi). The BeagleBone Black ships with Linux kernel 3.8 but can support a number of different operating systems that provide a versatile environment for development.58 Both the Raspberry Pi and BeagleBone have cameras that easily interface with the boards, with the Raspberry Pi camera costing $25

In a final implementation, the team would consider a custom board as opposed to using a development board. The team did an initial estimation of what components would be needed for a custom board and what the layout, weight, and cost savings might be for this custom board (Figure 22). The estimate includes only the major components such as the connectors and integrated circuits as they are the primary factors that make up the weight and the size of the board.

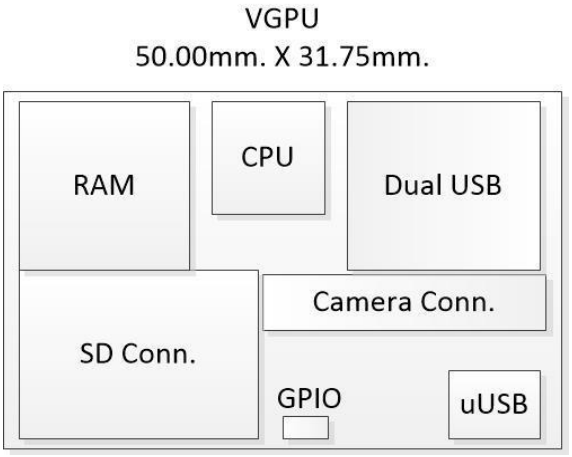


Figure 22: Custom VGPU Layout

The final dimensions of the estimate seen in Figure 22 are 50 x 31.75 mm, which is a decrease of 67% from the 85.6 x 56 mm of the Raspberry Pi. By removing the Ethernet port, 3.5 mm jack, HDMI port, RCA jack, one of the 15-pin connectors, and 22 of the 26 GPIO pins an estimate of 24 grams was removed, cutting the weight of the VGPU in half (the Raspberry Pi weighs 46 grams). The cost saved by removing these components was estimated to be $8.95 by finding sourcing components on DigiKey. The cost of making 600 four-layer boards (see section 8.8) of this size is estimated to cost $1.40 per board,

which is just less than half the estimated cost of the full Raspberry Pi board.[61] However, assuming design time of 500 hours at $100 dollars per hour, the total design costs would be $50,000. With a cost savings of only $10.35 per board, the total savings would be $\$10.35 * 600 = \$6{,}210$. Despite the weight savings of over 50% from a custom board, the additional cost of $43,000 does not justify this work. Unless a higher volume was produced to amortize the design cost, the Raspberry Pi would be used.

4.2.9.1.3 Decision Criteria

The major criteria for choosing a device were cost, hardware capabilities, development environment, size, and I/O ports. There was to a tradeoff between these functions as different options were considered.

4.2.9.1.4 Decision

The team evaluated the amount of RAM used after designing the prototype and found that about 89 MB of RAM was used to stream telemetry data along with 480p video. For the production model, the team estimates approximately double the RAM usage. This extra RAM is used for data needed for obstacle avoidance and other expandable features.

The team chose to use a Raspberry Pi as it has a camera made for it and a team member had some experience using one coming into the project. Arduino boards simply do not have enough RAM to process video streaming. The BeagleBone was ruled out because of its costly camera of $88 and lack of a second USB port. The Raspberry Pi Model B was picked over Model A because of the increased RAM and the extra USB port. The second USB port was important because one USB port was needed for serial communication with the flight controller while another houses the Wi-Fi dongle.

4.2.9.1.5 Implementation

The Raspberry Pi is mounted to the platform of the quadcopter in a protective plastic case and powered by the battery through the BEC of an ESC, which provides 5 V and up to 2 A to the board. This is sufficient, as the maximum current drawn by the Raspberry Pi is 700 mA.

4.2.9.1.6 Unit Test

The test of the Raspberry Pi was a pass-fail test to confirm that it could be powered through the BEC and that it was able to stream both video and telemetry data to the GUI. The Raspberry Pi passed this test under the final power distribution implementation (see Section 4.2.7 for power distribution design and test).

4.2.9.2 Software

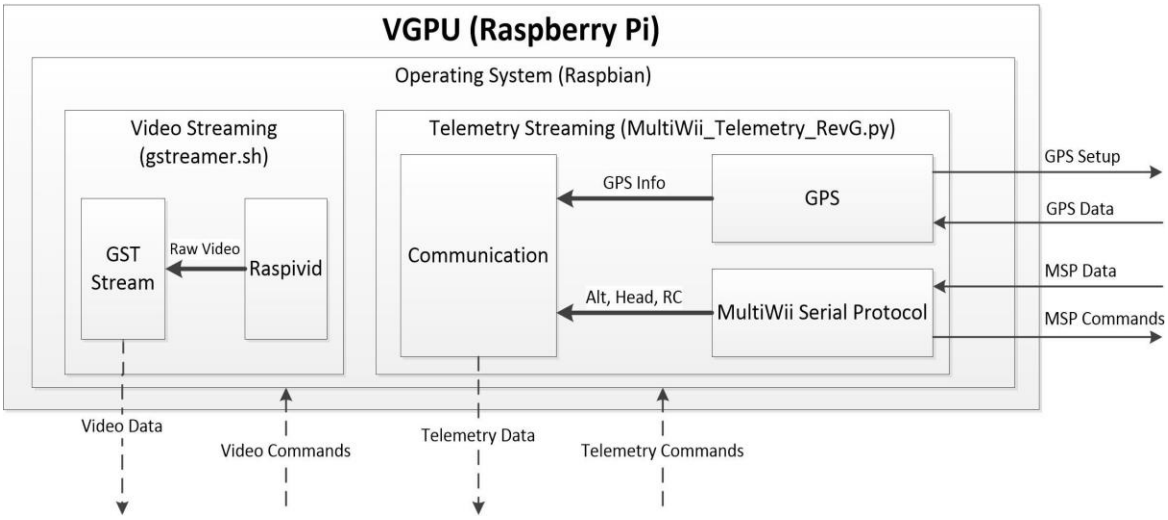


Figure 23: VGPU Software Block Diagram

Table 15: VGPU Software Block Diagram Interface Descriptions

Signal	Description	Type
Raw Video	The raw video captured by the raspivid command is forwarded to the GStreamer pipeline for compression to send to the base station.	Raw Video
GPS Info	The NMEA string from the GPS module is parsed and the relevant data is sent on to the communication function where it is joined with the Altitude, Heading, and RC input data.	String See Below (Table 16)
Alt, Head, RC	The telemetry data (altitude, heading, and RC input values) from the MultiWii is converted to string type and then sent to the communication function where it is concatenated with the GPS data and then sent down to the base station.	String See Below (Table 16)

Table 16: VGPU Protocols

Protocol	Description
GPS Info	This data includes the variables of longitude, latitude, timestamp, and number of satellites. Longitude and latitude are in the GPS form of degrees, minutes, and seconds, and the timestamp is coordinated universal time
Alt, Head, RC	Alt is the altitude of the quadcopter recorded in meters and Head is the compass heading of the quadcopter recorded in degrees. RC consists of pitch, roll, yaw, and throttle values that are being received by the quadcopter.

4.2.9.2.1 Video Streaming

The team's final code is available for analysis and transparency on the website under code. Transparency is especially important in the video software as many privacy concerns arise. To combat this, the group will notify individuals if their faces appear in video recorded by the quadcopter. This way, video taken can be used for demonstration purposes throughout the project. As this video stream is used for security purposes in the production model, there is a tension between justice and transparency. The customer will not likely want to broadcast the video being captured as that could lead to security risks. The team tried to be transparent in the design of the system, but it is ultimately up to the user how transparent they will allow the system to be.

4.2.9.2.1.1 Requirements

The video stream for prototype 1.0 shall be of the quality of at least 240p with a latency of less than or equal to one second as defined in REQ 2.2.1.4.A.P1, and REQ 2.2.1.4.D.P1. The final implementation shall have 1080p quality, 60 frames per second, and a latency less than or equal to 250 ms as well as a horizon to horizon field of view as defined in REQ 2.2.1.4.A, REQ 2.2.1.4.B, REQ 2.2.1.4.C, and REQ 2.2.1.4.D.

4.2.9.2.1.2 Alternatives

The team investigated three separate video streaming options throughout the development of the prototype; a stream using RTSP over TCP/IP, using GStreamer over UDP/IP, and using FFmpeg over UDP/IP. These three alternatives were chosen because of the choice of a Raspberry Pi and its camera module as the VGPU.

There are a number of video compressions that can be used to decrease the bandwidth necessary to send the video. H.264 is a video compression technique that reduces the size of the data sent using a lossy compression. Without compression, it would take a bandwidth of 18 Mbps to stream 320x240 video at 30 fps and a bandwidth of 220 Mbps to stream high definition (1280x720) video at 30 fps.

$$320 * 240 \rightarrow 76{,}800 \frac{pixels}{frame} * 8 \frac{bits}{frame} * 30 \frac{frames}{second} = 18\ Mbps$$

$$1280 * 720 \rightarrow 921{,}600 \frac{pixels}{frame} * 8 \frac{bits}{pixel} * 30 \frac{frames}{second} = 220\ Mbps$$

H.264 can reduce the necessary bandwidth to approximately 1.8 Mbps (320x240) and 15 Mbps (1280x720).[62]

4.2.9.2.1.3 Decision Criteria

The criteria to choose a video streaming method included video quality, video latency, and ease of implementation into the GUI. The team saw latency as the most important criterion as a low latency is imperative for the final application of the prototype. The video quality was next most important because it is important to have a viewable video without many glitches.

4.2.9.2.1.4 Unit Test

All three video methods use raspivid when they are running on a Raspberry Pi, a built-in video capturing command, with parameters providing 480p video at 30 frames per second, with the exception of GStreamer, which is captured at 60 frames per second. They all have the option to use H.264 compression. H.264 was chosen, as it is the built-in compression type on the Raspberry Pi.

The first method of video streaming to the GUI tested was using RTSP[63] via TCP/IP. The RTSP stream was sent using a VLC library[64] to be implemented into the GUI using the C# VLC library. The

RTSP stream was able to successfully stream 480p video to the GUI, but the latency was 1.8 seconds on average (see section A.6.1 in the Appendix).

This high latency caused the team to pursue another video streaming option. GStreamer was the next choice because it looked as if it could provide very low latency. Unfortunately, GStreamer could not be implemented into a C# GUI, as the C# bindings had not been released yet. GStreamer is an open source library for streaming of multimedia.[65] GStreamer had less documentation and examples than RTSP video streaming making it harder to stream video from the Raspberry Pi. GStreamer builds a pipeline that encodes the video and sends it via UDP/IP. The pipeline is the way GStreamer sends the video stream in the format required, such as specifying the H.264 format, the IP address to send to, and the port number. The base station computer can receive the pipeline by receiving the data on the specified UDP/IP port and then decoding it. GStreamer provided a 480p video stream with a latency of 0.2 seconds. The problem with GStreamer was that it was not compatible with C# at the time it was originally investigated because the C# bindings were not yet released. To overcome this issue, the team used a button in the GUI to run the compiled C++ application. See section 5.2.1.2 for further details.

Again, the team looked somewhere else for video streaming with lower latency that was compatible with the C# GUI. Using UDP/IP to send the video seemed like a lower latency option because UDP/IP does not require acknowledge packets to be sent as TCP/IP does. The team hoped to decrease latency by streaming video similar to the RTSP stream over TCP/IP, but instead sending it over UDP/IP. This led to the use of FFmpeg[66] and psips. FFmpeg is a command line tool that can convert video formats, while psips is a tool that compresses the sequence parameter set (SPS) and picture parameter set (PPS) data into an H.264 stream.[67] The SPS and PPS data is needed so the receiver knows what type of video it is receiving and therefore properly decodes it. The RTSP protocol packages and sends the SPS and PPS data separately, so in order to stream H.264 video using FFmpeg, the SPS and PPS data needed to be integrated into the stream; this is what psips takes care of. The video stream using FFmpeg successfully streamed 480p video into the GUI but with similar latency to that of the RTSP stream (1.7 seconds).

For a comparison test between the various video streaming methods, see section 6.2 and Appendix A.6.2

4.2.9.2.1.5 Decision

Although the team would like to have the video incorporated into the GUI, it was more important to have a lower latency video, and this caused us to choose GStreamer as the video implementation. A separate window is therefore needed in order to view the video.

4.2.9.2.2 Telemetry Streaming

The telemetry software involves gathering sensor data from the quadcopter flight controller and the GPS module to provide GPS positional data for the quadcopter. This data is concatenated and sent over TCP/IP to the base station computer. It is broken up into three main functional software groups: GPS, MultiWii Serial Protocol, and Communication.

4.2.9.2.2.1 Requirements

The telemetry software must be able to successfully obtain altitude and compass data from the flight controller. In addition, it must gather the GPS position of the quadcopter. All of this data must reach the base station for tracking to occur. This satisfies requirements REQ 2.2.1.2.C.P1, REQ 2.2.1.2.E.P1.

4.2.9.2.2.2 Operating System

There are many OSs to consider for the Raspberry Pi such as Fedora Remix, Arch Linux, and Raspbian. Since both telemetry streaming and video streaming are happening simultaneously, an OS is needed to ensure both processes have time on the processor. Many of these OSs are readily available and free, so cost and availability were not major criteria. Familiarity was a factor in choosing an operating system, as well as the popularity and thus development community surrounding a particular OS. As such, the team chose to use Raspbian OS because of its deep support, popularity, and team familiarity.

4.2.9.2.2.3 Language

Python is one of several programming languages available on the Raspberry Pi. Other languages include the C family, Java, and Perl. Python was selected because of its ease of development, as well as the fact that one of the team members had prior experience with the language. Python also had the necessary serial and socket communication libraries necessary for this project.

4.2.9.2.2.4 GPS

The GPS section of code interfaces directly with the GPS module. The first major choice was where to interface with the GPS module. It could either be connected to the flight controller, or connected directly onto the VGPU. The module was originally placed onto the flight controller, but the data was no longer received in NMEA format because it passed through the MultiWii Serial Protocol (mentioned below). As such, there was no timestamp included with the GPS data. In order to debug some of the latency issues in later system integration testing, the timestamp of the GPS module was necessary. Because of this need, the GPS module was moved and connected directly to the Raspberry Pi digital I/O pins. The GPS module used the 5 V (pin 2), Ground (pin 4), UART (universal asynchronous receive/transmit) transmit (pin 6), and the UART receive (pin 8). Figure 24 shows a layout of these connections.

The software first ensures the GPS module is operating at a baud rate of 115,200 and a refresh rate of 10 Hz. This refresh rate was originally at 1 Hz, but was increased to 10 Hz in order to decrease the communication latency. After that, the software it gathers the GPS data on the serial port. It parses the data into separate longitude, latitude, number of satellites, and timestamp variables. With these variables updated, the communication section of the code can access them for sending to the base station.

3.3V	1	2	5V
I2C0 SDA	3	4	DNC
I2C0 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
DNC	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 21	13	14	DNC
GPIO 22	15	16	GPIO 23
DNC	17	18	GPIO 24
SP10 MOSI	19	20	DNC
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
DNC	25	26	SP10 CE1 N

Figure 24: Raspberry Pi Digital I/O Pins

4.2.9.2.2.5 MultiWii Serial Protocol

The MultiWii Serial Protocol (MSP) is a protocol used to communicate with the MW flight controllers. It defines specific message packets that allow external devices to communicate with the flight controller. Figure 25 shows the format of a data packet. By changing the message type, the telemetry code can access the many different data values from the sensors on the flight controller.

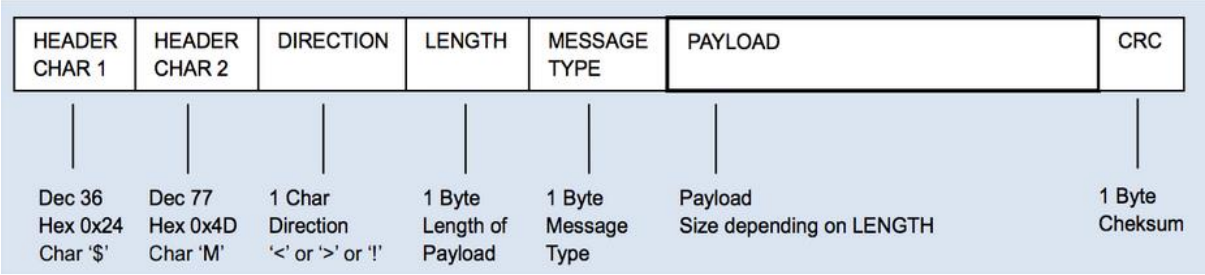


Figure 25: MultiWii Serial Protocol

The MSP section of the code handles the message passing between the Raspberry Pi and the MW flight controller. It gets the altitude data, heading data, and RC input values from the flight controller. This data is converted from ASCII, little endian form, and 2's compliment resulting in an integer. Once this is complete, the variables for altitude, heading, and RC (roll, pitch, yaw, and throttle) are updated for access from the communication section.

The Raspberry Pi was connected to the MW flight controller through a 5 V FTDI basic breakout board and a USB cable. From this, the breakout board is connected to the serial pins on the flight controller, which is highlighted in green in Figure 26 below.

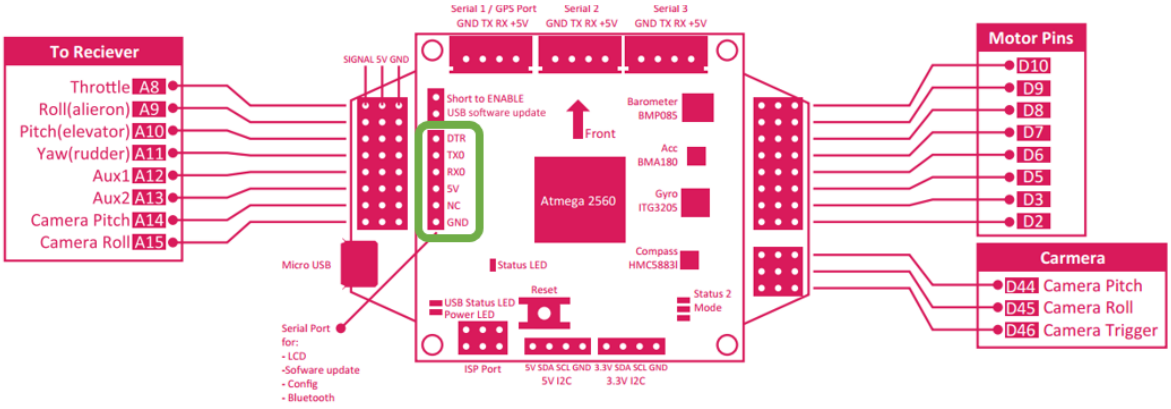


Figure 26: MultiWii Pro Connection

4.2.9.2.2.6 Communication

The communication section of the code handles the interface with the base station, specifically the GUI running on the base station computer. At the beginning of streaming, it establishes a TCP/IP connection with the GUI. The communication sends all of the data mentioned in the sections above. This data is sent at 10 Hz, which is limited by the GPS module. While the GPS module is the limiting factor, this data rate is more than adequate for the tracking needs of the system.

Table 5 discusses the complete data messages that are sent to the base station.

4.2.9.2.2.7 Unit Test

There were two main unit test performed on the telemetry streaming code. The first compared the MSP section code to that of the open-source MW GUI (used for PID tuning and changing flight modes). This MW GUI shows the current heading and altitude data when connected to the quadcopter. These values were compared to those of the telemetry streaming code to prove its effectiveness in gathering this data itself. The second test verified the GPS code by comparing the longitude and latitude values to that of a map.

4.2.10 Camera

A camera is used to record and stream video of the ground vehicle and its surroundings.

4.2.10.1 Requirements

The camera shall provide at least 240p quality for the prototype. The camera will help satisfy the following customer requirements: REQ 2.2.1.3.D.P1, REQ 2.2.1.4.A.PI, REQ 2.2.1.4.B, REQ 2.2.1.4.C.P1, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, REQ 2.2.2.2.A, and REQ 2.2.2.3.A.

4.2.10.2 Alternatives

Some possible camera solutions are the camera module for the Raspberry Pi,[68] a GoPro camera[69], or a USB webcam. USB webcams come in many shapes and sizes and range in price from $4.00 to $200.

4.2.10.3 Decision Criteria

The team made the camera decision based on the cost of the camera, the viewing angle, the weight, picture quality, maximum frame rate, and the ease of integrating the camera onto the quadcopter. Ease of integration includes mounting to the quadcopter as well as compatibility with the VGPU.

4.2.10.4 Prototype Decision

The team chose the camera module for the Raspberry Pi as it is a low cost camera. At only $25, it was much less expensive than a $200 GoPro and in the low range of USB webcam costs. It also integrates easily into the Raspberry Pi selected as the VGPU because it was designed for the Raspberry Pi. The field of view of this camera is only 42 degrees (see unit test below for calculation) but this is acceptable for the prototype.

4.2.10.5 Production Decision

For the prototype, the team is using the Raspberry Pi camera but the production design might use a different camera based on the testing results. The Raspberry Pi camera board costs $25, has a 42-degree viewing angle, weighs 0.13 ounces, can capture video in 1080p at 60 fps. It is also designed to integrate into the Raspberry Pi specifically. The least expensive, current GoPro camera (Hero3 White Edition) costs $200, with a 94-degree field of view, weighs 2.6 ounces, can capture 1080p video at 30 fps, and does not easily integrate into a Raspberry Pi. One specific USB webcam capable of providing 1080p video (Genius WideCam F100[70]) costs $41, claims a 120 degree field of view, weighs 2.9 ounces, captures 1080p video at 30 fps and has a USB connector that could connect to a Raspberry Pi but is not designed for one. The team selected the Raspberry Pi camera, as it is only 5% of the weight of the next lightest alternative, costs 38% less than the webcam, captures video at the required quality and frame rate, and is the simplest to integrate into the Raspberry Pi. To meet the viewing angle requirement, multiple cameras could be used and the image stitched together, or a wide-angle lens adapter added.

4.2.10.6 Implementation

The camera is mounted to the front of the quadcopter with an arbitrary downward viewing angle since there is no field of view requirement for the prototype (REQ 2.2.1.4.C.P1). This provides a good compromise between seeing ahead and seeing below. Figure 15 shows a picture of this implementation.

4.2.10.7 Unit Test

In order to test the viewing angle of the Raspberry Pi camera, the team measured the width and height of the field of view. The camera was set up 279 cm away from the wall and the field of view was marked. The width was 212 cm and the height was 140 cm. Solving for the angle of view yields a view of 42 x 28 degrees.

5 Base Station Design

This chapter discusses the design of the base station, which controls the quadcopter and receives the streaming video. Hardware and software design decisions are discussed for the base station portion of the project. A picture of the final base station is seen below in Figure 27.

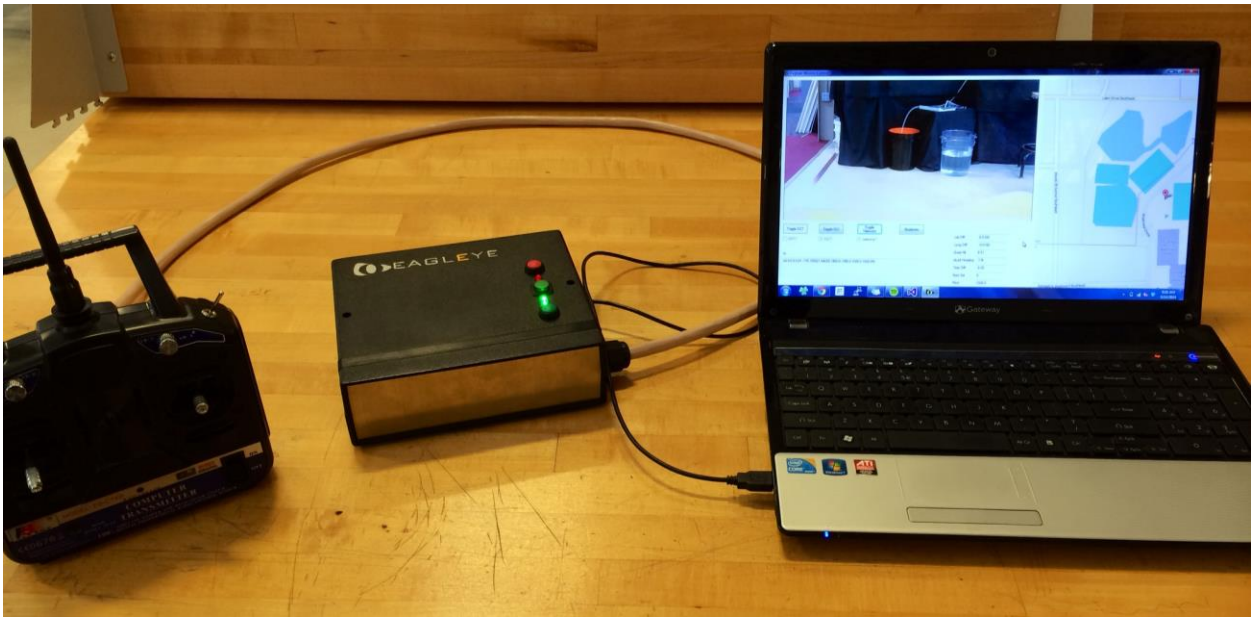


Figure 27: Final Base Station Implementation

5.1 Base Station Architecture

The base station hardware block diagram, seen in Figure 28 below, is composed of several main subsystems. Table 17 lists the signals and protocols between these subsystems. A laptop (part of the base station) is connected to an external microcontroller board via USB. Since there are no easily implemented products to control RC vehicles with a laptop, this board passes along controls to the RC transmitter. A GPS module interfaces with the microcontroller to indicate the position of the base station. Lastly, a Wi-Fi router provides the network over which streaming video and telemetry is received from the quadcopter.

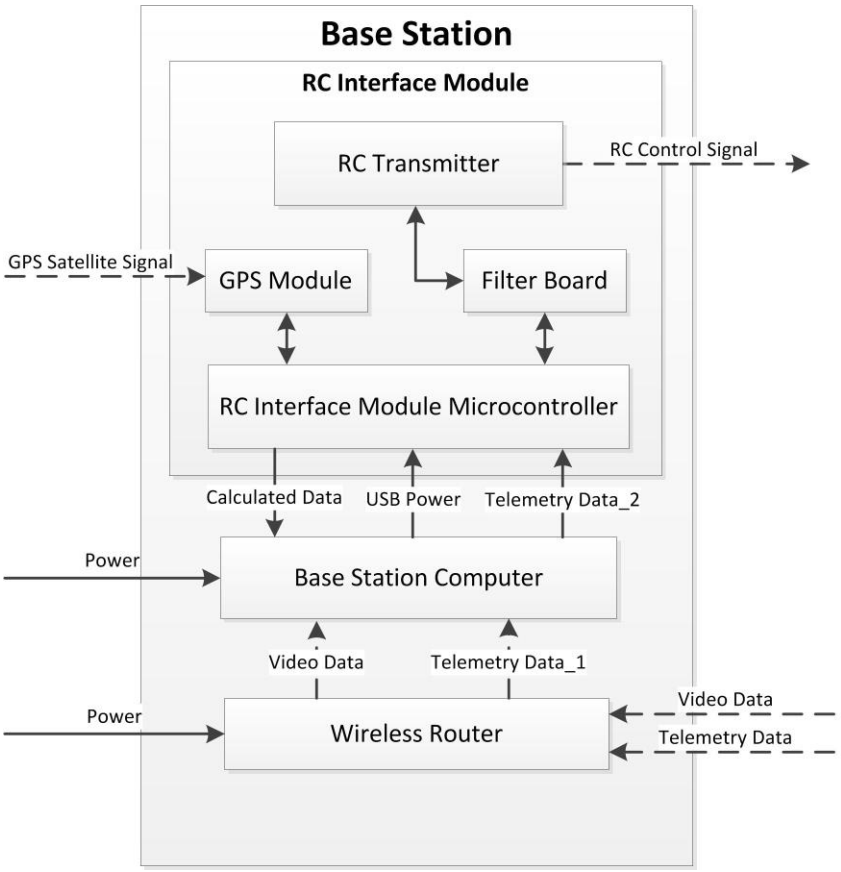


Figure 28: Base Station Hardware Block Diagram

Table 17: Base Station Hardware Interface Descriptions

Signal	Description	Protocol
Video Data	This is the video data signal received from the quadcopter via the wireless router. For details on this signal, see Figure 10 and Table 3.	802.11n
Telemetry Data_1	This is the telemetry data signal received from the quadcopter via the wireless router. For details on this signal, see Figure 10 and Table 3.	802.11n
Telemetry Data_2	This is a subset of the telemetry data received by the base station computer. Only the GPS data and the quadcopter's heading are sent from the base station computer to the RC interface module, as the other telemetry data is not needed.	Micro USB 2.0
Calculated Data	The difference in the GPS position of the quadcopter relative to the base station is sent from the RC interface module to the base station computer to be displayed in the GUI.	String See Calculated Data in Table 19

USB Power	The RC interface module is powered by USB from the base station computer	5 V
Power	A car power inverter powers both the base station computer and Wi-Fi router.	120 VAC 60 Hz

Figure 29, below, shows the software block diagram. There are two main software packages on the base station, based on the hardware platform: the base station computer and the RC interface module. The GUI runs on the base station computer and it contains two main software blocks: video streaming and telemetry streaming. The video streaming block handles the UDP/IP video stream sent from the VGPU. Telemetry streaming receives the TCP/IP data from the VGPU (on the quadcopter) and sends it on to the RC interface module. The GUI also sends commands to control the software on the VGPU over SSH. The RC interface module gets the telemetry data from the GUI, performs data calculations, and determines the correct tracking response. This response is sent over RC to control the quadcopter. The RC interface module also handles communication with its own GPS module. Table 18 discusses all of the signals sent within this software block. Both the wireless router and the base station computer are powered externally from a source within the car via power inverter. The base station computer powers the RC interface module via USB connection.

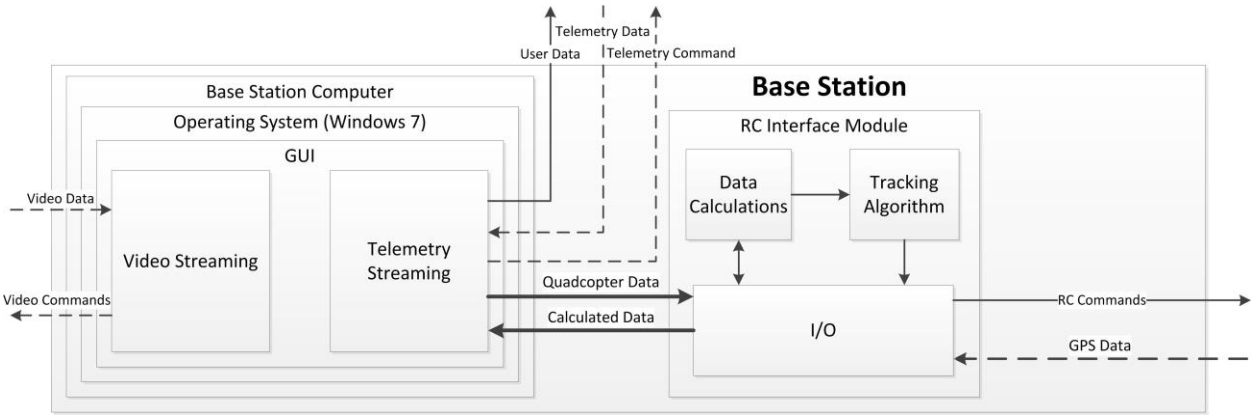


Figure 29: Base Station Software Block Diagram

Table 18: Base Station Software Block Diagram Interfaces

Signal	Description	Type
Quadcopter Data	After the GUI receives the telemetry data, it filters this information to send consistent and relevant data to the RC interface module for tracking.	String See Below (Table 19)
Calculated Data	The RC interface module compares the quadcopter data with the ground GPS data to compute the differences in location. The results of this computation is sent to the GUI via the calculated data signal.	String See Below (Table 19)

Table 19: Base Station Software Protocols

Protocol	Description
Quadcopter Data	Quadcopter data is a subset of the telemetry data. Quadcopter data begins with a single 'A' character and ends with a 'W' character. Between these characters is information about the quadcopter. This information consists of latitude, longitude, altitude, heading, time, number of satellites, roll, pitch, yaw, and throttle for the quadcopter.
Calculated Data	Calculated data is a necessary part of the tracking algorithm that is sent to the GUI. Calculated data begins with a single 'A' character and ends with a 'W' character. Information in between these is latitude difference, longitude difference, base station latitude, base station longitude, and time difference.

5.2 Base Station Components

This section discusses the design process for the base station hardware and software components.

5.2.1 Base Station Computer

The prototype 1.0 base station computer is a laptop running Windows 7 OS. This computer runs the GUI, which displays video as well as sends quadcopter telemetry information to the RC interface module. Details on the process that lead up to these decisions are in the following sections.

5.2.1.1 Hardware

5.2.1.1.1 Requirements

The base station shall be easily transportable. For prototype 1.0, the computer must be rugged enough to fall up to one meter without failures. In addition, this will help satisfy the following customer requirements: REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, REQ 2.2.1.4.A.PI, REQ 2.2.1.4.B, REQ 2.2.3.3.A, REQ 2.2.3.3.B.P1, and REQ 2.2.3.3.C.PI.

5.2.1.1.2 Alternatives

The team considered using a laptop, desktop, smart phone, tablet, single board computer, or custom computer built into the vehicle.

5.2.1.1.3 Decision Criteria

The criteria used to choose a computer for this project were its ability to support a GUI, portability, cost, and relative computing power, which is needed for the graphics processing involved in a GUI. The computer also needed to have a display large enough for the video stream to be useful.

5.2.1.1.4 Decision

The team chose to use a laptop computer owned by a team member for this project. A desktop has almost no portability making it an easy alternative to eliminate. Buying a laptop, smartphone, or tablet for the project would allow it to be more optimized for the system, but would drive the cost of the project up significantly. The team chose Jared's Gateway since he was the one primarily developing the GUI.

5.2.1.1.5 Implementation

The base station computer resides inside the vehicle and is connected to the RC interface module. The team used a 300 W BESTEK® Power Inverter to power the computer. The computer requires 90 W of power and the Wi-Fi router requires 17 W of power. Therefore, the base station components use roughly a third (107 W) of the rated power provided by the power inverter (300 W).

5.2.1.1.6 Unit Test

The GUI ran on the base station computer throughout development and performed adequately.

5.2.1.2 Software

The base station computer contains the GUI, which runs on Windows 7 OS, but is supported on Windows XP and later. In addition, the base station computer displays the video stream from the quadcopter as well as flight statistics.

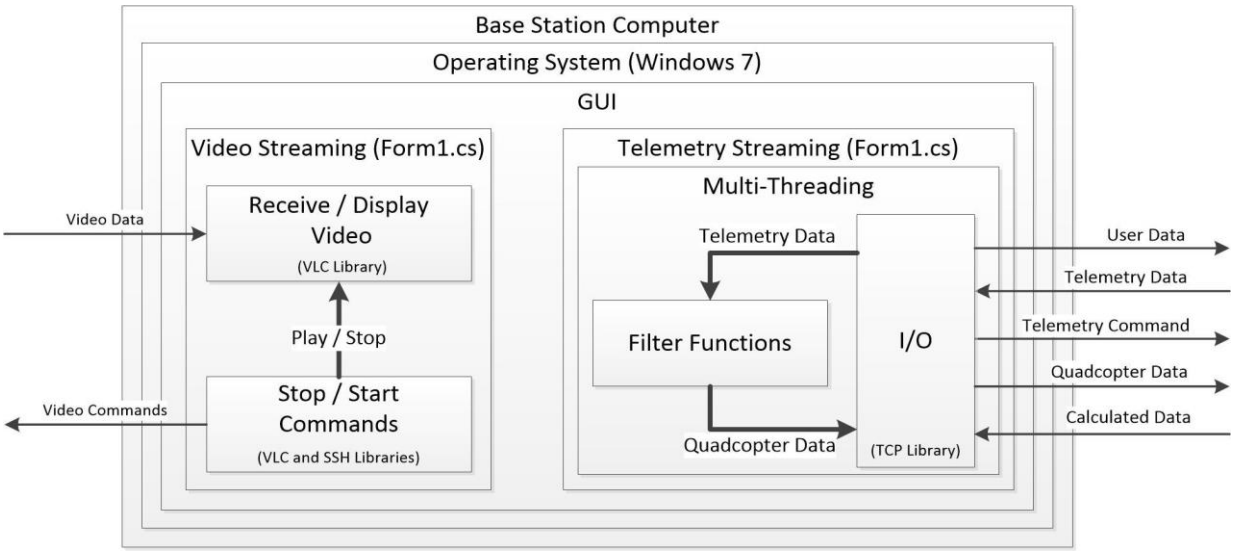


Figure 30: Base Station Computer Software Block Diagram

Table 20: Base Station Computer Software Block Diagram Interfaces

Signal	Description	Type
Play / Stop	Initializes / terminates the VLC player plugin for the GUI.	VLC Command See Below (Table *21*)
Telemetry Data	The telemetry data string contains several instances of quadcopter telemetry information. The quadcopter sends information via TCP/IP and the GUI collects all information available on the TCP/IP read buffer and stores it into the telemetry data string. This string can be several lines long.	String See Below (Table *21)*
Quadcopter Data	The filter functions parse the telemetry data and return only the most recent telemetry instance. All other telemetry instances contained in the telemetry data string are outdated at this point and are discarded. Only the most recent telemetry instance is stored to the quadcopter data string.	String See Below (Table *21*)

Table 21: GUI Software Protocols

Protocol	Description
Play / Stop	When the user clicks a button to toggle video, a play or stop command is sent to the VLC player plugin to turn on and off the video respectively. These commands are built in to the VLCPluginClass for the playlist class.
Telemetry Data	As mentioned earlier, telemetry data is a string that starts with 'A' to identify the beginning of a new data packet, contains a 'W' to signify the end of the standard data. This data consists of everything on the TCP/IP buffer at the time it is read. This telemetry data contains multiple data packets separated by these 'A' to 'W' instances.
Quadcopter Data	Quadcopter Data is the string sent from the GUI to the RC interface module. It consists of 'A', quadcopter latitude, quadcopter longitude, quadcopter altitude, quadcopter orientation, quadcopter time, and 'Z'

5.2.1.2.1 Requirements

The GUI shall display the streaming video from the quadcopter. The final production GUI shall show various statistics from the quadcopter such as speed, altitude, and battery life (recorded as flight time remaining). In addition, for the production prototype, the GUI shall provide an interface to disable autopilot and control the quadcopter manually. A final view of the GUI for prototype 1.0 is seen in Figure 31 below. In addition, the GUI will help satisfy the following customer requirements: REQ

2.2.1.4.A.PI, REQ 2.2.1.4.B, REQ 2.2.1.4.D.P1, REQ 2.2.3.1.A, REQ 2.2.3.1.B, REQ 2.2.3.3.A, REQ 2.2.3.3.B.P1, REQ 2.2.3.3.C.PI, REQ 2.2.3.3.D, REQ 2.2.3.3.E.

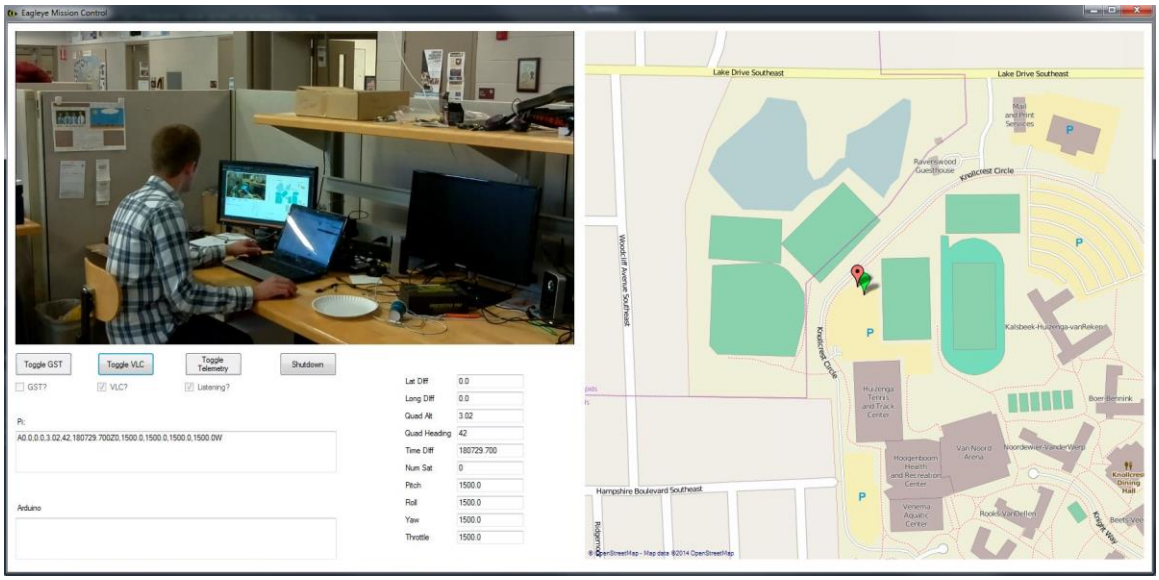


Figure 31: Final GUI

5.2.1.2.2 Operating System

5.2.1.2.2.1 Alternatives

Since the team used a laptop for the base station computer, the team considered Windows 7, Windows XP, Ubuntu, and Fedora for the base station OS.

5.2.1.2.2.2 Decision Criteria

Since the team chose a laptop with Intel's core i5 processor for the base station computer during development, the OS needed to be compatible with the Intel x86 architecture. In addition, it was useful to choose an OS for which several development environments were available.

5.2.1.2.2.3 Decision

The team decided to use Windows 7 as the OS for the base station computer. In addition, since the GUI was written in C# (see "Language" below), the team found it preferable to develop in Visual Studio running on Windows, since both are maintained by Microsoft and interact well together. The team used Windows 7 since Microsoft is no longer providing support for Windows XP. Windows would also be used in a production model.

5.2.1.2.3 Language

5.2.1.2.3.1 Alternatives

The team explored Python and C# as the languages to implement the GUI on the base station computer. These languages were the only ones investigated due to their high-level nature and availability of libraries to help implement the GUI. Specifically for Python, the team looked into pygame, gui2py, PyGUI, and Tkinter modules to create the GUI. Concerning C#, the team only looked into using Visual Studio's default Windows Forms Application to create the GUI.

5.2.1.2.3.2 Decision Criteria

Languages were evaluated based on simplicity of implementation, number of available pertinent libraries, and aesthetics.

5.2.1.2.3.3 Decision

The team decided to use C# as the language to implement the GUI. C# has several built in wrappers to quickly create a simple and powerful GUI such as System.Windows and System.Drawing. In addition, there are several open source libraries to help with networking, video streaming, and sending SSH commands such as System.Net, VLCPluginClass, GMap.NET, and Tamir.SharpSsh.

5.2.1.2.3.4 Implementation

The team created a multithreaded GUI written in C#. The GUI needed to be multithreaded in order to handle several different tasks such as listening for TCP/IP information, playing video, printing flight statistics, and sending telemetry information via serial simultaneously. C# has several helper files that are automatically generated. Team software is located in the Form1.cs and Form1.Designer.cs files of the GUI project folder located here.

5.2.1.2.4 Telemetry Streaming

5.2.1.2.4.1 Alternatives

The GUI needed to receive telemetry information via TCP/IP from the quadcopter, send this information to the RC interface module, and receive back calculated data from the RC interface module. The GUI could use interrupts or polling to receive information from the quadcopter and RC interface module.

5.2.1.2.4.2 Decision criteria

The team evaluated the method of gathering telemetry information based on simplicity of code structure and the code execution time.

5.2.1.2.4.3 Decision

The team used a polling technique to gather telemetry data from the quadcopter and calculated data from the RC interface module. Polling was deemed as a simple and efficient technique to use. Therefore, the team decided to use a continuously polling loop to retrieve telemetry and send it to the RC interface module.

5.2.1.2.4.4 Implementation

In the GUI, there is a listening function (listen()) that performs the telemetry streaming. First, this function establishes a TCP/IP connection with the quadcopter. Next, it enters a loop. In this loop, the GUI gathers all information on the TCP/IP buffer, filters this information, sends it to the RC interface module over serial, reads all calculated data on the serial receive buffer, displays flight statistics to the computer screen, and repeats the loop. The amount of time it takes to repeat this loop affects the communication latencies. Details on telemetry latency requirements are found in section 6.1.3. The filtering described earlier in Table 18 to ensure the RC interface module receives consistent data. The team established a communication protocol to identify individual messages on a communication buffer (TCP/IP or serial). The team sent a single ‘A’ character before a set of information and a single ‘W’ after the message. This allowed the team to read an entire buffer and filter the message by only passing along the most recent ‘A’ to ‘W’ instance. The user can toggle this functionality by clicking the “toggle telemetry” button. Note: the telemetry communication is completely independent of the video streaming.

However, if telemetry communication is terminated, the RC interface module will recognize it is no longer receiving telemetry information and will return manual control to the pilot. The telemetry information must be sent to the RC interface module as quickly as possible in order to minimize latencies and increase the tracking robustness. Therefore, it is preferable to have a dedicated thread to perform this function. Using Visual Studio's performance analyzer the team discovered the GUI used 50% CPU utilization during telemetry and video streaming. This means additional processes can run on the computer. The number of additional processes, however, depends on their CPU demand. C# has two main libraries for multithreading: a thread class and a backgroundworker class. The team chose to use the backgroundworker class because it is easier to interrupt the process running on the backgroundworker thread. In addition, when the thread is interrupted, it raises a flag signaling it was successfully canceled. Using this functionality, the team can easily stop the listening function and reset variables once the thread stops listening. The team initialized a backgroundworker instance to take care of this function.

5.2.1.2.4.5 Unit Test

Initially, pop up boxes were used to confirm the GUI could receive information over TCP/IP. Next, messages were sent over serial to the RC interface module from the GUI. Once these pieces were in place, a message received over TCP/IP was sent via serial and viewed using a serial monitoring program (PuTTY). These steps confirmed the message was successfully received and passed through the GUI using this technique.

Once communication was confirmed, an initial communication system test seen in section A.2.1 of the Appendix showed latency once connection was established. The team discovered TCP/IP and serial reads only consumed one line of information, but several lines existed on the respective read buffers. This indicated that the GUI communication was the bottleneck of the system. Therefore, the most recent telemetry information was not always being used. To fix this, the team flushed read buffers once communication was established to remove erroneous information and the team updated read commands to consume all information on the read buffer rather than a single line. In doing so, communication latencies did not increase since the most recent message was passed along. The only trade-off to this technique is that the GUI limits the telemetry information refresh frequency. However, the GUI refreshed information multiple times a second and did not inhibit the system from meeting its tracking accuracy requirement. Since all the information available on a buffer was consumed, the GUI needed to filter all but one line of telemetry information (the most recent line) for the RC interface module. These techniques ensured the RC interface module was receiving the most current information.

5.2.1.2.5 Video Streaming

5.2.1.2.5.1 Alternatives

The team decided between viewing the video through a window in the GUI and creating a separate pop up window to view the video. In addition, the team needed to decide how the user would start and stop the video if desired. The video could be initialized and terminated on startup and power down or via SSH commands performed behind the scenes.

5.2.1.2.5.2 Decision criteria

The decision of where and how to playback the video was based on simplicity for user interaction. In addition, the team had to balance the tradeoff of making things simple for the user and giving the user full control when it comes to initializing video.

5.2.1.2.5.3 Decision

The team decided to integrate the video into the main GUI window for user simplicity. This minimizes the number of windows the user must keep track of. In addition, the team decided to give the user more control and enable the user to turn video on and off with buttons in the GUI.

5.2.1.2.5.4 Implementation

As noted above, the team decided to give the user more control and allow the user to turn the video on and off. This was a tradeoff balancing user control with simplicity. Since the team gave the user more control, we wanted to make this control procedure as simple as possible. Therefore, total video control is integrated into one button - "toggle video". This button sends an SSH command to the quadcopter to start or stop the camera on the quadcopter, the associated processes, and the integrated video screen in the GUI for user convenience. Upon termination, the quadcopter camera will turn off and the video will display a black screen.

Video taking UAVs have been the point of controversy lately. Therefore, our project presents possible concerns with privacy, so the GUI must be clear on how the video is being used. Moreover, the team took extra care to be transparent about the intention as well as capabilities of their product.

5.2.1.2.5.5 Unit Test

Video streaming software unit tests were performed in conjunction with the system integration test for video streaming (see section 6.2).

5.2.1.2.6 Maps

5.2.1.2.6.1 Alternatives

The team desired to implement a map into the GUI to increase the amount of information available to the user. The team looked into several APIs (Application programming interfaces) for C# to integrate a map into the GUI. Some of these options were SharpMap, Googlemaps, DotSpatial, and GMap.NET.

5.2.1.2.6.2 Decision criteria

The decision of where and how to view the map information was based on simplicity for user interaction, ability to integrate the map in the main GUI window without opening a web browser, ability to use the map without internet access, and ability to plot points on the map via GPS location.

5.2.1.2.6.3 Decision

The team decided to use GMap.NET since it could easily integrate into the main GUI window and could plot several points via GPS coordinates. In addition, cached map data allows users to view the map without internet access. The only downside is that users must cache map information while internet access is available before removing internet connection.

5.2.1.2.6.4 Implementation

The team integrated GMap.NET into the GUI and added markers for both the quadcopter and the base station on the map. Both quadcopter and base station passed their GPS coordinates when communicating telemetry information. The respective GPS information is passed to a function that updates the map (UpdateMarker()). It was required to make this function a delegate method since the thread gathering telemetry information (via listen()) did not have access to the map since it did not create the map. Therefore, the thread that gathered telemetry information passed the GPS coordinates along to the delegate methods to take care of updating the map asynchronously. In addition to GPS coordinates a

Boolean argument, isQuad, is also passed. This Boolean determined what color the marker would be. Both quadcopter and base station used different marker colors to avoid confusion.

5.2.1.2.6.5 Unit Test

Once both quadcopter and base station GPS coordinates were reported to the map, the team moved each subsystem to ensure the map updated appropriately.

5.2.2 RC Interface Module

An RC interface module was needed to compare the GPS data and control the quadcopter actions through an RC transmitter. This module translates the telemetry data being sent from the base station computer's USB port into the RC controls used by the quadcopter. Figure 32 shows the block diagram of the RC interface module.

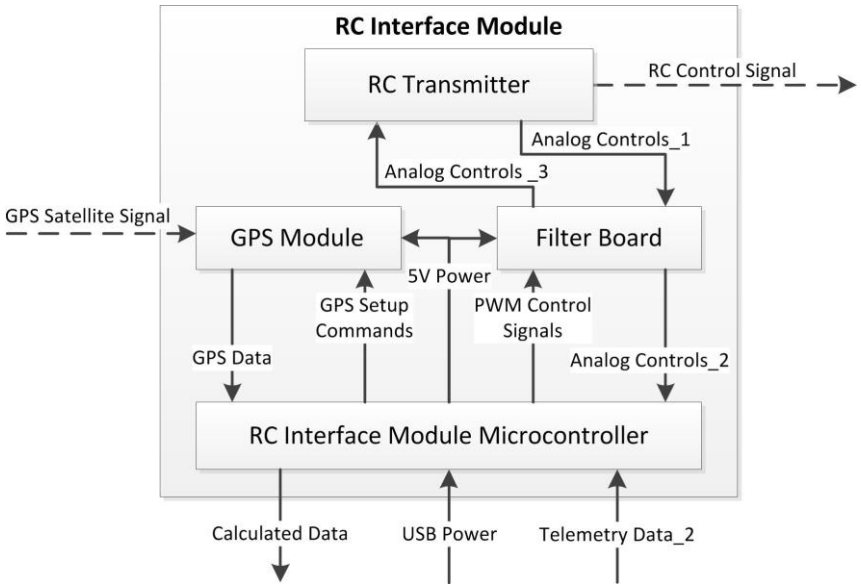


Figure 32: RC Interface Module Hardware Block Diagram

Table 22: RC Interface Module Hardware Block Diagram Interfaces

Signal	Description	Type
Analog Controls_1	Analog controls being output by the potentiometers of the RC transmitter. Used for manual controls.	0-5 V DC
Analog Controls_2	Analog controls after being voltage divided by filter board. Used for reading manual controls.	0-2.5 V DC
Analog Controls_3	Analog controls sent back to the RC transmitter to control quadcopter actions.	0-5 V DC
5 V Power	Provides power to the Adafruit GPS module from microcontroller.	5 V DC
GPS Setup	GPS setup is a signal sent upon initialization via digital I/O pins. This information sets the correct settings on the Adafruit GPS	PMTK NMEA 0183

Commands	module such as baud rate (115,200) and refresh rate (10 Hz).	String[26]
PWM Control Signal	A PWM signal output by the microcontroller that is later converted to RC controls for quadcopter	0-3.3 V PWM
GPS Data	The Adafruit GPS sends GPS data to the RC interface module including longitude and latitude.	NMEA 0183 String[26]

5.2.2.1 Hardware

5.2.2.1.1 Microcontroller

Part of the RC interface module is a microcontroller capable of running flight software and driving I/O pins. This microcontroller takes in various pieces of telemetry data as well as potentiometer values from the RC transmitter and analyzes the appropriate action based on the tracking algorithm.

5.2.2.1.1.1 Requirements

The microcontroller shall send tracking controls to the RC transmitter and receive telemetry data from the base station computer over a serial port. This portion also helps satisfy the following customer requirements: REQ 3.2.3.2.A, REQ 3.2.3.3.C, and REQ 3.2.3.3.C.P1.

5.2.2.1.1.2 Alternatives

The team considered the Raspberry Pi Model B, Arduino Uno, Arduino Due, Arduino Mega, and BeagleBone Black as viable options for the RC interface module microcontroller. For a final product, a board with the exact capabilities needed would likely be custom made to minimize cost. Table 23 shows a list of the boards the team looked at and their hardware specifications. Note: this board is required to be compatible with an Adafruit GPS module; however, all alternatives met this requirement.

Table 23: RC Interface Module Development Board Alternatives

	Raspberry Pi B[56]	**Arduino Uno**[57]	**Arduino Due**[57]	**Arduino Mega**[57]	**BeagleBone Black**[58]
Price	$40	$22	$40	$40	$45
I/O (Pins)	20	20	66	70	92
Adafruit Compatible	Yes	Yes	Yes	Yes	Yes
Pin Output	50 mA at 3.3 V or 5 V	50 mA at 3.3 V	800 mA at 3.3 V or 5 V	50 mA at 3.3 V	3.3 V, 5 V
USB Type	2 ports A	B	Micro B	B	A

5.2.2.1.1.3 Decision Criteria

Price was a factor, as well as the number and capabilities of I/O pins to ensure that all communication could be received from the laptop and sent to the RC transmitter. This includes serial ports and pulse width modulation (PWM) analog outputs. Although not a requirement, it would be ideal if developers of the flight controller and the RC interface module could use the same communication cable (USB micro Type B) for fast interchangeability with computers used for programming. The board

needed to have a 5 V output pin in order to power the Adafruit GPS module. This is not shown in the above table because all the boards met this requirement. The compatibility with the selected GPS module had to be considered, as the GPS module needed to be connected to the RC interface module. The power consumption was not considered for this section, as all boards were reasonably small (less than 5 watts).

5.2.2.1.1.4 Prototype Decision

The team chose to use the Arduino Due as the RC interface module microcontroller. It provided the required amount of I/O pins with room for expandability, as well as the serial ports needed for communication with the GPS module and base station computer. It was chosen over the BeagleBone Black because of the team's familiarity with Arduino boards. The Arduino Due also used a USB micro Type B for communication with the base station computer, making it a more convenient option for development.

5.2.2.1.1.5 Production Decision

A custom board option was considered for a final product, as it could be produced for less cost than the other options when produced in volume. By eliminating unnecessary components, the custom board could be approximately 5.6 x 4.8 cm resulting in an area of 27 cm^2. The existing Arduino has an area of 52 cm^2, making it a reduction in size of 50%. The layout with basic components and costs is shown in Figure 33 below. With an approximate fabrication cost of $10, the total cost comes to $23, making it the most inexpensive and spatially efficient option for very large volumes.

Roughly estimating 500 hours for design work, at $100 dollars per hour, this would amount to design costs of $50,000. Since the team estimates only 600 sales in the first three years, design costs would not be amortized significantly enough to warrant this custom board since the savings would only be $(\$40 - \$23) * 600 = \$10{,}200$. The break-even cost for this custom board would have to be $50,000 in savings. Therefore, the Arduino Due would be used for full production.

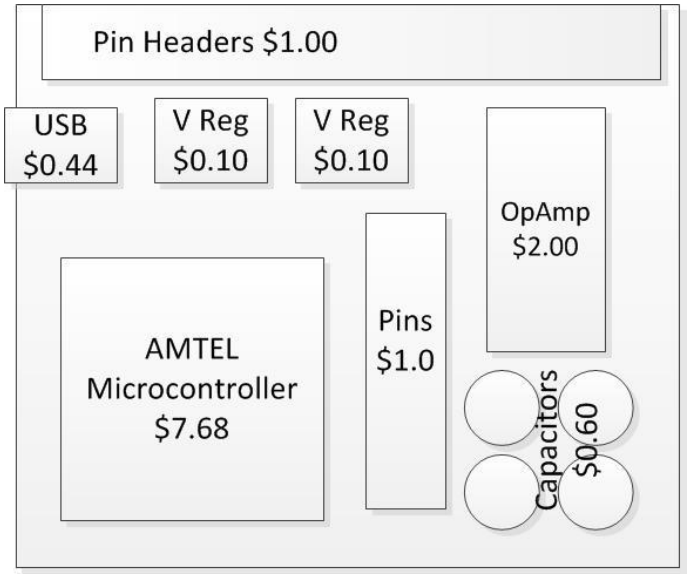


Figure 33: Custom RC Interface Module Layout

5.2.2.1.1.6 Implementation

The RC interface module microcontroller is implemented per the RC interface module architecture block diagram (Figure 28 and Figure 32) and the accompanying table (Table 17 and Table 22).

5.2.2.1.1.7 Unit Test

The RC interface module microcontroller was tested to see that it could perform its various tasks. The GPS module sent serial data to the microcontroller, which parsed data and sent the results back to the base station computer. Next, the microcontroller was connected to the filter board (see section 5.2.2.1.2) to verify it could read multiple analog signals and write controls simultaneously to the PWM pins. This test was run with the quadcopter to see the motors running when the RC interface module was sending commands.

5.2.2.1.2 Filter Board

Since the chosen microcontroller development board, and all the other possible options output PWM signals as their analog outputs, a custom board was needed to provide signals to the RC transmitter. In addition to filtering, the signals coming from the RC transmitter range from 0-5 V, while the microcontroller can only read and write from 0-3.3 V. These issues brought the need to step down and up the signals before and after the microcontroller respectively. This was initially not an issue, as the first RC transmitters used only used a 0-3.3 V scale.

5.2.2.1.2.1 Requirements

The filter board shall be capable of converting a 0-5 V analog signal from the RC transmitter to a signal on a 0-3.3 V scale for the RC interface module microcontroller. It must also be capable of stepping up a 0-3.3 V signal to a 0-5 V signal linearly to go the opposite direction. The solution shall be easily powered inside the base station, requiring no more than 5 V and a small battery in series. The board shall also convert PWM signals to pure analog signals with no delay or jitter viewable in the RC commands as displayed in the MW GUI. The stepped down signal shall be capable of charging the analog read capacitors in the microcontroller quickly enough to allow for the use of four channels to control the quadcopter.

5.2.2.1.2.2 Alternatives

Alternatives for this board included using operational amplifiers for the step up and step down functions, using a voltage divider for the step down function and an operational amplifier for the step up, or purchasing logic converters made for similar purposes that were presumed to have a linear output. A low pass filter with a cutoff frequency of less than 10 Hz was needed to convert a PWM signal to analog. 10 Hz was chosen because it is well below the frequency of the Arduino PWM output of 500 Hz. The two options for the operational amplifier used to step up would be to use a more expensive operational amplifier with a larger output swing with a given voltage supply, or place a small battery in series with the power supply, giving a lower range operational amplifier the supply voltage needed to output a swing of 0-5 V. The reason a separate power supply was not considered was that the filter board was to be powered by the microcontroller, which is capable of up to 5 V. The team also decided on whether to make the board in house or to send it out for fabrication.

5.2.2.1.2.3 Decision Criteria

The reliability of the components was the most important aspect of the filter board as any glitch in the filter could cause damage to the RC interface microcontroller, or send a faulty command to the quadcopter causing a crash. This means that the voltage inputs and outputs should be constant with time

(meaning a lowered voltage due to battery drain was undesirable), and the final software should work without constant adjustment and rewriting of code because of this lower voltage. Power consumption and simplicity were also strong decision points. Resistor and capacitor component values were also chosen based on their availability.

5.2.2.1.2.4 Decision

Initially the decision was made to purchase the step up and step down circuit (BOB-12009) from Spark Fun due to the ease of implementation and a relatively small cost ($3). The team purchased the parts with limited information on the uses, and once implemented into the RC interface module they proved ineffective. When the team connected the RC interface module to the MW GUI to view the RC values being sent to the quadcopter, the response of the signal was not linear. The details of this non-linearity were not explored, as it was obvious the part could not be used for this purpose, and the team wanted to move forward with a solution. It was determined from this test that the logic converter would not work for analog signals and would only work for high or low signals. While this behavior is defined by the part name (logic level converter), the team had hoped it used linear circuitry like an operational amplifier to do conversions. The team had to choose a new option to step up the voltage outputs. The logic converter was sufficient, however, for both of the auxiliary channels, as they are only high or low. Therefore, the team implemented them into the RC interface module for these channels.

The team then decided to use a voltage divider for the step down circuit in order to get a readable voltage for the microcontroller. While this solution wastes power, the team decided to accept this loss in order to save on design time. We chose 51 kΩ for the step down resistor values as these were commonly available and fulfilled the requirements. The circuit for this voltage divider is shown in the filter board schematic (Figure 34). With the chosen resistor values, the power wasted was at most 0.12 mW (by using$P = V^2/R$, where R = 51 kΩ and V = 5 V). With 2500 mAh batteries, it would take 212 minutes assuming four potentiometers at 5 V to exhaust the batteries in the transmitter.

Choosing small resistors caused the input voltages to drop since the RC transmitter potentiometers providing the voltage were not capable of supplying enough current. If the resistor values were chosen to be higher, it would cause a ghosting condition to occur on the analog read functions of the microcontroller. This is caused by the input terminal not receiving enough current. Fifty-one kΩ was the only resistor value the team had access to that showed neither signs of ghosting or current leakage.

For the step up functionality of the filter board, an LMC6484IN rail-to-rail operational amplifier was chosen. This was chosen over another op amp, such as the LM224n, because it did not need a battery in series with its power supply to achieve the required voltage swing. The resistors used for the amplifier were chosen to be 1.3 kΩ and 620 Ω in order to achieve a gain of roughly three. The team chose a gain of three to ensure that the whole range of output was utilized for the op amp (0-5 V). These resistors were chosen because the output swing required was dependent on the resistors, and anything higher than 1.3 kΩ would decrease the output swing of the operational amplifier. By using the logic converter for the auxiliary switches and the operational amplifier for the four directional channels the team only needed one operational amplifier integrated circuit.

The resistor value for the low pass filter was chosen to be 620 Ω as described above. The capacitor was chosen to be 100 µF because it showed no delay or jitter that could be viewed in the MW GUI. This was found by testing multiple capacitors ranging from 10 µF to 5800 µF. Choosing a higher capacitor value led to a longer time to charge the capacitor and a delay in the output. Choosing a smaller capacitor yielded more jitter in the final signal, which would result in unstable flights. A final circuit of the filter board is shown in Figure 34. The cutoff frequency of the low pass filter was about 3 Hz, well below the requirement of 10 Hz.

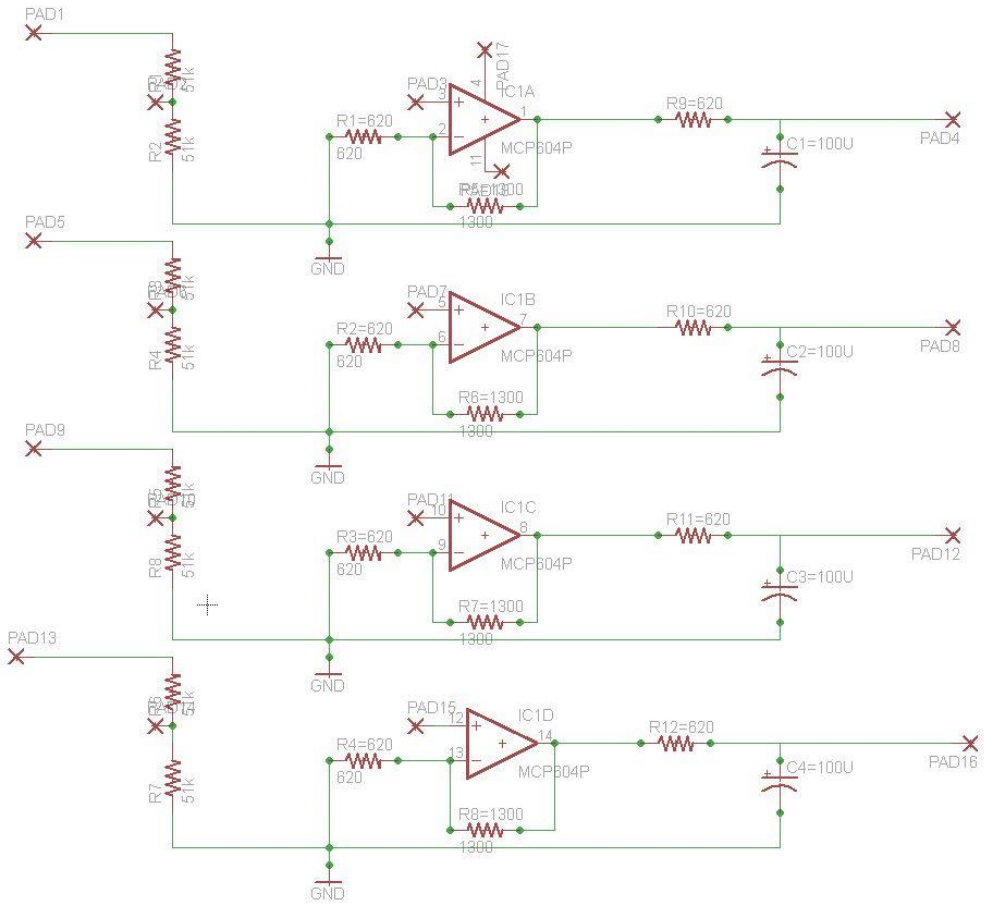


Figure 34: Filter Board Circuit

Once the team decided on all the components for the circuit, the team chose to fabricate the board in-house to minimize lead-time and cost. A diagram showing the entire process of reading controller potentiometers from the controller to writing RC controls is shown in Figure 35 below. Figure 36 shows the layout of the board and Figure 37 shows a photo of the actual board.

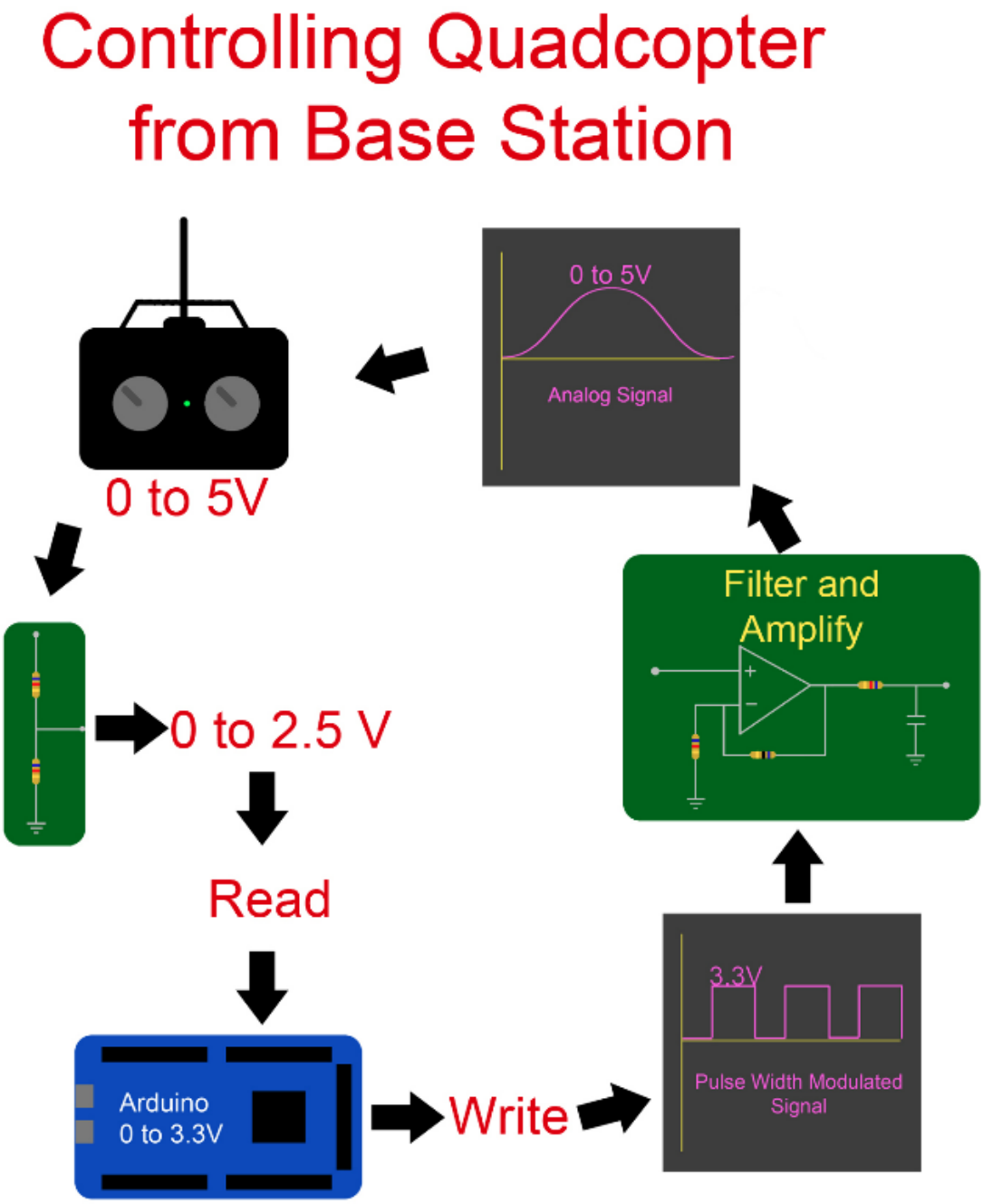


Figure 35: RC Interface Module Output Diagram

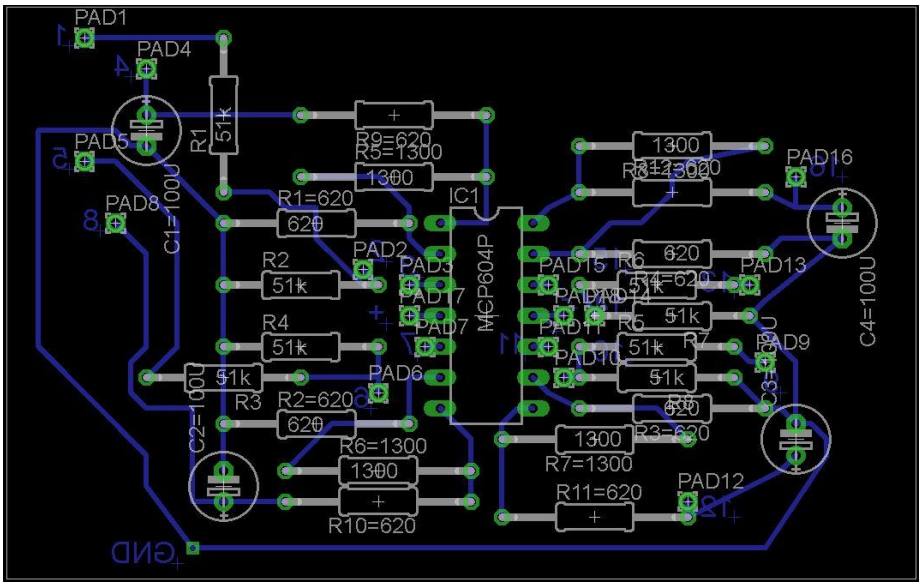


Figure 36: Filter Board Layout

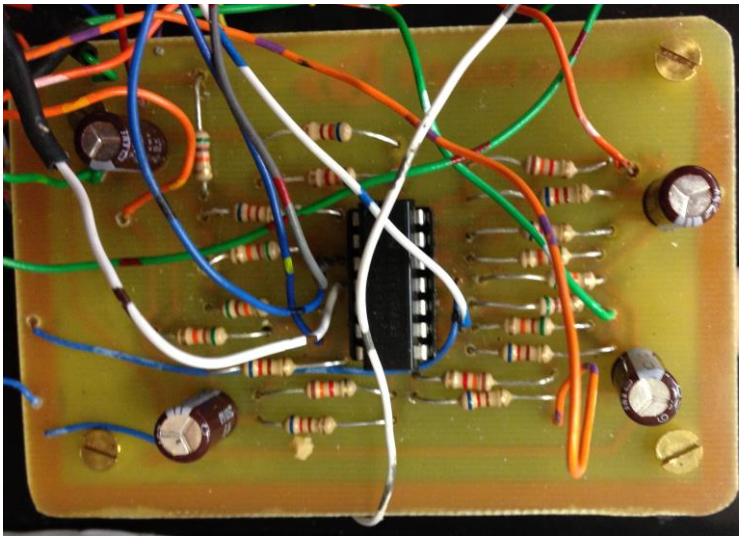


Figure 37: Filter Board Photo

5.2.2.1.2.5 Implementation

The team soldered the final components to an in-house fabricated board and used them with the microcontroller. This design allowed for pass through functions to be written where the base station is controlling the quadcopter but is getting the controls from the RC transmitter's potentiometer sticks. The voltage signal from the RC transmitter is first sent through a voltage divider, and then read by the microcontroller analog input to an integer between 0 and 900, which is the full range of the input after the voltage divider. Since the standard command to write to a PWM port on the microcontroller uses an 8-bit value (0-255), the float is converted to a 0-255 scale and is written back to a PWM pin. For the tracking algorithm, the PWM output values are calculated based on the data received by the GPS module and the base station computer. More details on this conversion from GPS coordinates to PWM are discussed in section 5.2.2.2.

5.2.2.1.2.6 Unit Test

The team wrote a test program to read potentiometer values from the RC transmitter and print those vales on the 0-900 scale as well as the 0-255 scale to a serial monitor. As potentiometers moved up and down, or left and right, the values on both scales corresponded to those movements, validating the design. This pass through test would later be used for manual flights, but is not part of the final implementation of the board.

5.2.2.2 Software

Autopilot code is executed in the base station and controls the quadcopter without the need for human interaction. A block diagram showing the flow of the RC interface module software is shown in Figure 38 with accompanying descriptions in Table 24. For the production prototype, obstacle avoidance needs to be included in the software, but will not be implemented or discussed further in this project.

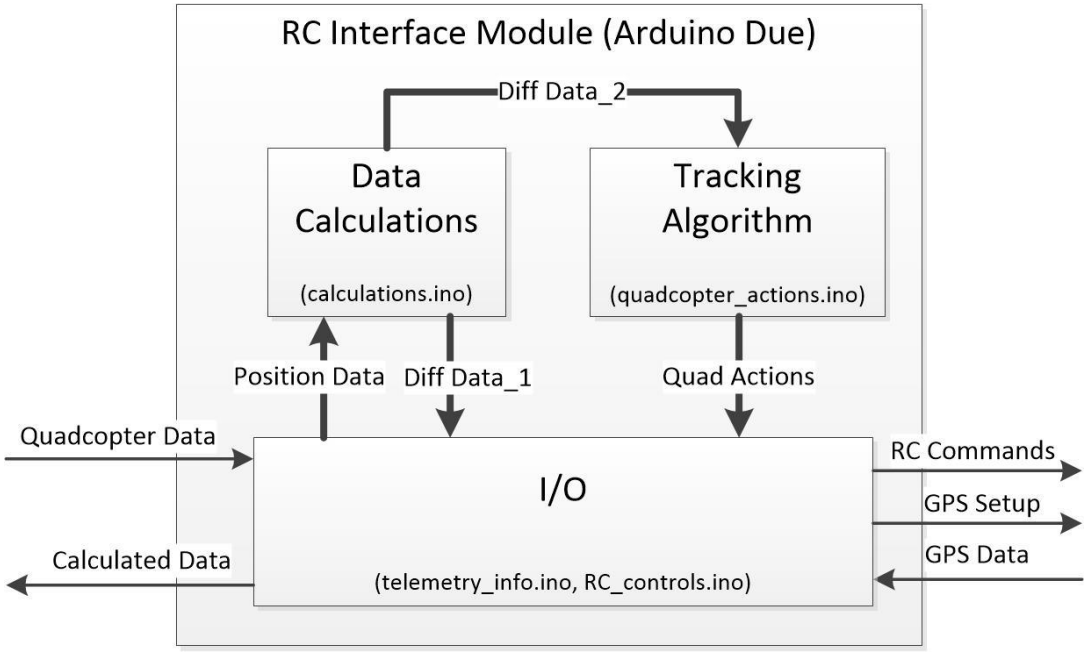


Figure 38: RC Interface Module Software Block Diagram

Table 24: RC Interface Module Software Block Diagram Interface Descriptions

Signal	Description	Type
Position Data	Parsed car GPS and quadcopter GPS data in addition to quadcopter heading.	NMEA float +/- 3 m
Diff Data_1	Differences in latitude/longitude and orientation calculated and output for the GUI.	Float
Diff Data_2	Differences calculated and used in the tracking algorithm to control the quadcopter.	Float
Quad Actions	A set of variables (throttle, yaw, pitch, and roll) that represent an analog write signal going to the filter board through PWM pins.	Float from 0-255

5.2.2.2.1 Requirements

Controls shall be given to the quadcopter in order to fly it to within 7.2 meters of the x, y position of the vehicle and a variable altitude above the vehicle. Figure 2 in REQ 2.2.1.2.D.P1 demonstrates this requirement. Control methods shall also allow for manual flight mode, while still allowing for programmed automatic takeoff and landing functions. In addition, the tracking algorithm will help satisfy the following customer requirements: REQ 2.2.1.3.F.PI, REQ 2.2.3.1.A, REQ 2.2.3.1.B, REQ 2.2.3.2.A, REQ 2.2.3.3.C.PI, REQ 2.2.3.3.D.

5.2.2.2.2 Alternatives

The team examined two ways to provide navigation for the quadcopter autonomously: by sending GPS coordinates to the base station or, by sending GPS coordinates to the quadcopter.

The first way was to have the quadcopter send its GPS coordinates from the flight controller or VGPU to the base station computer. The base station would then compare its own GPS position to that of the quadcopter and calculate the appropriate maneuvers, sent back to the quadcopter through a RC transmitter. This technique would allow for easier manual control of the quadcopter as well as the option of programming automatic landing and takeoff features. In this option, the flight control software is still used for quadcopter stability; however, it is not used for tracking and positioning.

The second way of controlling is to send the vehicle's GPS location to the quadcopter and then allow on-board software to compute the next flight path. While this option would be a logical step in the direction of obstacle avoidance, it would require more work to implement manual control for debugging purposes. This method would require dynamic modification of the waypoint software on the flight controller. Since this feature is not included in the flight control software, it must be done directly by the team.

5.2.2.2.3 Decision Criteria

The criteria for this decision were the bandwidth necessary for each option, the effectiveness of the product, and ability to implement manual flight controls for debugging. Calculated values for bandwidth requirements to stream video are shown in section 4.2.9.2.1 and show 1.8 Mbps for this project, and up to 35 Mbps for a production version (using H.264 compression). The bandwidth to send the telemetry data is 5.6 Kbps ($70\ chars * 8\frac{bits}{char} * 10\ Hz = 5.6\ Kbps$) which makes the total bandwidth

for this project less than 2 Mbps and 36 Mbps for a production version. The first option requires a slightly greater amount of data being sent back and forth through Wi-Fi since it will be sending about twice as much telemetry data; however, it is quite small (less than 0.05%) when compared to common router bandwidths in excess of 100 Mbps, so it is not a significant criterion for this decision. Manual flight control override is also a necessary feature.

5.2.2.2.4 Decision

The team decided to implement the option of navigation computation on the base station (thus sending RC signals to the quadcopter). This allowed for manual flight modes and the possibility of automatic takeoff and landing programs as well. Since the two options proposed were not mutually exclusive, a final product could implement the other option as a source of added accuracy and redundancy. Note: although the other option would make it easier to add obstacle avoidance, it was not chosen because the team knew obstacle avoidance was beyond the scope of this project.

5.2.2.2.5 Implementation

The team implemented a way of tracking that used telemetry sent down from the quadcopter. More details on the tests of this algorithm are discussed in section 6.5.

5.2.2.3 Tracking Algorithm

Once the team chose a means of directional controlling, the team chose a following algorithm. The following algorithm is the method the code uses in order to track the vehicle.

5.2.2.3.1 Requirements

The tracking algorithm shall be capable of tracking a stationary or moving base station within 7.2 meters. In addition, the tracking algorithm will help satisfy the following customer requirements: REQ 2.2.1.3.F.PI, REQ 2.2.3.1.A, REQ 2.2.3.1.B, REQ 2.2.3.2.A, REQ 2.2.3.3.C.PI, REQ 2.2.3.3.D.

5.2.2.3.2 Alternatives

The first following method considered was to use incremental speed responses based on the distance from the vehicle. In this method, the quadcopter would move at its fastest speed when it is farthest away from the base station and slowest when it is closest. Figure 39 shows a representation of these speeds. In the figure, the red area corresponds to the highest speed, while the green area represents the slowest speed.

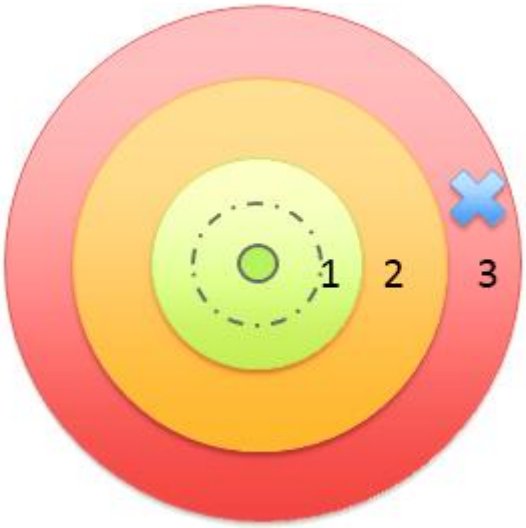


Figure 39: Distance Dependent Speed Method

The second option considered was to accelerate toward the base station, causing the quadcopter to move fastest as it entered the 7.2 meter radius. For a stationary base station, this method requires a way of stopping the quadcopter without it becoming unstable, but it allows for better tracking of a moving vehicle than the first option. For a moving base station, this method would call a velocity match function once inside the radius. Testing information for the stop function is discussed section 6.4.5.

5.2.2.3.3 Decision Criterion

The major decision criterion was the anticipated effectiveness of the tracking algorithms. Since the team knew it would likely not be able to test both options, they chose that which showed a promise of being effective for a stationary *and* moving base station.

5.2.2.3.4 Decision

For the tracking algorithm, the team decided that the quadcopter would use the acceleration method to follow the vehicle. This gave the quadcopter the ability to catch up to a moving vehicle, as opposed to trailing the vehicle, like the incremental speed option.

5.2.2.3.5 Implementation

Incremental development techniques were especially important in this portion of the project. Initially an algorithm was written to only match the GPS position of the car with the quadcopter. With this implementation, the quadcopter would always face north, pitching to manage its north/south position, and rolling to match its east/west position. This implementation was at a disadvantage because the quadcopter camera was often not facing the vehicle it was following. This method was chosen for the first implementation due to the simplicity to create a basic tracking program. In order to minimize safety concerns during testing, the GPS position was modified in the software to have the quadcopter fly to a position ten meters north of the base station as multiple team members were standing at the base station. See section 6.5.3 for test results of this tracking method.

The second and final implementation used the orientation data from the quadcopter flight controller to always face the direction of travel. This is in effect the direction of the vehicle with respect to the quadcopter. If the quadcopter would overshoot the base station, the tracking algorithm continually turns to face the base station based on its position. This program first calculates the angle of travel based on the latitude and longitude differences, and yaws to point in the direction of travel. A diagram of this is

shown in Figure 40 below. Once the angle of the quadcopter is within ten degrees of the angle of travel, the quadcopter starts to pitch forward continually up to 10% of maximum pitch, creating a smooth acceleration in the direction of the vehicle. Ten degrees was chosen because values smaller were tested and resulted in a lot of stopping as the compass of the quadcopter is not precise enough to hold orientation within anything less than ten degrees. This implementation allows for the quadcopter to always have the vehicle in the camera's view. See section 6.5.4 for tests involving this orientation based tracking. Once within three meters of the base station, the quadcopter stop function is called. Three meters was chosen for this because it was the smallest distance that the quadcopter would not actively fly away from the base station because of the accuracies of the GPS modules.

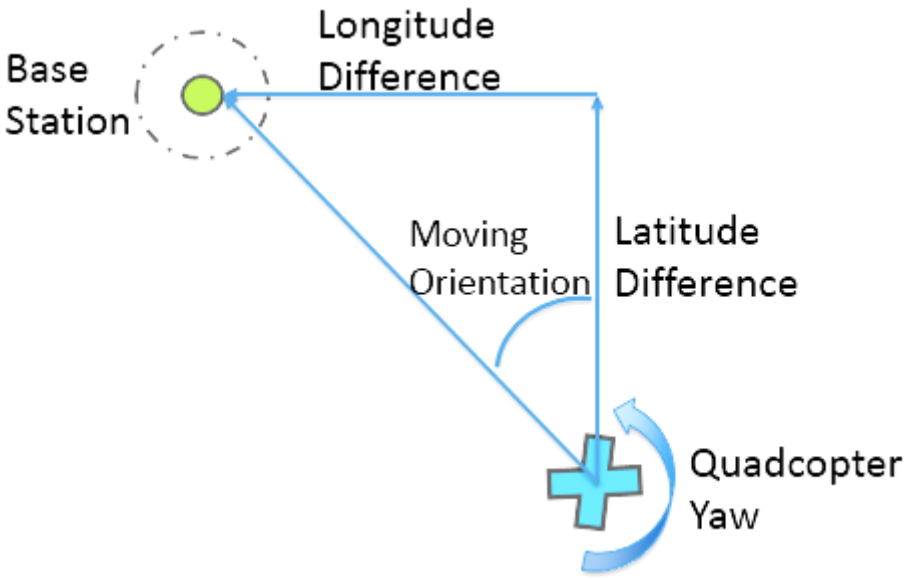


Figure 40: Orientation Tracking

The team chose these two implementations as a set of iterative steps instead of mutually exclusive options. Most of the functions used in the initial tracking algorithm could also be used for the final implementation with only small modifications.

Once the following program recognizes that the quadcopter needs to move in a direction, a value of ten percent greater than center voltage of the potentiometer is written to the pitch channel of the RC transmitter, causing the quadcopter to fly forward. This ten percent value was chosen by test 6.4.1.5. Since a constant pitch or roll in the quadcopter results in an acceleration, the change in tilt results in a jerk - the derivative of acceleration. To ensure that the quadcopter does not jerk at an unstable rate, the flight control software slowly increments the amount the quadcopter pitches forward ending in a ten percent increase. Because the risk of crashing was constantly present, the team did not run quantitative tests on the maximum safe jerk. The process of reaching a maximum acceleration takes five seconds, but a user of the code has the ability to quickly change how fast this jerk is. The team chose this jerk because it showed good results in test 0. Note: Any changes would need testing similar to those conducted for the existing jerk.

Once the quadcopter reaches the 7.2 meter radius of the car, it then matches the velocity of the vehicle by jerking in the same manner. If the base station is stopped or has a velocity less than 1 m/s, the quadcopter is stopped using a recursive stop function. One meter per second was the maximum speed the

team saw inherent in the GPS data when the base station was stationary. The stop function reads the amount the quadcopter is pitching forward and reverses the direction of tilt for 0.6 seconds per test 0.

By changing the acceleration toward the base station instead of moving at a constant rate, the follow program gives the quadcopter the ability to catch up and match the velocity of the vehicle. Increasing the speed of the quadcopter as the distance from the base station increases never allows for the quadcopter to catch up. This is because as the quadcopter starts to speed up, it gets closer to the base station, thus causing it to slow down again. This effect could be mitigated for some speeds based on the speed zones used, but there would always be a point where the problem arises. The chosen algorithm moves the quadcopter faster as the time it has been outside of the 7.2 meter radius increases.

5.2.2.3.6 Unit Test

In order to verify the software was working, various steps were transmitted through the serial port showing calculations and actions. Initial tests were run when the follow program was given artificial GPS data and the differences in position were printed through the serial port. Next, the RC values were printed at every iteration of the software loop and artificial GPS data was given in each direction, giving the ability to check the program's actions for any possibility of flight positions.

Since testing of the effectiveness of the tracking algorithm in a real situation requires the use of the whole system, further tests are discussed in Chapter 6.

5.2.3 RC Transmitter

The RC transmitter is used to send controls to the flight controller on the quadcopter. It is also used as a means for manual flight.

5.2.3.1 Hardware

5.2.3.1.1 Requirements

The RC transmitter shall be compatible with the RC receiver on the quadcopter or come with a receiver that can be easily implemented on the quadcopter. It shall have the correct control features of throttle, pitch, yaw, and roll to control the quadcopter. In addition, the RC transmitter will help satisfy the following customer requirements: REQ 2.2.1.2.E.P1, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, and REQ 2.2.1.3.C.P1.

5.2.3.1.2 Alternatives

An RC transmitter module and an RC receiver are used to send RC communication to the quadcopter. The first option explored and tested was the Turnigy 6x transmitter. The second option considered was the FlySky FS-CT6B.

5.2.3.1.3 Decision Criteria

Cost, lead-time, and ease of modification were considered in deciding which RC transmitter to use.

5.2.3.1.4 Decision

A Turnigy 6X transmitter was initially used due to its immediate availability to the team. However, after two instances of unsolvable defects, the team chose to reevaluate that decision. The Turnigy 6X transmitter costs $30 and has a lead-time of about two to four weeks while the FlySky transmitter costs $40 but has a lead-time of five to seven days. The lead-time was more important than cost at the time of this decision, as well as the fear of another Turnigy transmitter being defective. Because of this, the team chose the FlySky transmitter.

5.2.3.1.5 Implementation

The team incorporated the new transmitter's receiver into the quadcopter and the transmitter was connected to the RC interface module (see section 5.1 for details).

5.2.3.1.6 Unit Test

The team tested the RC transmitter by viewing the RC values received by the quadcopter using the MultiWii GUI.

5.2.4 GPS Module

The base station GPS module is used to determine the geographical location of the base station.

5.2.4.1 Hardware

5.2.4.1.1 Requirements

The GPS module shall have the accuracy necessary to track the vehicle's position within three meters. In addition, the GPS module will help satisfy the following customer requirements: REQ 2.2.1.2.C.P1, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, and REQ 2.2.1.3.D.P1.

5.2.4.1.2 Alternatives

GPS modules come with either a USB interface, or digital I/O pins to transmit and receive data. Specifically the team considered a GlobalSat USB GPS receiver, which costs $27.89,[71] and an Adafruit GPS breakout board, which costs $39.95.[44] All alternatives considered use the NMEA 1083 protocol. The Adafruit GPS board operates between 3.0-5.5 V and has a maximum current draw of 25 mA giving a typical power consumption of 152 mW at 5 V. The Adafruit board claims an accuracy of less than three meters with a maximum refresh rate of 10 Hz. The GlobalSat GPS module operates between 4.5-6.5 V with a typical current draw of 80 mA giving a typical power consumption of 400 mW.[72] The GlobalSat module claims an accuracy of about five meters and does not provide refresh rate information in the data sheet.

5.2.4.1.3 Decision Criteria

Cost, time required for implementation, accuracy, and power are factors affecting the alternative chosen. Time required for implementation was determined by support forums and documented projects detailed online.

5.2.4.1.4 Decision

Because the GPS calculations are being performed on the microcontroller, it seemed intuitive to connect the GPS directly to the microcontroller instead of receiving it on the computer through USB and then sending it to the microcontroller. The team chose to use the Adafruit GPS board as it interfaces with the RC interface module microcontroller selected via analog input pins, and claimed a better accuracy than the GlobalSat GPS. In addition, the increased refresh rate of the Adafruit GPS decreased the time it took the RC interface module to receive the GPS data and take action.

5.2.4.1.5 Implementation

The GPS module is mounted in the RC interface module and connects to the microcontroller's digital I/O pins to send data. Figure 41 and Figure 42 show the inside and outside of the RC interface module, which houses the GPS module.

Figure 41: RC Interface Module External Photo

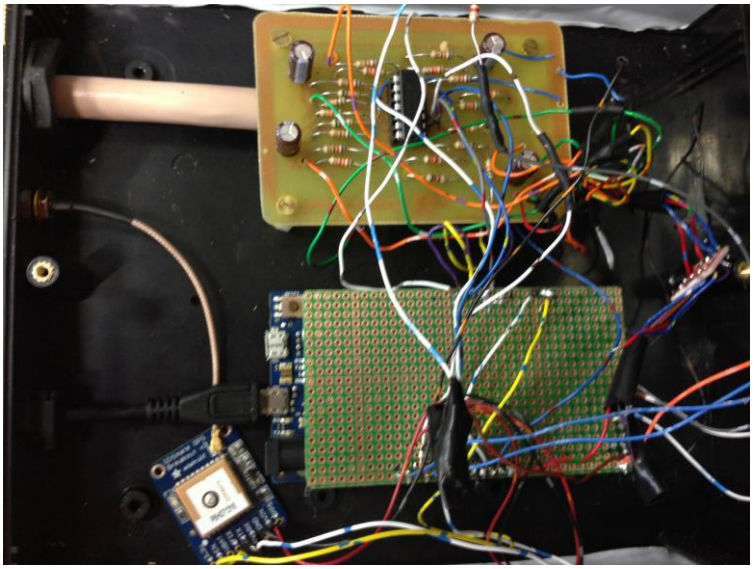


Figure 42: RC Interface Module Internals

5.2.4.1.6 Unit Test

The team gathered the GPS coordinates from the GPS module by printing the data on a serial monitor and verified those with coordinates online to confirm that they are received correctly. As seen in Chapter 6, the GPS module was used in tandem with full communication to test the system response and positional precision.

5.2.4.2 Software

There were only minor amounts of software changes needed for the GPS module. The RC interface module microcontroller sends PMTK NMEA messages in the GPS setup command that increases the baud rate of the device from 9600 to 115,200 and increases the refresh rate from the factory

default 1 Hz to a maximum 10 Hz. The motivation for this change was to improve the latency of the feedback in the tracking algorithms (See section 6.1.3).

5.2.5 Wireless Router

The wireless router receives the video and telemetry information from the quadcopter.

5.2.5.1 Hardware

5.2.5.1.1 Requirements

The wireless router used in the system shall be capable of sending multiple types of information over simultaneous wireless channels. This includes sending video, quadcopter telemetry information, and controls from the base station. The router must provide the necessary bandwidth to stream HD video and telemetry data. To send 240p video for the prototype the router must have a bandwidth of 1.8 Mbps (see section 4.2.9.2.1 for calculations). Continually sending GPS data takes a bandwidth of 5.6 Kbps (see section 5.2.2.2.3 for this calculation). This gives a total bandwidth requirement for the router of 1.9 Mbps. The router must have some ability to select communication channels to avoid interference with the RC transmitter, since they both operate at 2.4 GHz.

In addition, the router helps satisfy the following customer requirements: REQ 2.2.1.2.E.P1, REQ 2.2.1.3.A.P1, REQ 2.2.1.3.B.P1, REQ 2.2.1.3.C.P1, REQ 2.2.1.4.A.PI, REQ 2.2.1.4.B, and REQ 2.2.1.4.D.P1.

5.2.5.1.2 Alternatives

Router alternatives for this project include an AT&T 2Wire router[73] and an Apple Airport Express router.[74] The two frequencies used by IEEE 802.11n are 2.4 and 5 GHz.

5.2.5.1.3 Decision Criteria

Since all commercial routers provide the necessary bandwidth and channel selection capabilities outline above, the cost of the router, range, and time necessary for implementation were the major criteria. The time necessary to get the routers implemented was found by first looking at the AT&T 2Wire router and attempting to set it up as a network, which was not allowed by a custom firmware running on it. After two hours attempting to change the AT&T router settings, the Apple router was tested and set up in less than one hour in order to fulfill the requirements.

For a final implementation of this project, 5 GHz would be considered because it would not interfere at all with the 2.4 GHz RC transmitter.

5.2.5.1.4 Decision

Since the AT&T 2Wire router had a custom firmware that would have needed to be modified for the project, the Apple Airport Express router was used since it was possible to set up for the needs of the project.

5.2.5.1.5 Implementation

The router is located in the vehicle for this project. This allows the base station to connect to the VGPU onboard the quadcopter. Static IP addresses are set up for the VGPU and the laptop in order to make video communication simpler to implement for the prototype version, where only one instance of the product is in use. Static IP addresses were chosen so that the VGPU knows the base station computer's IP address is unchanging and can therefore send information to the same location every time.

5.2.5.1.6 Unit Test

The router was tested by connecting multiple devices to it and running the video stream as well as streaming the telemetry data. The router passed this test, as it was able to simultaneously provide connection to all of the devices while streaming both telemetry and video information during full integration tests.

6 System Integration and Testing

This chapter discusses the system integration tests the team performed to ensure the system was working correctly. It is divided into function sections of telemetry streaming, video streaming, GPS tracking, RC control via RC interface module, and tracking algorithm. In addition, these functional sections are again divided according to the team's iterative building philosophy.

6.1 Telemetry Streaming

This was a test to see if full communication could be achieved from the quadcopter to the base station. This test was performed in several stages. The first step was to pass a message from the quadcopter, through the GUI, and into the RC interface module. Second, the team attempted to pass an actual telemetry message (with GPS coordinates) from the quadcopter, through the base station computer, and to the RC interface module. The last test attempted to quantify the latency experienced with communication and evaluate it.

6.1.1 Test: Full System Communication

This was the first test of telemetry streaming to see if messages sent from the quadcopter could be passed through the GUI and received on the RC interface module.

6.1.1.1 Equipment

- Base station computer
- RC interface module
- Quadcopter

6.1.1.2 Steps

After unit communication tests had been performed on the respective subsections, the system was tested by attempting to pass a message from the quadcopter through the base station computer to the RC interface module. In order to view the results, the RC interface module then sent the message back to the base station computer to be viewed in a debugging text box. These tests can be seen in Appendix A.2.

6.1.1.3 Units of Measurement

This was a pass / fail test.

6.1.1.4 Definition of Success

If the information sent from the quadcopter is received on the base station computer, then the test is a success. A full pass of this test is if the data sent from the quadcopter is passed through the base station computer and received on the RC interface module correctly.

6.1.1.5 Results

This test was performed several times. After working through several bugs, such as setting the correct baud rates and standardizing the message passed, the team achieved test success. The next step was to perform a system communication test with live telemetry data.

6.1.2 Test: Communication with Live Telemetry Coordinates

This test took the previous test step further by actually sending live telemetry data. This tested various parsing functions in both the base station computer and RC interface module to ensure that they selected the correct data variables.

6.1.2.1 Equipment

- Full system

6.1.2.2 Steps

The quadcopter sent a specific data message with GPS information and telemetry data over TCP/IP. The base station computer received this message and forwarded onto the RC interface module via serial port. Again, the RC interface module sent the message back to the computer for viewing and debugging purposes in the GUI.

6.1.2.3 Units of Measurement

This was a pass / fail test.

6.1.2.4 Definition of Success

This test would be a success if telemetry information seen was identical to the message sent from the quadcopter.

6.1.2.5 Results

Initial testing proved that team members did not agree on the order of variables to be passed. Some individuals passed, latitude as the first item in the message, while others passed longitude as the first variable. This miscommunication was fixed and subsequent tests proved that a standardized message was being correctly sent and received.

6.1.3 Communication Latency

The final communication test was one that measured communication latency. This testing used the timestamps of the GPS modules to measure the latency of the system.

6.1.3.1 Equipment

- Full system

6.1.3.2 Steps

The system communication was initialized. Among other things, the time gathered from the base station's GPS (Adafruit) was compared to the time gathered from the quadcopter GPS (MediaTek). This difference was reported back to the team via base station computer GUI debugging boxes as well as debugging files. Communication latency was investigated in the tests from section A.2 in the Appendix and mentioned in section A.5.3.2 of the Appendix.

6.1.3.3 Units of Measurement

The unit of measurement was time difference in seconds.

6.1.3.4 Definition of Success

With a top speed requirement of 4.2 m/s (REQ 2.2.1.2.B.P1), a tracking tolerance of 7.2 m (REQ 2.2.1.2.C.P1), and a worst case of 6 m offset due to GPS modules, the telemetry streaming latency could only account for a maximum of 1.2 m or error at the top speed. $1.2m/4.2\frac{m}{s}$ gives a maximum delay of 0.3 seconds. Streaming telemetry data with latency lower than 0.3 seconds is considered a success.

6.1.3.5 Results

Initial tests showed the latency could take ten seconds to stabilize and then was roughly 1-2 seconds delayed. This was unacceptable. The team implemented serial flushes on the GPS serial ports to

prevent any queuing of data. A subsequent test shows nearly immediate stabilization of latency; however, it was stable around 0.9 seconds. In order to improve this latency, the team changed the refresh rates of both GPS modules from their default speeds of 1 Hz to 10 Hz. This allowed the system to communicate much faster since the data received was much faster. A final test with this new speed increase showed average latencies of 0.2 seconds.

6.2 Video Streaming

The video streaming tests performed quantify the performance of various video streaming protocols. Tests were performed to determine video latency, as well as video quality.

6.2.1 Video Latency

This test helps quantify the latency of each video streaming protocol.

6.2.1.1 Equipment

- Base station computer
- Quadcopter

6.2.1.2 Steps

With telemetry and video streaming running, the team used a stopwatch to quantify the video latency in seconds. The camera pointed at the stopwatch and a picture was taken of the stopwatch and the screen to which the video was being displayed. The difference between the time on the stopwatch and the time depicted on the screen showed the latency of communication. Figure 43 shows this process. Section A.6.1 in the Appendix shows the full latency test.

6.2.1.3 Unit of Measure

The unit of measure was time in seconds to determine the video latency. This test can be seen in section A.6 of the Appendix.

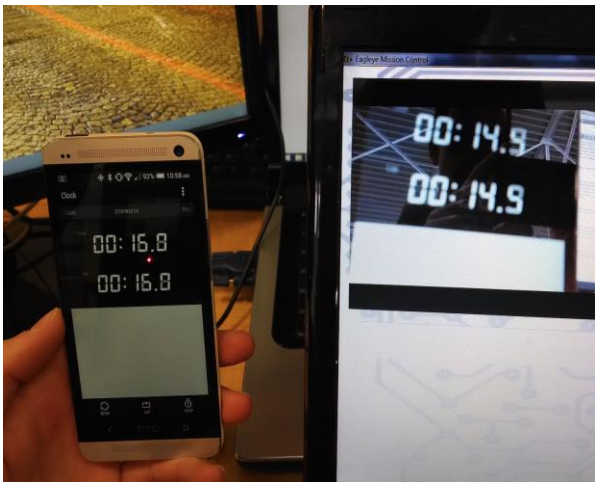
6.2.1.4 Definition of Success

A successful video latency test results in a measured video latency of less than or equal to one second as described in requirement 3.2.1.4.D.P1.

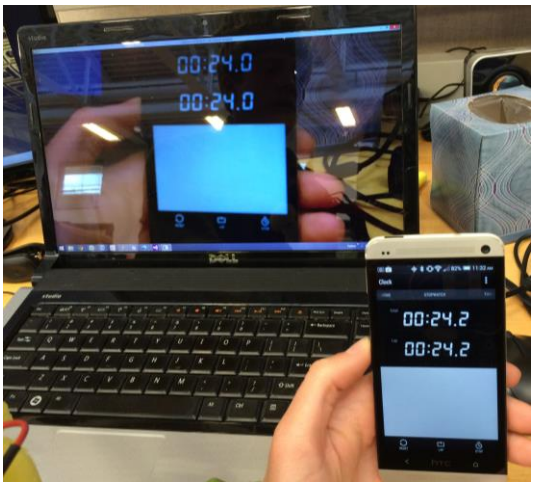
6.2.1.5 Results

RTSP experience a latency of about 1.8 seconds. This test was repeated with GStreamer yielding a latency of about 0.2 seconds. Lastly, FFmpeg was tested and a video latency of 1.7 seconds was observed. Figure 43 shows the results of this test. GStreamer was clearly the best video protocol concerning latency.

RTSP



GStreamer



FFmpeg

Figure 43: Video Latency Results

6.2.2 Video Quality

With latency tested, the team wanted to compare each video stream to their respective video quality. If GStreamer turned out to have far worse quality than the others did, its lower video latency would be outweighed by its poor quality.

6.2.2.1 Equipment

- Base station computer
- Quadcopter

6.2.2.2 Steps

The team flew the quadcopter and streamed telemetry data in order to inspect the video quality during an actual flight, as that is when the video is used. This test can be seen in section A.6.2 of the Appendix.

6.2.2.3 Units of Measure

Four types of errors were used to classify video quality: Type 3, 2, 1 and 0.5. The type numbers correlate to full screen blurs, partial screen blurs, minimal screen blurs, and video jitter respectively. Each error type was weighted. Type 3, 2, 1, and 0.5 were weighted 15, 10, 5, and 2 points respectively. A high score is bad as it shows that particular video protocol had more errors and thus worse quality. Choosing the type number for a flaw was difficult at times, but having two team members watch the stream and agree on the errors helped keep the tests consistent.

6.2.2.4 Results

Figure 44 shows the result of this test. GStreamer proved to be the best protocol in terms of video quality. A recognized weakness of this test is that we knew which video stream had the lowest latency and therefore could potentially bias the results in favor of that protocol. The next iteration of this test would have multiple unbiased parties view the video stream and rank the flaws in order to eliminate this potential bias.

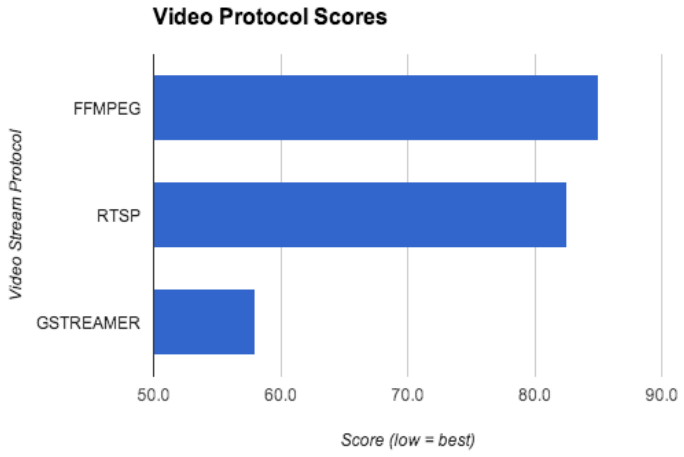


Figure 44: Video Protocol Quality Scores

6.3 GPS Tracking

The tracking tests are used to slowly expand and develop the autonomous tracking algorithms. The first two tests deal with GPS calculations in the microcontroller. The results from this calculation determine where the quadcopter is located with respect to the base station and where it needs to fly next. After this, the remaining tests inspect the RC commands being sent to the quadcopter. These are slowly integrated with the GPS position results to develop the full autonomous tracking algorithms.

6.3.1 Stationary GPS Precision

The first GPS test compares the GPS positions of the GPS modules. Both modules are in the same location, so in theory the calculated differences should be zero meters. Several different tests were performed, many in different arbitrary locations, to help increase variety of the testing environments.

6.3.1.1 Equipment

- Base station
- MediaTek GPS
- VGPU

6.3.1.2 Steps

For the stationary GPS system precision tests, both GPS modules were placed in the same location in a car. For the stationary tests, the car did not move while data was recorded. This was done to confirm REQ 2.2.1.2.C.P1. See section A.5.3 of the Appendix for full results

6.3.1.3 Units of Measurement

Distance offset in meters was used to determine how accurate the GPS modules were.

6.3.1.4 Definition of Success

Since the GPS modules were placed in the same location, the distance offset should be zero meters. The test was considered a success if the difference was reported within a 7.2 m radius as seen in Figure 2 per REQ 2.2.1.2.C.P1.

Table 25 below shows the results of a different stationary test. While the longitude was well within the bounds, the latitude most certainly was not. One possible source of errors was that this location was close to buildings and featured lots of tree cover. It is possible that one or both GPS modules were not able to obtain an accurate GPS fix. Cases like these were handled on an individual basis so that the team could learn from them and improve system performance.

Table 25: Another GPS Stationary Result (10 Hz)

	Time (s)	**Latitude (m)**	**Longitude (m)**
Average	0.2	9.07	1.40
Standard Deviation	0.05	3.28	1.13

6.3.2 Moving GPS Precision

The moving GPS precision test expanded upon the stationary test. Both GPS modules were again in the same vehicle, however now the vehicle was moving. This test, while checking the calculated difference between the GPS positions, also helped the team realize the importance of improving communication latency. The initial moving test was done at 1 Hz refresh rates, and then later repeated with the GPS modules operating at 10 Hz. An important realization was made during these tests. In order to meet REQ 2.2.1.2.C.P1, the GPS modules have to be synchronized with very low latency. While latency does not affect stationary tracking, it has a major effect when the modules are moving.

6.3.2.1 Equipment

- Base station
- MediaTek GPS
- VGPU

6.3.2.2 Steps

Moving GPS precision tests were performed many times. The first test was with a 1 Hz GPS refresh rate, and later tests with a 10 Hz GPS refresh rate. These tests are located in Appendix A.5.3.1 and A.5.3.2, respectively. For these tests, the MediaTek GPS module, VGPU, and base station were placed less than one foot apart inside a vehicle. Once a connection had been established, the team began driving. GPS differences were saved to a file on the base station computer for post-test processing. This process was done for many different routes.

6.3.2.3 *Units of Measurement*

Distance offset in meters was used to determine how accurate the GPS modules were.

6.3.2.4 *Definition of Success*

This test was an informative test about the accuracy of our two GPS modules. Because we were only looking for information from this test, there was no definition of success, but instead, lessons learned about GPS accuracy in different situations as well as the effect of telemetry data latency on system performance.

6.3.2.5 *Results*

In one test, the team drove at speeds around 22.35 m/s (50 mph) and gathered the average longitude and latitude differences. For this particular test, the team was driving north and south; therefore, they expected to see greater latitude differences than longitude differences. Because the time difference was roughly 0.7 seconds, $\left(0.7s * 22.35\frac{m}{s} = 15.6\,m\right)$ this means that GPS data could be off by 15.6 m just because of communication latency. This is an unacceptable range, and it pushed the team to increase the refresh rates of the GPS to reduce this large threshold. Table 26 below shows the data results.

Table 26: Moving GPS Precision Test (1 Hz)

	Time (s)	**Latitude (m)**	**Longitude (m)**
Average (104 points)	0.73	8.81	3.50
Standard Deviation	0.13	10.59	1.54

Figure 45 and Table 27 below shows the results for the test repeated at the 10 Hz refresh rate.

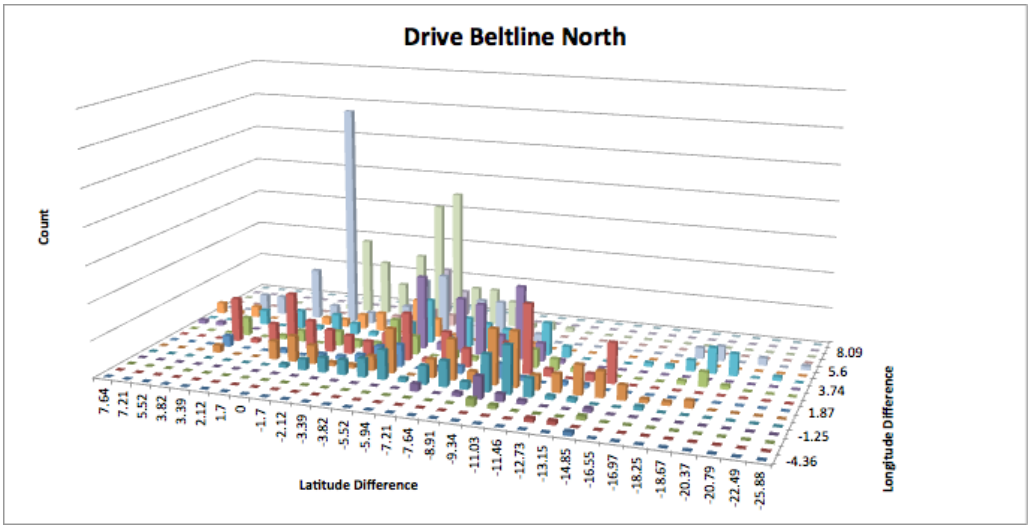


Figure 45: Moving GPS Precision Test (22.35 m/s, 10 Hz)

Table 27: Moving GPS Precision Test (22.35 m/s, 10 Hz)

	Time (s)	Latitude (m)	Longitude (m)
Average (1036 points)	0.20	6.27	3.95
Standard Deviation	0.04	4.51	2.20

The team was pleased with the results given the car could travel four meters at these speeds with a 0.2 second latency. In addition to the tolerance due to communication latency, the six-meter tolerance still exists due to GPS specifications. Combined, this gives a ten-meter window due to tolerances. This is over half of the radius of the previous window with 1 Hz refresh rate. About 56% of the data recorded fell inside the radius requirement of 7.2 m.

In order to test in the most realistic situation the team tested GPS precision while the modules were moving at speeds on average at 16.9 m/s (38 mph) through various city streets. Multiple stoplights were encountered during the test causing speeds to vary from a complete stop to 20.1 m/s (45 mph). Figure 46 and Table 28 below show the results of this test.

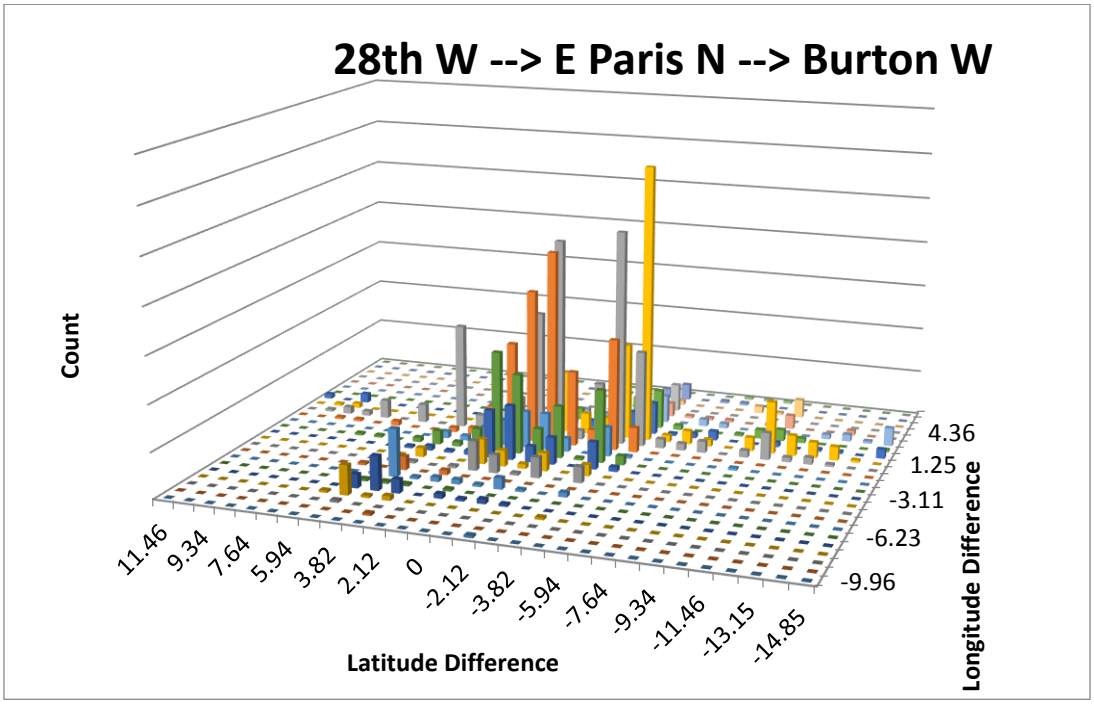


Figure 46: Moving GPS Precision Test (16.9 m/s, 10 Hz)

Table 28: Moving GPS Precision Test (16.9 m/s, 10 Hz)

	Time (s)	Latitude (m)	Longitude (m)
Average (1779 points)	0.2	3.86	1.84
Standard Deviation	0.05	3.01	1.73

The results of this data showed that 88% of the data points fell within the 7.2 m radius. This was a good result. The interesting fact is that direction of travel changed multiple times, as well as the speed. This was definitely the most realistic of the GPS precision tests performed.

Nevertheless, the team was pleased with the results gathered. The 10 Hz refresh rate on the GPS modules greatly reduced communication latency and thus increased GPS precision. With the GPS data proven, the team was able to eliminate a potential source of error in the tracking algorithm.

6.4 RC Control via RC Interface Module

The team performed flight tests to iteratively determine how the system behaved when controlled via base station through the RC interface module. These tests were not used with any GPS or telemetry feedback from the communication loop.

6.4.1 RC Pass Through

This test ensured full manual control was maintained through the RC transmitter even when it was communicating with the quadcopter via the RC interface module.

6.4.1.1 Equipment

- Full system
- Safety net

6.4.1.2 Steps

A program was created to give a full pass through ability to the RC transmitter. This means the quadcopter should fly just as if the RC transmitter were not connected to the RC interface module. To start, the RC values being received by the flight controller were viewed using the open sourced MW GUI. Once these were verified, the quadcopter was flown and controlled using the basic maneuvers: pitching, rolling, yawing, and increasing throttle. Team members held the safety net underneath the quadcopter in case throttle needed to be cut to stop erratic flight behavior.

6.4.1.3 Units of Measurement

This was a pass / fail test determined by inspection.

6.4.1.4 Definition of Success

A successful test was reached once the quadcopter was flown to a hover and controlled as if a regular transmitter was used.

6.4.1.5 Results

Initial tests with the MW GUI were conducted using a purchased logic converter to control the voltage steps both up and down. This test resulted in a non-linear step up and step down of the values. Once the design decision in section 5.2.2.1.2 was made, the test was successful, showing that full control of the quadcopter from the base station microcontroller was possible.

6.4.2 Pitch and Roll Recording

This test was used as a means for gathering the pitch and roll potentiometer data coming out of the RC transmitter. In order to determine what roll and pitch input voltages corresponded to actual quadcopter flight, manual control was used, and the pitch and roll voltages were recorded during specific flight maneuvers. This data was incorporated into the tracking algorithm to autonomously control the quadcopter in a stable manner.

6.4.2.1 Equipment

- Full system
- Safety net

6.4.3 Steps

The team created a program to record RC values during RC pass through as mentioned in the prior section. The quadcopter was flown to a hover while the pass through function was running; enabling the microcontroller to echo the exact RC input values. Once a physical switch on the transmitter was activated, the program would read that switch through a digital input pin and print out the values of the movement in question (pitch or roll). These values were analyzed to get a rough estimate of the appropriate values for the follow program.

6.4.3.1 Units of Measurement

This test was pass/fail, seeing as it would record data or it would not.

6.4.3.2 Definition of Success

A successful flight test was one where the quadcopter was controlled to pitch and roll at certain speeds and the values being sent through the microcontroller were recorded for later analysis.

6.4.3.3 Results

The result of this test was that the quadcopter behaved as expected, and the pitch and roll values were recorded for later use.

6.4.4 Autonomous Pitch and Roll Start

With pitch and roll values recorded, the team needed to test the values on the quadcopter to ensure that the autonomous movement was controlled and stable. The test used to show this is described below.

6.4.4.1 Equipment

- Full system
- Safety net

6.4.4.2 Steps

The quadcopter was flown to a hover at roughly two meters high, but this time when the physical switch on the RC transmitter was flipped, the program would take control of the axis of movement that was under inspection. This test was performed for both pitch and roll independently. Team members held the safety net underneath the quadcopter in case throttle needed to be brought down on the transmitter for an emergency landing.

6.4.4.3 Units of Measurement

This test was a pass / fail test, whether or not the quadcopter responded as expected.

6.4.4.4 Definition of Success

A successful test was one that the quadcopter slowly began accelerating on the axis of test until the team members could no longer keep up with the quadcopter.

6.4.4.5 *Results*

When the quadcopter responded as expected, it was considered a success. It took several iterations of this testing in order to tweak the output values for roll and pitch to achieve the desired response. With this test complete, the team could incorporate this part of the autonomous control into the tracking algorithms. The video of the autonomous pitch test can be found here.

6.4.5 Autonomous Pitch Stop

Since the tracking algorithm chosen would cause the quadcopter to be moving at its fastest speed when entering the 3-meter radius around the base station, a stop function was needed for cases where the base station was stationary. This function reverses pitch for a prescribed amount of time in order to stop within a small distance.

Equipment

- Full system
- Safety net

6.4.5.1 Steps

The team wrote a program such that when the switch was activated, the stop function would control the direction of movement and bring it to a stop. This was verified in the MultiWii GUI to verify correct system response before actual flight-testing. This initial test verified that the test would not have any adverse effects on the quadcopter, which could cause it to become unsafe. Next, the quadcopter was manually hovered at about two meters high and the pilot started the quadcopter moving forward. Once the switch was activated, it called the stop function. Team members held the safety net underneath the quadcopter in case throttle needed to be cut for an emergency landing.

6.4.5.2 Units of Measurement

This test was measured in meters.

6.4.5.3 Definition of Success

A successful test would be one that showed the quadcopter stopping in six meters or less.

6.4.5.4 Results

Since the purpose of the stop function is to bring the quadcopter stop, the test was run multiple times until a full stop was achieved without starting backward movement of the quadcopter. The final stopping function yielded a stopping distance of 3 m +/- 1 m, well within the required distance of six meters. The video of the pitch stop test can be seen here.

6.5 Tracking Algorithm

With telemetry streaming, roll and pitch controls verified, and stop functions finished, the tracking algorithm was realized. By combining all of these subsystems, tests were performed to ensure that they worked together correctly, and resulted in the correct system response. These tests ranged from reading RC values to actually flying the quadcopter and running the program.

6.5.1 RC Response Test

Before testing the tracking algorithm with a flying quadcopter, a basic test was executed to verify the RC interface module algorithm was sending the correct RC commands based on GPS locations.

6.5.1.1 Equipment

- Full system
- Equipment cart
- Vehicle

6.5.1.2 Steps

The full loop of communication was initiated without streaming video, and the base station was powered from a parked vehicle. Next, the quadcopter was placed on a movable cart with a computer connected to the flight controller. The computer was used to inspect the RC values via the MultiWii GUI. To test the actions of the tracking algorithm, the cart with the quadcopter was moved to multiple positions relative to the base station. These positions were determined in order to test the actions of every possible combination of relative positions (i.e.: north, northeast, east, southeast, etc.). Once in each position, the values of the RC receiver were analyzed for correctness. Full documentation for this test can be seen in section A.4.1 of the Appendix.

6.5.1.3 Units of Measurement

This test was pass / fail.

6.5.1.4 Definition of Success and Results

Table 29 shows the positions tested, and the expected actions, which were all achieved. All relative positions are greater than 7.2 meters away in order for the program to take appropriate actions.

Table 29: Expected Reactions to Position Changes

Relative Position of Quadcopter	**Desired Pitch**	**Actual Pitch**	**Desired Roll**	**Actual Roll**
North	Lowered	Lowered	Centered	Centered
North East	Lowered	Lowered	Lowered	Lowered
East	Centered	Centered	Lowered	Lowered
South East	Raised	Raised	Lowered	Lowered
South	Raised	Raised	Centered	Centered
South West	Raised	Raised	Raised	Raised
West	Centered	Centered	Raised	Raised
North West	Lowered	Lowered	Raised	Raised
Within 7.2 m Radius	Centered	Centered	Centered	Centered

6.5.2 Motor Response Test

After verifying that the RC interface module responded by sending the correct RC values, the test was performed on a fully implemented quadcopter. This was not a continuous flying test in the formal sense, as quadcopter "hopping" was performed.

6.5.2.1 Equipment

- Full system

6.5.2.2 Steps

The quadcopter was moved to each possible position relative to the base station (i.e.: north, northeast, east, southeast, etc.) and the throttle was increased to allow the quadcopter to fly just high enough to verify it was pitching and/or rolling in the correct direction before bringing it back to the ground. This type of flight "hopping" helped lower the risk of severe crashes in the event of incorrect system response.

6.5.2.3 Units of Measurement

This test was pass / fail.

6.5.2.4 Definition of Success and Results

When each portion of the test was run, the quadcopter should tilt toward the base station when the flight "hop" was performed. Although motor values were thought to be confirmed per earlier tests, the quadcopter seemed to fly in the wrong direction several times. After testing by quadcopter hopping, the quadcopter correctly navigated towards the desired location. From this point, the team could implement their first complete tracking algorithm as discussed below.

6.5.3 North-Facing Flight Tracking

The first full tracking algorithm assumed the quadcopter always faced north, hence the "north-facing" naming convention. With all of the previous steps completed prior, a full flight test was performed to see the results of this tracking algorithm. The stop function was removed from this test since, once initiated, it never returned control of the quadcopter to the tracking algorithm. Because of this, the quadcopter would drift with wind or momentum once either the latitude or the longitude data came within the threshold. For this test, the GPS calculations were set at an offset of ten meters north of the actual base station. This was done for safety reasons as multiple team members were standing at the base station to initialize the test.

6.5.3.1 Equipment

- Full system
- Equipment cart

6.5.3.2 Steps

The communication loop sending telemetry from the quadcopter to the base station was initiated while the quadcopter was on the ground. The quadcopter was then flown to a hover roughly ten meters away from the base station and the program initiated via switch. Yaw was controlled manually in order to keep the quadcopter facing north. Throttle was also manually controlled so that an emergency landing could be performed at any time during the test. In the software that parses the telemetry and base station position data on the RC interface module, the ground position was modified to be 10 m north of the actual base station. This ensured that the quadcopter would fly at a given position that was not occupied by team members and equipment.

6.5.3.3 Units of Measurement

Since the test was to see whether the quadcopter would track towards the base station, and correct if it drifted out of bounds, this test was pass / fail.

6.5.3.4 Definition of Success/Results

A successful test was one in which the quadcopter continually flew toward the base station, always correcting back towards the base station when it drifted out of the threshold. The quadcopter

tracked the base station within about 20 m, which was expected since a stop function was not being used. More information on this test is found in section A.4.2 of the Appendix. A video of this test is available here.

6.5.4 Orientation Tracking

The next iteration of tracking algorithm included yaw control. In this algorithm, the quadcopter would yaw towards the base station and then begin pitching. As the base station moved, the quadcopter would continue to yaw and pitch, so roll control was not needed in this new tracking method. Two major tests were performed on this new form of tracking and both are described below.

6.5.4.1 Orientation Flight Test

The orientation flight test investigated the yaw control of the tracking algorithm. A preprogrammed flight path was determined, and the quadcopter controlled via that path. These results helped the team realize their second iteration of full tracking.

6.5.4.1.1 Equipment

- Full system
- Safety net

6.5.4.1.2 Steps

The software for the test was written for use without GPS positioning, but with the full communication loop running in order to gather orientation data. The program was started as soon as the quadcopter was hovering at around two meters, and a switch initiated by the user started the program. The program initially ran as if it was ten meters south of the base station and began moving to the north. Once its maximum acceleration was achieved, the program stopped the quadcopter and changed the artificial position to ten meters east of the base station. Since the quadcopter at this point was facing north, the orientation part of the program under test should turn the quadcopter to face west and then accelerate forward. This process was repeated for three sides of a large square. The fourth leg of this square was no conducted in order to ensure the quadcopter did not fly at the operators. The quadcopter was then stopped and throttle was cut, causing it to fall into the safety net. The throttle in this test was again in manual control mode. Team members held the safety net underneath the quadcopter in case throttle needed to be cut for an emergency landing

6.5.4.1.3 Unit of Measurement

Since the test is based on whether or not the quadcopter turns to the appropriate direction before accelerating, this test was pass / fail in nature.

6.5.4.1.4 Definition of Success/Results

A successful test is one where the quadcopter autonomously flew three sides of a box without losing control or becoming unstable. This test was run successfully two times. The third test resulted in the quadcopter hovering on the edge of the orientation threshold. This caused the quadcopter to drift northeast instead of due north, and as a result the quadcopter hit a divider. The test was run a fourth time with the same success as the initial runs. Figure 47 below shows the heading data during this final test. As one can see, the quadcopter starts north, makes a banking turn towards the west, and finishes facing south. More information on this test can be found in section A.3.3 of the Appendix, along with further explanation on the crash. A video showing this box test is found here.

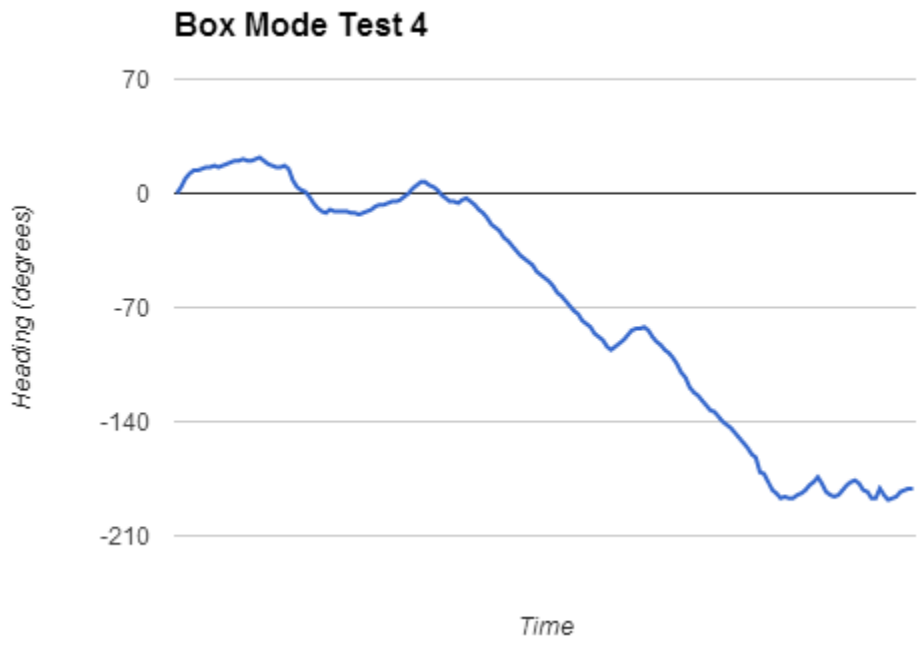


Figure 47: Orientation Flight Test

6.5.4.2 Orientation Flight Test with Tracking

This test combines the proven orientation above and adds it to the full telemetry communication loop. This is a test of the second iteration of full tracking. With a stationary base station, the quadcopter was tested to ensure it turned toward the base station and pitched forward. Then, the stop function was initiated by the tracking algorithm to stop the quadcopter within the tracking threshold. Once the stationary test was validated, the test was run again to verify that it worked with a moving base station.

6.5.4.2.1 Equipment

- Full system

6.5.4.2.2 Steps

To test the final orientation based tracking algorithm the team set up the communication on a movable cart. Once this was established, the team walked around the base station with the quadcopter mimicking flight while viewing RC input data through the GUI. Once this part of the test showed the quadcopter was getting correct RC commands the team tried the test with a flying quadcopter. The quadcopter was flown to a hover in a position approximately ten meters southeast of the base station, and a physical switch on the RC transmitter activated the follow algorithm. Throttle was maintained as a manual control in case the quadcopter needed to be brought down quickly. Once stationary tracking was validated, the test was run from a vehicle, which was moving. This tested the moving base station case of the follow program.

6.5.4.2.3 Unit of Measurement

This test was measured in meters between the base station and the quadcopter.

6.5.4.2.4 Definition of Success

A successful test was one in which the quadcopter would turn toward the base station and begin slowly pitching toward it. Once over the base station, the follow program should call its pitch stop function, causing the quadcopter to stop within 7.2 meters of the base station. Given then quadcopter does not have a way of stopping drift due to wind or small amounts of momentum, a successful test could drift out of the radius, causing the quadcopter to turn and pitch back to the radius.

6.5.4.2.5 Results

When the test was run, the follow program was first validated by walking with the quadcopter and inspecting the RC values. This inspection showed that the quadcopter was trying to yaw to face the base station. Once the quadcopter was turned to face the base station, it received commands to begin pitching toward the base station. These successful results were enough to try the test with the quadcopter flying. With the quadcopter hovering outside the radius, the switch was flipped, and the quadcopter flew into the radius and an attempt to stop was observed. Due to wind, the quadcopter often drifted out of the radius within 10 seconds or missed the radius completely, however the program responded appropriately by turning the quadcopter around again and pitching toward the base station. The team considered this a success. A video of the stationary test is shown here.

With a successful stationary case completed, the team set up the test in a vehicle and moved while the tracking algorithm was running. The result was that the quadcopter continued to update the position it was trying to reach as the base station moved. Like the prior iteration, the quadcopter often blew out of the radius due to wind, but the quadcopter continued to adjust its pitch and orientation to overcome this disturbance. A test video with the moving base station is available here.

7 Project Management

This chapter describes the organizational structure of the project. It shows how the team organized project documents, design areas, management assignments, conflict resolution, and individual safety protocol. In addition, this chapter compares predicted the work breakdown schedule (WBS) and the budget with actual results. Finally, this chapter discusses the method of approach used by the team.

7.1 Documentation Organization

The team organized all of its documents into Google Drive, which is online real-time collaboration software that is entirely cloud-based. It allowed the team to work on various documents simultaneously from any location. The documents are organized into several main categories; Code, Design Journals, Manuals, Media, Miscellaneous Notes, Planning Documents, Status Reports, Team 08: Turn-In, and Tests. Only one document resides outside of these categories, and that is the Time Tracking document. This is accessed frequently by the team and is where each team member records time. Because of the frequency of access, it was not placed into a sub-category. Below, each of the main categories is described.

7.1.1 Code

The Code folder is where the team kept all of the working software code for the project. This provided an area from which each team member did revision control, keeping a record of software as it changed throughout the course of the project. This also allowed all members to see the various source code for consistency and brainstorming purposes.

7.1.2 Design Journals

Design Journals were the place where each team member kept a design area as they worked through various portions of the project. Each team member was required to keep track of various sketches and calculations as design work was completed. There was a separate folder for each team member to keep this information. It also provided an audit trail to prove various design decisions along the way.

7.1.3 Manuals

The Manuals section contains manuals for different components relating to the project. This could be a manual for the flight controller on the quadcopter, documentation on the Raspberry Pi, or information on the RC transmitter.

7.1.4 Media

The Media folder contains many various media that pertains to Team Eagleye. This includes a wide variety of videos during testing throughout the year. It also holds the team posters, concept art, and each of the presentations that were given during the year.

7.1.5 Miscellaneous Notes

Miscellaneous Notes is a place where team members placed anything that does not seem to fit anywhere else. This section holds different brainstorming that the team did on finding a solution. This is also the area where notes were kept on meetings with the team's faculty advisor and industrial consultant.

7.1.6 Planning Documents

The Planning Documents section holds the majority of documents the team has created. Version control was implemented such that any modifications done to a reviewed document are done to a new revision of the document name. This more clearly defines different revisions than those provided by

Google Docs' automatic revision history. This also helped the team keep a baseline history of past revisions in the event that they need to be revisited. This folder also holds the team budget, block diagrams, decision matrix, and feature matrix. It is also where various sub-sections of the Project Proposal Feasibility Study (PPFS) and Design Report were created and initially edited. Lastly, it holds the group's WBS and Gantt chart.

7.1.7 Status Reports

Weekly status reports were created in this area and emailed to our faculty advisor. These status reports include hours for each person during the week, for the entire project, as well as the group's running total of hours. The reports also include tasks completed, goals for the next week, along with any issues that have hindered progress and may result in the need for outside help.

7.1.8 Team 08: Turn-in

The Turn-in folder is the main folder shared with our faculty advisor. It is where documents were placed by the team for official review. Any document placed in here only returned to its original destination as a new revision. This ensured that edits are not made on documents currently awaiting review; it also was a critical part in implementing version control.

7.1.9 Tests

"Tests" is the file location where the team placed various tests that were performed throughout the year. This included both unit tests as well as integration tests. Each test has a specific folder that contains the raw data, charts, and graphs in a spreadsheet, and a summarizing file. This file explains the test that was performed, displays the results, and includes links to the source code that was used during the test.

7.2 Team Organization

This section describes each team member's design areas and management tasks. In addition, this section also establishes guidelines for how the team is to collaborate, design, and maintain safety. The team organization was done in such a way that each member had specific technical and managerial assignments.

7.2.1 Technical Assignments (Design Areas)

The project was been split into six different design areas. Each team member was assigned as the primarily lead on at least one design areas, and as a secondary lead for at least one different area. The amount of design areas assigned to an individual was directly related to the relative size of design work in each area. The size of design work was estimated by what the team thought the complexity of each piece was. This changed throughout the year as areas became larger or smaller in scope then originally predicted. Having secondary members on a design area allowed for better cohesion within the group. It also helped with task management and troubleshooting, as the secondary person provided support and helped when a design area was behind or experiencing difficulties. These design areas are seen in Table 30 and the individual design areas are detailed below.

Table 30: Design Areas and Team Member Assignments

Design Area	Primary	Secondary	Base Station Components	Quadcopter Components
Quadcopter	Drew	Robert	RC Transmitter	All
Telemetry Data from Quadcopter	Drew	Spencer	Base Station Computer, Wireless Router	Flight Controller, GPS Module, VGPU
Tracking Algorithm	Robert	Spencer	Base Station Computer, RC Interface Module, GPS Module	None
RC commands	Robert	Spencer	RC Interface Module, RC Transmitter	RC Receiver
GUI	Jared	Spencer	Base Station Computer, RC Interface Module, Wireless Router	VGPU
Video Streaming	Spencer	Jared	Base Station Computer, Wireless Router	VGPU, Camera

7.2.1.1 Quadcopter

The quadcopter design area incorporated all flight aspects of the quadcopter. This included getting the quadcopter assembled and calibrating the flight controller in order to achieve stable flight. Other aspects of this design area included crash repairs, physical modifications needed for any reason, and the mounting of additional hardware. This section required all components of the quadcopter, as well as the RC transmitter from the base station.

7.2.1.2 Telemetry Data from Quadcopter

This section included sending GPS coordinates and other telemetry data from the flight controller and GPS module through the VGPU. From here, it is sent to the base station computer using the network provided by the wireless router. The base station computer must receive this data and convert it into a usable form.

7.2.1.3 Tracking Algorithm

Once the base station computer receives the GPS position and telemetry data from the quadcopter, it is sent to the RC interface module. This is where the tracking algorithms reside. The tracking algorithm then sends its corresponding data to the RC commands design area.

7.2.1.4 RC Commands

The tracking algorithm output tells the RC interface module what action is needed, and the RC commands design area handled the conversion of flight commands to RC signals. Once this conversion was complete, they were sent over RC to the RC receiver on the quadcopter.

7.2.1.5 GUI

The GUI design area included displaying both the video stream and telemetry data. It handled all of the communication between the VGPU on the quadcopter and RC interface module. In addition, it provided an interface from which a user can interact with the system, as well as a debugging interface for testing the system.

7.2.1.6 Video Streaming

This design area worked with both the VGPU and camera as well as the base station computer. The best video streaming protocol was found such that it met all previously defined requirements. This design area required communication between the base station computer and VGPU over the network provided by the wireless router.

7.2.2 Management Assignments

Jared Zoodsma: Financial Manager - Maintained records of financial expenditures and budget status. All part orders were made through Jared in order to avoid duplicate or incompatible components being ordered.

Robert Hoff: Marketing Manager - In charge of all media and public aspects of group work. Duties included in marketing were posters, website updates, social media, and graphical diagrams requiring computer software.

Drew Brandsen: Professionalism Manager - In charge of maintaining document formatting and professional look. Drew proofread team documents before turning them in so that consistency was maintained between all documents.

Spencer Olson: Team Manager - In charge of keeping track of deadlines set by the faculty adviser, as well as intermediate deadlines. He was also responsible for turning in completed assignments and aiding overall team direction.

7.2.3 Working Guidelines

As seen above, the project has been broken into several design areas. Each member of the team was assigned at least one primary design area and other secondary design areas, as extra assistance was needed. If a component or task fell into one of these design areas, it was the primary investigator's responsibility to ensure the task was completed. The primary investigator asked for assistance from the secondary investigator, or others if need be, but the direct responsibility fell on the primary investigator.

Frequent meetings were held to ensure the team was up to date on members' current tasks. In addition, a spreadsheet was used to keep track of team hours. Every time a team member entered time, he also classified the type of work (design or administrative), included what area he worked on, what was accomplished, and what will be worked on next. This was another way the team stayed accountable and current on what other members were working on.

In the event a team member did not meet a deadline, the team member was expected to complete work in order to make up for the short falling. If the deadline was a stretch goal that has little to no effect on other components, the repercussions were minimal. If deadlines were continually missed, a meeting would have been held with all team members with the importance of making deadlines being stressed. Thankfully, throughout the year all team members were hard working and met most deadlines on time. No one person was consistently delaying the team causing the need for a team meeting. In addition, the team reviewed the workload for all team members to ensure all members have reasonable expectations and made appropriate changes when necessary.

7.2.4 Safety Guidelines

The team had a number of guidelines that provided limits on how work is done. When flying without the tether, two team members needed to be present. This was for both safety and debugging reasons. If a team member gets hurt while flying the quadcopter it is important to have someone else there to help. In the event of a crash, it helped to have more than one set of eyes to see the crash. Since each person may have differing opinions on what the crash can affect, having an extra set of eyes made the debugging of errors much smoother after crashes.

7.2.5 Team Communication and Accountability

The team met weekly in order to discuss overall direction and scheduling. These meetings helped to foster communication between team members, ensuring that all members meet expectations and know what needed to be completed next. An hourly record of member project work was also recorded to ensure that it was obvious if a team member is pulling too much, or not enough, weight in the project. Team members hold each other accountable for putting in considerable time every week in order to keep the project moving, but also understand when personal circumstances arise.

7.3 Schedule and Work Breakdown Schedule

The following section details the schedule of the project and discusses major milestones completed throughout the project.

7.3.1 Hours Summary

During the course of the year, the team kept track of what time was spent on what areas of the project. This was done using a spreadsheet online. The team also developed a WBS to estimate and guide the schedule of design and administrative tasks. This estimate includes a contingency factor of 1.5 to help mitigate unplanned delays and missed tasks. A list of the hour breakdowns is seen in Table 31 below.

Table 31: Summary of Hours

	Design Time	**Admin Time**	**Total Time**
Project Estimate	*773 hours*	*416 hours*	*1189 hours*
Project Estimate (1.5x)	1159.5 hours	624 hours	1783.5 hours
Project Actual	1115.75 hours	839.0 hours	1954.75 hours
Difference	**43.75**	**215.0**	**171.25**

As you can see, the team has completed a total of 1115.75 design hours out of 1159.5 hours estimated, or about 43.75 hours below estimate. Administrative hours prediction was low as the team went 215.0 hours over the estimate of 624 hours. Overall, the team is 171.25 hours above the estimated total hours. However, this is only 9.6% higher than the predicted hours, which is very good for a yearlong project. A cumulative graph of hours worked this year is seen in Figure 48. The red area shows the design hours and the blue area shows administrative hours. The team is happy with the mixture of administrative hours and design hours. A complete copy of the team's WBS is found on the team's website.

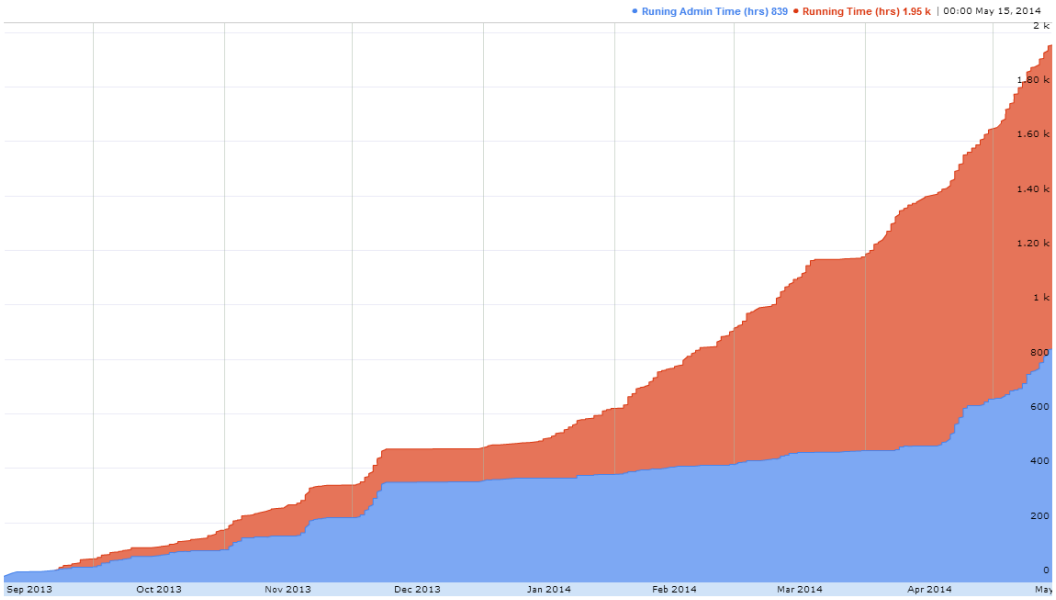


Figure 48: Graph of Hours Worked on Project

7.3.2 Milestones

Major milestones observed in this project are seen below.

- *Project Selection and Approval*
- *October 14 - First Presentation*
- *October 17 - Streaming Video*
- *November 13 - Stable flight of Prototype 0.1*
- *November 19 – Operational Budget*
- *December 2 - Second Presentation*
- *December 6 - Full control of Prototype 0.1*
- *December 9 - PPFS*
- *January 22 - Prototype 0.2 flying, basic RC control through Arduino achieved*
- *February 21 - Basic GUI, telemetry data successfully gathered from MW*
- *March 5 - Third Presentation*
- *March 14 - Base Station under construction, flying with video stream*
- *April 5 - Prototype 1.0 assembled and flying*
- *April 11 - Autonomous tracking of stationary Base Station*
- *April 15 - Orientation added to autonomous tracking*
- *April 25 – CEAC Review*
- *May 5 - Fourth Presentation*
- *May 5 – Faculty Review*

- *May 10 - Final Presentation and Senior Design Open House*
- *May 15 - Final Design Report*

7.4 Operational Budget

In order to manage funds and stay within a reasonable financial range, the group used a budget to project costs and track funds appropriately. The budget estimation is compared to the actual funds that were spent. A summarizing section at the end discusses some of the discrepancies that occurred as well as the final resulting expenses.

7.4.1 Project Costs

Table 32 is the team's operational budget. On the left hand side are the budgetary projections as of December 9, 2013. It includes a contingency factor of 1.5 to anticipate failures, crashes, and unanticipated costs. On the right hand side of the table are the actual unit prices and quantities purchased. The final column shows the net difference between the two.

Table 32: Team 08: Eagleye Budget

Budget Estimation as of 12/9/2013			**Actual as of 5/15/2014**		
Part	**Unit Price**	**Quantity**	**Unit Price**	**Quantity**	**Difference**
Flight Controller[75]	$160.00	1	$65.36	1	$94.64
GPS Module[35]	$40.00	2	$39.99	1	$40.01
ESC[76]	$20.00	5	$13.44	5	$32.80
Set of Four Propellers[77]	$10.00	3	$9.77	7	-$38.40
Raspberry Pi Camera[59]	$25.00	1	$34.11	1	-$9.11
Raspberry Pi[78]	$35.00	1	$35.00	1	$0.00
Arduino Due[79]	$50.00	1	$45.01	1	$4.99
Wi-Fi Dongle[80]	$15.00	1	$9.99	2	-$4.98
XBee Antenna[81]	$20.00	2	$0.00	0	$40.00
Brushless Motors[82]	$15.00	5	$17.71	7	-$48.97
RC Transmitter[83]	$30.00	1	$34.97	4	-$109.88
Frame and Landing Gear[84]	$80.00	1	$19.99	1	$60.01
Cables and Connectors[85]	$50.00	1	$28.93	1	$21.07
Shipping Costs	$100.00	1	$113.55	1	-$13.55
Expandable Features	$100.00	1	$17.38	1	$82.62
Unplanned Quad Parts	$0.00	0	$92.86	1	-$92.86
Unplanned Base Parts	$0.00	0	$83.51	1	-$83.51
	C. Factor	1.5x		Current Total:	**$1005.26**
	Total:	**$1,455.00**		**Remaining:**	**$449.74**
				Percentage:	**69.09%**

7.4.2 Summary

The first thing one notices is that the team is currently under budget by nearly $450.00 (31%). There are a number of reasons that this occurred. First is smarter component selection. During interim, another flight controller was tested that proved equally capable as the $160.00 APM. This flight controller saved the team nearly $100 in costs and included a GPS module, adding another $40.00 in savings. The team also had a number of components donated by Professor Yoon Kim. These parts included four motors and an aluminum frame and landing gear set. This saved the team another $150. However, there were additional expenses in items such as propellers, RC transmitter, and other unplanned quadcopter and base station components. The RC transmitter was a particular point of pain as the first two transmitters were destroyed through their connections to the RC interface module. As such, the team needed to order two more RC transmitters (one as backup). As a result, this particular component went well over budget. However, through the donations gathered and wiser component selection, the team showed great stewardship in the funds they were granted. However, for a real world project, going over in hours causes the project to go over budget too. As section 7.3.1 highlights, the team went slightly over the expected time for the project.

7.4.3 Sources of Funding

The team received two main sources for funding the project. One source of funding came from the team's corporate sponsor, DornerWorks. DornerWorks graciously offered to donate $1,000 towards the team's budget. Calvin Engineering Department's Senior Design Fund provided the remaining $455 outlined in the budget above. This value was originally $726.50, however, the team was pushed towards cutting down on the budget in order to meet the wider Senior Design budget for this academic year.

7.5 Method of Approach

The Method of Approach section describes how the team executed the design process. It also discusses specific design norms that influenced design decisions, as well as research techniques used.

7.5.1 Design Methodology

When deciding on how a design should come together, the team used a strategy that combined research and group brainstorming sessions. When a design decision was needed, the group recorded various options and the effectiveness of each were subsequently researched and evaluated. Once the group had ample time to research the topic, those who found the most pertinent material presented it to the group in an informal manner and a decision was ultimately made.

As for the actual designing, the team employed agile development. This strategy encourages small but positive design steps towards achieving the final design goal. This was especially evident in the software design, which included many revisions as new features and bug fixes were added to the code.

7.5.2 Research Techniques

In order to find information that was relevant to topics under question, the team used a combination of forums, informational websites, scientific journals, as well as professors on campus. Depending on the topic, the team chose the most effective source, which usually depended on the size of the design in question. For more broad design questions, research databases and scientific journals were utilized, while smaller technical questions were easily answered by forums. One of the major goals of this portion was to consider all relevant options including new options uncovered by research.

8 Business Plan

This chapter develops a comprehensive business plan around the project. Beginning with the overview, the plan discusses the vision for the company, a current market study, competition, a business model, and lastly the financial forecast. The next chapter will give a conclusion for the project by stating major risks as well as analyzing the feasibility of the project.

8.1 Overview

Eagleye LLC is an engineering design firm that specializes in small, unmanned aerial vehicles (UAVs). Eagleye was founded on the principles of trust, justice, and transparency. Eagleye LLC's staple product is a small autonomous quadcopter. This quadcopter is designed to capture aerial video and stream the video to a remote location in real time. In addition, the quadcopter is capable of following a ground vehicle autonomously. This product could be applied to various security situations such as aerial reconnaissance over moving targets or highly maneuverable perimeter surveillance.

8.2 Vision and Mission Statement

The vision and principles of the company are listed below and describe the drive that is behind the company.

8.3 Vision for the Company

Eagleye LLC is a company that strives to improve the protection of military and diplomatic personnel by using UAV systems. The team understands the sacrifice troops and leaders make for this country and its freedom. Eagleye LLC's mission is to give back by providing them with one more layer of security as they travel in hostile environments. This gives Eagleye's work on UAV systems a sense of justice and purpose.

8.4 Values and Principles on which the Business Stands

The products created by Eagleye represent its employees personally. Poor products reflect poorly on those who designed and built them. As such, Eagleye strives for the highest quality in each of its products and operations. Above all, the products produced must convey a sense of trust. If those using Eagleye's products cannot trust the information they are being given, the entire purpose of the quadcopter product is lost. The other major principle is transparency. It must be inherently clear what information the users are receiving from the system. In other words, the product must never be so difficult or complicated to use that it actually decreases the awareness of the user. These principles guide Eagleye LLC as its team designs and produces next-generation UAVs.

8.5 Marketing Study

Eagleye's target market consists primarily of military and security companies. Although military services most likely have some form of UAV or satellite services already, Eagleye attempts to fill a need that is not currently met by existing UAVs. This product would fall between an individual's first person ground view and an existing UAV or satellite flying thousands of feet in the air. It would also fall into a price range that is not currently filled by the UAV market. The following sub-sections detail the market size and growth, along with regulator restrictions, barriers to entry and exit, and lastly key success factors.

8.5.1 How Large is the Market?

The current market for military spending on UAVs is about 10.6 billion[86] and for non-military spending is near 6.6 billion[87]. This equates to a total market of around 17.2 billion dollars for both the military and nonmilitary spending on UAVs.

8.5.2 Is it Growing or Shrinking?

The future of UAV technology looks bright, according to AIAA (American Institute of Aeronautics and Astronautics), "*non-military application of Unmanned Aerial Vehicle (UAV) technology is the fastest growing manufacturing sector in the global aerospace industry and expected to grow by 700% between 2012 and 2018."*[87] Furthermore, another article by Market Research Media continues to say that it expects military UAV spending to be over $86.5 billion during the next five years.[86] This shows that both sides of the market (commercial and military) are rapidly growing and expected to continue doing so.

8.5.3 Regulatory Restrictions

The UAV industry has been subject to significant regulation and restriction since military drone strikes as well as privacy concerns have made it a fiercely debated topic. There is a perception among Americans that UAVs represent a threat to citizen privacy and security; however, this is starting to change. Increasingly more Americans are recognizing the vast potential uses for small UAVs in nonmilitary markets such as farming.[87] In fact, the United States Congress recently passed legislation, the Federal Aviation Administration (FAA) Reauthorization Act, which will replace the ban on UAV use with a set of commercial regulations by 2015.[88] Because these regulations are not yet well established, Eagleye LLC will carefully monitor the regulatory situation of the industry, but it is expected that the predominantly untapped commercial small UAV market will quickly become both legal and profitable.[86] There are two additional concerns related to regulatory restrictions, safety requirements, and military standards. Due to the nature of the quadcopter product's status as a small electronic vehicle with fast-moving parts, Eagleye LLC will be careful to comply with safety requirements as the FAA releases them. It is expected that these requirements will be similar in nature to model plane, electronic device, non-car vehicle, and privately owned airplane regulations. When considering products that are intended for military use, Eagleye LLC will be sure to satisfy military requirements as they are stated in any contracts secured; typically, military-grade products have very exact specifications and documentation that is even more detailed on the design process.

8.5.4 Barriers to Entry and Exit

Several factors will create barriers to enter this market. First and most considerable is the large upfront capital needed. Economies of scale will need to be utilized to amortize design costs over multiple sales. In addition, Eagleye will need to work closely with the customers to achieve customer satisfaction and eventually increase name recognition among potential customers.

8.5.5 Key Success Factors in the Industry

A key step towards success for the company will be exposing the overlooked niche this product covers for security and military clients. Companies will not be willing to purchase new UAVs when they do not see the need for them, so it is critical that companies are made aware of how this product can bolster their security without requiring additional labor.

8.6 Competition

Competition is an important factor to consider when entering a market. The competition section looks at both existing competitors in this market, as well as other companies that could enter this market.

8.6.1 Existing Competitors

Existing Competitors are listed below and all compete on some level with Eagleye LLC.

8.6.1.1 Hex AirBot

Hex AirBot manufactures a competing quadcopter product for $160; however, this quadcopter is designed and marketed towards enthusiasts as a toy.[89] Eagleye LLC will respond to this competitor by including features for and focusing on a customer basis of commercial security companies. One crucial feature Eagleye will include that Hex AirBot does not is GPS and vehicle-following abilities.

8.6.1.2 DJI

DJI is an active competitor in the small UAV market, offering four different copters between $110 and $15000.[90] Their target customer base is primarily enthusiasts and those who require airborne video streaming.[90] Eagleye LLC will respond to this competitor in two main ways. First, the product will include features not offered by DJI's cheaper models, such as GPS navigation and vehicle tracking. Second, Eagleye LLC plans on designing and marketing its product for commercial security companies as opposed to exclusively enthusiasts and media professionals.

8.6.1.3 FireBox

Firebox designs and supplies various electronic products; one of these is the MicroDrone quadcopter for $120.[91] The MicroDrone quadcopter is marketed towards enthusiasts as a toy for primarily indoor use.[91] Eagleye LLC will respond to this competitor by focusing on the features that Firebox failed to include, such as video capturing, outdoor flight optimization, and GPS navigation.

8.6.1.4 Lockheed Martin

Lockheed Martin is an active player in the larger military drone market with peripheral activity in the commercial UAV market. They manufacture many unique models of UAVs of all sizes.[92] They could represent a significant threat to Eagleye's market niche with some of their models, such as the quad-vtol UAV; however, they too are more heavily focused on military contracts with very expensive drones than they are on commercial or private security UAVs.[92] Lockheed Martin appears to dedicate much more of its resources to the R&D of new products than it does to the expansion and mass production of its current ones.[92] Eagleye LLC will respond to this competitor by focusing on commercial and private security applications for its significantly cheaper quadcopter product.

8.6.1.5 Walkera

Walkera designs and supplies collectible RC helicopter and quadcopter products. These products are focused on aesthetics and impressive flight patterns, ranging from $70 to $500.[91] Eagleye LLC's quadcopter product is significantly different from Walkera's models because of its video streaming, GPS navigation, and vehicle tracking features. Eagleye's product will also be designed for functionality and robust operation, as opposed to solely aesthetics, which will help them target commercial security companies instead of RC enthusiasts.

8.6.2 Potential Competitors

Potential competitors listed below all have the capability to enter this market and present a viable threat to Eagleye LLC.

8.6.2.1 Northrop Grumman

Northrop Grumman is an active player in the larger UAV market; they manufacture several different drone models. The smallest of Northrop Grumman's UAVs has a 10-foot wingspan, which is still significantly larger than Eagleye LLC's staple product.[93] It is also important to note that the majority of their models are planes, not quadcopters, and that they focus almost exclusively on military customers.[93] If Northrop Grumman were to build smaller, cheaper, more locally useful products, they

could enter the same market niche as Eagleye LLC. This is unlikely because it would represent a very significant change in their target customers and Northrop Grumman currently possesses military contracts, which tend to be reliable source of business.

8.6.2.2 AII Corp

AII is another current competitor in the larger UAV market; they manufacture a few different drones of varying sizes. All of their drones are plane, not quadcopter styled, and they focus nearly exclusively on expensive military and defense applications for their UAV products.[94] If AII were to begin producing UAVs that were smaller and cheaper, they could enter Eagleye LLC's market niche, but this is unlikely, due to the stable nature of their current military customers.

8.6.2.3 SightLogix

SightLogix is a provider of perimeter security technology such as cameras and sensors.[95] This company is interesting because they could be either a potential customer or a potential competitor. If they decide to offer aerial quadcopter surveillance products, they could outfit a quadcopter from Eagleye LLC with their cameras and sensors, or instead develop a competing quadcopter product. Eagleye LLC will ensure that its products are high quality enough to both encourage the purchase of their product by commercial security companies and discourage the in-house development of competing products by those companies.

8.7 Business Model

The business model shows how the company plans to operate. It includes the desired market image, company goals, a SWOT (strengths, weaknesses, opportunities, threats) analysis, as well as a competitive strategy.

8.7.1 Desired Image and Position in the Market

Eagleye desires to be a highly respected producer of small-scale UAVs for security and military purposes. This market niche is nearly empty and ready for new innovative players.

8.7.2 Company Goals and Objectives

The company goals below briefly describe various objectives in both operations, finances, and in other areas.

8.7.2.1 Operational

Eagleye LLC hopes to become a highly innovative company that uses latest technology to design and build UAVs for military and other security customers.

8.7.2.2 Financial

Eagleye must be a profitable venture. Using the first prototype as a showcase, the company hopes to gain further funds to create a robust final prototype that can be used to sell the idea of a quadcopter with these features and abilities to interested customers.

8.7.2.3 Other

Eagleye hopes to inspire a company culture of transparency and collaboration to help spur creativity and continuous improvement amongst employees.

8.7.3 SWOT Analysis

The SWOT Analysis for Eagleye LLC is described below. It discusses the internal strengths and weaknesses of the company, along with external opportunities and threats.

8.7.3.1 Internal Strengths

A major strength of Eagleye is its leaders; they are people who strive for quality and understand that the work they complete reflects on them. Another strength is the product itself. It has highly expandable set of unique features that will set Eagleye apart from other producers of UAVs.

8.7.3.2 Internal Weaknesses

A weakness that is apparent with Eagleye LLC is a lack of market experience and low name recognition. Since Eagleye is a startup, its sales representatives will need to have strong confidence in their products and use extensive knowledge of both the market and customer in order to win additional business. In addition, it is important to ensure Eagleye produces highly reliable products to gain customer confidence.

8.7.3.3 External Opportunities

Primarily, Eagleye LLC hopes to move into a market niche that is not highly developed. Small-scale UAVs can exhibit high quality while being much more cost effective compared to larger well-known UAVs, like the Predator Drone, which runs close to 4 million dollars per vehicle.[96] Eagleye LLC hopes to exploit these differences in building and marketing their products for less money than the competition.

8.7.3.4 External Threats

One threat, which Eagleye LLC recognizes, is the presence of competing companies in similar niches in the UAV manufacturing and design market. Unlike Eagleye LLC, the majority of these companies, such as Northrop Grumman and Lockheed Martin, are well established with very expensive products. These companies have been producing successful products for decades, developing trust between them and their customers. Since Eagleye is a startup, the company will need to provide a flawless product up front in order to begin building similar trust with customers.

8.7.4 Competitive Strategy

A brief competitive strategy is described below to show how Eagleye plans to address competition.

8.7.4.1 Differentiation

Eagleye plans to compete using product differentiation. Because the systems designed are small scale UAVs for security systems, they are much smaller than what is on the market today. This allows Eagleye to operate at a significantly lower price point than most UAV manufacturers. Technology improvements in the last decade have made this possible by integrating many features into a much smaller package. By opening up this new market niche, Eagleye hopes to provide a new solution to customer security needs.

8.7.4.2 Focus

The initial focus will be demonstrating the usefulness and possibilities of small-scale UAVs, as well as encouraging the association of outstanding quality and trust with company products. It will be imperative to build trust with customers right away in order to continue competing in this market. Once brand name recognition and customer trust have been established, focus will shift to providing a greater portfolio of UAVs to allow for even more applications, further increasing the market share and diversifying the customer base.

8.8 Financial Forecasts

The financial forecasts for Eagleye LLC are detailed below. The forecasts include Pro-Forma Income Statements and Cash Flow statements for the first three years. Also included are the Break-Even Analysis and a Ratio Analysis. Each of these statements is derived from supporting calculations that are seen in the next section.

8.8.1 Key Assumptions

This financial analysis rests on several key assumptions including operation without stockholder investment, 10% annual interest rate on bank loans, even distribution of cash flows, no delayed purchases, and capacity-demand assumptions. Operating Eagleye LLC without stockholder investment is an assumption that should be very accurate, as the leadership does not wish to make Eagleye public. The effects of operating without stockholder investment are a more limited initial capital with less accountability and leverage. Assuming a 10% annual interest rate on bank loans is an assumption that is as accurate as research allows. Realistically, this number will deviate from 10% but there is no guaranteed prediction of future interest rates. The annual interest on bank loans affects the financial situation of Eagleye LLC by defining how much money must be paid for each dollar borrowed. Additionally, for this financial analysis it is assumed that all sales and costs are distributed evenly throughout the year in which they are recorded. This assumption makes calculations far simpler and should not significantly change final values. Finally, Eagleye LLC assumed that the company does not have any accounts receivable or payable within their first three years.

The key assumptions that Eagleye LLC is making in regards to capacity and demand involve their ability to produce quadcopters and their ability to sell those quadcopters. For the purposes of this financial analysis, it is assumed that Eagleye LLC will successfully manufacture 10, 200, and 400 quadcopters in each of their first three years respectively. This assumption has a far lower manufacturing quantity for the first year because that time will be spent primarily in product design and manufacturing plant startup. Furthermore, this assumption shows increasing manufacturing quantity as time progresses because of anticipated increase in customer relations and demand. A final assumption that Eagleye LLC is making is that all of its produced quadcopters will be sold within the year in which they were produced. This is intended to simplify calculations, and the assumption's accuracy is dependent upon the industry's current competitive and customer situation. A complete view of supporting calculations for the following financial statements is seen in Table 33 below. This table breaks down fixed and variable costs for each of the first three years.

Table 33: Supporting Calculations

Eagleye LLC Supporting Calculations			
	Fixed Cost Prices	**Fixed Cost Quantities**	
Building Rent & Utilities (Dollars/Month)	$ 10,000.00	12	Months
Design Time (Dollars/Man Hour)	$ 100.00	1200	Man Hours
Prototype Materials (Dollars/Unit)	$ 1,455.00	5	Units
Marketing Campaigns (Dollars)	$ 50,000.00	1	Campaigns
Machines & Tools (Dollars/Set)	$ 10,000.00	1	Sets
Administrative Costs (Dollars/Person)	$ 75,000.00	5	People
Selling Price (Dollars/Unit)	$ 15,000.00	*	Units
	Variable Cost Prices	**Variable Cost Quantities**	
Manufacturing Employee (Dollars/Hour)	$ 50.00	26	Man Hours/Unit
Parts (Dollars/Unit)	$ 800.00	*	Units
Distribution (Dollars/Unit)	$ 50.00	*	Units

*	**Sales**	
Year 1	10	Units
Year 2	200	Units
Year 3	400	Units

8.8.2 Financial Statements

Two financial statements were used to analyze the financial footing of Eagleye LLC. These are the Pro-Forma Income Statement and Cash Flow Statement. They are described in the following sections with the actual statement included.

8.8.2.1 Income Statement

The income statement shows the various revenues and costs for each of the first three years. This statement can be seen in Table 34 below. The most important number in this statement is the net income (or net loss) seen at the bottom of the table. As is seen, the first year results in a net loss of over half a million dollars. This is anticipated, as there are a lot of startup costs and further design work to get the product to production. By year two, the net income is over one million and year three results in a net income of over 2.6 million dollars.

Table 34: Income Statement

Eagleye LLC Pro-Forma Statement of Income			
Line Item	**Year 1**	**Year 2**	**Year 3**
Sales revenue	150,000	3,000,000	6,000,000
Variable Cost of Goods Sold	21,000	420,000	840,000
Fixed Cost of Goods Sold	138,633	138,633	137,275
Depreciation	1,429	3,164	3,688
Gross Margin	(11,062)	2,438,204	5,019,037
Variable Operating Costs	500	10,000	20,000
Fixed Operating Costs	545,000	545,000	545,000
Operating Income	(556,562)	1,883,204	4,454,037
Interest Expense	12,500	17,500	-
Income Before Tax	(569,062)	1,865,704	4,454,037
Income tax (40%)	-	746,282	1,781,615
Net Income After Tax	(569,062)	1,119,422	2,672,422

8.8.2.2 Cash Flow Statement

The cash flow statement, which is shown in the Table 35, highlights the major areas where cash will be spent and received. In this section the loan information, as well as the plan to pay it off are listed, showing the three-year pay off period. In addition, it shows the current value of the loan at each year. This section also highlights the purchase of and depreciation on equipment.

Table 35: Cash Flow Statement

Eagleye LLC Pro-Forma Statement of Cash Flows			
Line Items	**Year 1**	**Year 2**	**Year 3**
Beginning Cash Balance	-	(327,633)	639,953
Net Income After Tax	(569,062)	1,119,422	2,672,422
Depreciation expense	1,429	3,164	3,688
Invested Capital (Equity)	-	-	-
Increase (decrease) in borrowed funds	250,000	(150,000)	(200,000)
Equipment Purchases	(10,000)	(5,000)	(5,000)
Ending Cash Balance	(327,633)	639,953	3,111,064
* Assume no change in Accounts Receivable, Inventory or other current assets other than cash; Accounts Payable or other current Liabilities other than Notes Payable; Fixed Assets other than equipment; or Equity Accounts other than Retained Earnings			

8.8.3 Break-Even Analysis

Break-even analysis is detailed in Table 36 and shows a break-even point for sales of 55, 55, and 54 units in the first three years respectively. This break-even point was calculated for a unit cost of $15,000. Although the anticipated production in year one is only ten units, Eagleye anticipates surpassing the break-even point in years two and three, making up for lost ground.

Table 36: Break-Even Analysis

Eagleye LLC
Break - Even Analysis

Line Items	Year 1		Year 2		Year 3	
Sales revenue		150,000		3,000,000		6,000,000
Less: Variable Costs:						
Variable Cost of Goods Sold	21,000		420,000		840,000	
Variable Operating Costs	500		10,000		20,000	
Total Variable Costs		21,500		430,000		860,000
Contribution Margin		128,500		2,570,000		5,140,000
Less: Fixed Costs						
Fixed Cost of Goods Sold	138,633		138,633		137,275	
Fixed Operating Costs	545,000		545,000		545,000	
Depreciation	1,429		3,164		3,688	
Interest Expense	12,500		17,500		-	
Total Fixed Costs		697,562		704,296		685,963
Income Before Tax		(569,062)		1,865,704		4,454,037

Line Items	Year 1	Year 2	Year 3
Total Fixed Costs	697,562	704,296	685,963
Contribution Margin %	86%	86%	86%
Break Even Sales Volume ($)	814,274	822,135	800,735
Break Even Sales Units	55	55	54

Line Items	Equipment *Purchases*	Depreciation *Year 1*	*Year 2*	*Year 3*
Equipment Purchases Year 1	10,000	1,429	2,449	1,749
Equipment Purchases Year 2	5,000		715	1,225
Equipment Purchases Year 3	5,000			715
Totals	20,000	1,429	3,164	3,688

MACRS Rates (7-year recovery period)	0.1429	0.2449	0.1749
Interest Expense:			
Annual interest rate on debt	10%		
Average debt balance	125,000	175,000	-
Interest expense	12,500	17,500	-
Total Assets, (Cash & Equipment less Depcreciation)	(319,062)	650,361	3,122,783

8.8.4 Ratio Analysis

The Ratios used can be seen at the bottom of this section in Table 37. Note that all ratio values are negative for year one. This is because Eagleye LLC is operating at a loss while it incurs setup costs and before it begins collecting revenue. The debts to assets ratio in years one, two, and three are -39.18%, 26.91%, and 0% respectively. The goal was to pay off debt as quickly as possible to decrease liabilities. The total asset turnover is negative for year one, because Eagleye is not selling a significant number of quadcopters. Eagleye starts selling in year two and this ratio goes up to 4.61 times. After year two Eagleye pays off all debt and begin accumulating the extra cash (assets) that would go to paying interest on debt. For this reason, the total asset turnover goes down to 1.92 times in year three. Again, the profit margin is very poor in year one because the company does not have any sales; however, it gains nicely to 37.31% and 44.54% in years two and three respectively. The same is true for the gross margin of revenue that goes from -7.37% in year one to 81.27% and 83.65% in year two and three. The reason this ratio is so high once Eagleye starts selling units is because of a fairly large markup on the product.

Table 37: Ratio Analysis

Eagleye LLC Ratio Analysis			
Line Items	**Year 1**	**Year 2**	**Year 3**
Gross Margin of Revenue	-7.37%	81.27%	83.65%
Profit Margin on Sales	-379.37%	37.31%	44.54%
Total assets turnover	-0.47	4.61	1.92
Debt to Assets Ratio	-39.18%	26.91%	0.00%

8.9 Summary

In conclusion, Eagleye LLC plans to provide high quality and highly reliable UAVs for security purposes. We feel there is a unique market niche for our product since existing UAV systems can cost millions of dollars. Our low cost and unique product will distinguish us in the growing market of autonomous surveillance. UAV surveillance is a multi-billion dollar market that is only expected to grow. As can be seen in the cost estimation sections above, we believe this is a viable business venture and worth pursuing.

9 Conclusion

This chapter summarizes the project in terms of the final project results. This is broken down into the overall results, costs, and time spent. In addition, the team gives recommendations on the next steps for future work that could be completed. Lastly, it ends with a summary for the project as a whole.

9.1 Project Results

The final results of the project are broken down into three categories; overall, final costs, and total time spent. Each of these categories is explained below.

9.1.1 Overall

The team was able to build a quadcopter system that flew stably. A base station was also assembled that included the necessary displays, computation, and communication equipment to fly the quadcopter autonomously. The team achieved complete communication throughout the system and provided useful information to the user in the GUI. Video streaming and map information was also implemented in the GUI in order to provide the situational awareness required for the user. Lastly, autonomous tracking with orientation was achieved. This was a significant milestone, and the team is happy that they reached all of their major goals for this project.

9.1.2 Final Costs

The team acquired $455 from Calvin's engineering department and a $1000 donation from DornerWorks. The team was very concerned about the budget since the beginning, knowing that any moment during flight tests a crash could set the team back several hundred dollars. With this in mind, the team took several actions in an effort to reduce the risk of running over budget. First, the team used a flight controller that was significantly less expensive than the one budgeted for, yet still performed as needed. In addition, the team acquired several donations courtesy of Professor Yoon Kim and Justin TeBrake. Finally, the team set contingencies in the budget for unplanned items and expandable features. This careful planning helped the team stay under budget

9.1.3 Time Spent

The team spent a total of 1425 hours on the project. This was 120% greater than their estimated time. Looking back on it, if a contingency factor of 2.0 had been used instead of 1.5, the estimation would have been much closer. The hours completed, however, show a dedication to the project by all members and their commitment to see it to completion.

9.2 Future Work

Despite the successes of this project, there is still work to be done. Given more time, the team would like to rebuild the quadcopter with a commercial frame, higher-end flight controller, and higher-end components. Additional sensors could also be added to improve the flight performance such as sonar and an optical flow sensor. These steps would help increase precision of flight (especially in wind), stability of the quadcopter, and the overall tracking.

The base station could also see some improvements as well. A custom board could be made that would replace the Arduino-filter combination. A new 5 GHz Wi-Fi router would cut down interference with the 2.4 GHz RC transmitter. Another option would be to use a lower frequency RC transmitter like 915 MHz. These hardware changes would help make the communication more consistent and reliable.

The software is likely the area that could see the most improvement. There are a lot of checks and fail-safes that could be added to make the software much more robust and deterministic in the corner cases of use. Before production, the team would want to know exact latencies of functions. One would

need to create system timers to find the latencies of the software. For example, one could time how long it takes to receive TCP/IP information, filter it, and send it to the RC interface module in the GUI. The autopilot code can always be tweaked to improve tracking as well. Additional methods of tracking like IR could be added in order to improve tracking accuracy and response time, especially when the quadcopter has line of sight to the vehicle. There are also improvements that could be made on the video and telemetry streaming areas to improve quality and amount of information gathered. Lastly, new features could be added to the GUI in order to increase system interaction and provide better data to the user.

A necessary feature for full autonomy is obstacle avoidance. This was outside of the scope of our project, but is necessary for a production model system. The team brainstormed a couple ways to implement this, but never had time to research it more. One way the team brainstormed was to use sonar sensors in front of the quadcopter. This would be a simple and quick addition to prevent the quadcopter from running into objects. However, it would be difficult to know which direction to move in order to avoid these objects. Another alternative would be to use a camera or several cameras and implement a form of computer vision to detect objects and determine the best course to circumnavigate the object. Adding obstacle avoidance would drastically increase the usefulness and market-readiness of this product.

9.3 Summary

Though this was an ambitious project, the Eagleye team showed the feasibility of this solution through first semester research and preliminary designs. Next, the team designed a working prototype in the second semester that met the project goals and did so while coming under budget. Autonomous tracking of a vehicle was achieved by using a quadcopter as a platform. Video streaming from the quadcopter showed the proof of concept for a product such as this in increasing the situational awareness of the user(s). The team knew from the beginning of the year that the scope of their project was large; however, they were dedicated to the project and worked closely with one another to ensure they met the project goals. While there are always things to improve and redesign, the team is proud of their work accomplished and is thankful to all those that helped them reach this goal.

REFERENCES

[1] (n.d.). In *Calvin College Engineering*. Retrieved December 8, 2013, from http://www.calvin.edu/academic/engineering

[2] Lane (2013, November 7). In *Wikipedia*. Retrieved December 9, 2013, from http://en.wikipedia.org/wiki/Lane#Lane_width_and_capacity

[3] Wikipedia. (2013, November 13). Wikimedia Foundation. In City Block. Retrieved December 6, 2013, from http://en.wikipedia.org/wiki/City_block.

[4] Enhanced F Scale for Tornado Damage. (2007, February 1). In Storm Prediction Center NOAA. Retrieved December 6, 2013, from http://www.spc.noaa.gov/faq/tornado/ef-scale.html.

[5]USA.com. (2013). In Grand Rapids, MI Weather. Retrieved December 6, 2013, from http://www.usa.com/grand-rapids-mi-weather.htm#HistoricalWind%20Speed.

[6] Taoke, G. T. (1989, March). Brake Reaction Times of Unalerted Drivers. *ITE*, 19-21. Retrieved May 12, 2014, from http://www.ite.org/membersonly/itejournal/pdf/jca89a19.pdf

[7] Ergonomics: Lifting limits for employees. (2006, September 2). In Michigan.gov. Retrieved December 6, 2013, from www.michigan.gov/documents/cis/wsh_lifting_176907_7.doc

[8] Whittle, R. (2013, 09). Drone skies. In *Popular Mechanics, 190*, 76. Retrieved from http://search.proquest.com/docview/1440248096?accountid=9844

[9] FPV Quadcopter Shot Down By Radar. (n.d.). In *Finkbuilt.* Retrieved December 8, 2013, from http://www.finkbuilt.com/blog/fpv-quadcopter-shot-down-by-radar/

[10] Hobby Lobby (2013, December 8). In *RC Helicopters | Beginner Heli | Collective Pitch from Hobby Lobby*. Retrieved from http://www.hobby-lobby.com/rc_helicopters_197_ctg.htm

[11] RC Helicopter Selector (2013, December 8). In *Skyartec Cessna 182 2.4 Ghz 5 Channel RC Airplane RTF Brushless Version*. Retrieved from http://www.rchelicopterselect.com/skyartec-cessna-182-2.4-ghz-5-channel-rc-airplane-rtf-brushless-version.html

[12] Squidoo (n.d.). In *Radio Controlled Blimps*. Retrieved December 8, 2013, from http://www.squidoo.com/RemoteControlledBlimp

[13] Blimpworks Airships (n.d.). In *Blimpworks Airships*. Retrieved December 8, 2013, from http://rcblimp.net/blimp_outdoor.htm

[14] Throwable Panoramic Ball Camera Captures 360 Shots When Sent Skyward (2013). In *Steves Digicams.* Retrieved from http://www.steves-digicams.com/news/throwable_panoramic_ball_camera_captures_360_shots_when_sent_skyward.html#b

[15] ArduCopter Introduction (2012, November 17). In *Google Project Hosting*. Retrieved December 8, 2013, from https://code.google.com/p/arducopter/wiki/ArduCopter_QuadIntroduction

[16] NEWBIE, looking for long flight time. (2010, November 27). In *RC Groups*. Retrieved December 8, 2013, from http://www.rcgroups.com/forums/showthread.php?t=1345539

[17] Remote Control Airplanes Review 2014. (n.d.). In *TopTenREVIEWS.* Retrieved December 8, 2013, from http://remote-control-airplanes-review.toptenreviews.com/

[18] 3DR RTF Quad (n.d.). In *3DRobotics Inc*. Retrieved December 8, 2013, from https://store.3drobotics.com/products/apm-3dr-quad-rtf

[19] Align RC 6 Channel Helicopter 600 (n.d.). In *Xheli*. Retrieved from http://www.xheli.com/15h-kx0160npa.html

[20] 3DR ARF APM:Plane (2013). In *3DRobotics Inc*. Retrieved December 8, 2013, from http://store.3drobotics.com/products/3DR-ARF-APM:Plane

[21] RC Blimp: Introducing the MicroBlimp (n.d.). In *Microflight*. Retrieved December 8, 2013, from http://www.microflight.com/MicroBlimp-RTF-Set

[22] Vendittelli, M. (2012). In *Quadrotor Modeling*. Retrieved December 8, 2013, from http://www.dis.uniroma1.it/~venditt/didattica/eir/01_Modeling.pdf
[23] Brain, Marshall, and William Harris. (2000, April 01). How Helicopters Work. Retrieved December 8, 2013 from http://science.howstuffworks.com/transport/flight/modern/helicopter.htm
[24] NMEA 0183. (n.d.). In *Wikipedia*. Retrieved December 8, 2013, from http://en.wikipedia.org/wiki/NMEA_0183
[25] Frequency-hopping spread spectrum. (n.d.). In *Wikipedia*. Retrieved December 8, 2013, from http://en.wikipedia.org/wiki/Frequency-hopping_spread_spectrum
[26] Speed limits in the United States. (2013, August 12). *Wikipedia*. Retrieved December 8, 2013, from http://en.wikipedia.org/wiki/Speed_limits_in_the_United_States
[27] (n.d.). In *NeverWet*. Retrieved December 8, 2013, from http://www.neverwet.com/anti-wetting.php
[28] Optical Flow Sensor. (n.d.).In *ArduCopter*. Retrieved December 8, 2013, from http://copter.ardupilot.com/wiki/optical-flow-sensor/
[29] Quandcopter control function layers. (n.d.). In *HefnyCopter*. Retrieved December 8, 2013, from http://hefnycopter.net/index.php?option=com_content&view=article&id=22:quadcopter-control-function-layers&catid=9&Itemid=103
[30] Engelbert, Wenzel, K. E., Masselli, A., & Zell, A. (2011). *Automatic Take Off, Tracking and Landing of a Miniature UAV on a Moving Carrier Vehicle*. Web: Journal of Intelligent & Robotic Systems.
[31] Build a Miniature High-Rate Speed Control with Battery Eliminator Circuit (BEC). (n.d.). In S*tefanv*. Retrieved December 8, 2013, from http://www.stefanv.com/rcstuff/escbec.htm
[32] Explanation of the Brushless Motor / ESC System and Capacitor Function. (n.d.). In *Traxxas*. Retrieved December 8, 2013, from http://traxxas.com/forums/showthread.php?8958604-Explanation-of-the-Brushless-Motor-ESC-System-and-Capacitor-Function
[33] USB. (2013, July 12). *Wikipedia*. Retrieved December 7, 2013, from http://en.wikipedia.org/wiki/USB

[34] Creating the camera board. (n.d.). In *Raspberry Pi*. Retrieved December 8, 2013, from http://www.raspberrypi.org/archives/3525
[35] Adafruit Ultimate GPS Breakout (n.d.). In *Adafruit*. Retrieved December 8, 2013, from http://www.adafruit.com/products/746#Technical_Detail
[36] AeroQuad 32 Flight Control Board Version 2. (n.d.). In *AeroQuadStore*. Retrieved December 8, 2013, from http://www.aeroquadstore.com/AeroQuad_32_Flight_Control_Board_Version_2_p/aq32-001.htm
[37]APM 2.5+ Assembled (Cables enter from top). (n.d.). In *3DRobotics Inc*. Retrieved December 8, 2013, from http://store.3drobotics.com/products/apm-2-dot-5-plus-assembled-set-top-entry
[38] AutoQuad 6 Autopilot Flight Controller (n.d.). In *Viacopter - Multicopters & Multirotors*. Retrieved December 8, 2013, from https://viacopter.eu/multirotor-shop/flight-controllers/autoquad-6-multicopter-store
[39]MWC MultiWii Lite Lightweight Version 4-axis Flight Control Board QUADX. (n.d.). Retrieved December 8, 2013, from http://www.goodluckbuy.com/mwc-multiwii-lite-lightweight-version-4-axis-flight-control-board-quadx.html
[40] MultiWii 328P Flight Controller w/FTDI & DSM2 Port. (n.d.). In *HobbyKing*. Retrieved April 22, 2014, from http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=35682
[41] MultiWii PRO Flight Controller w/MTK GPS Module. (n.d.). In *HobbyKing*. Retrieved April 22, 2014, from http://www.hobbyking.com/hobbyking/store/__26588__MultiWii_PRO_Flight_Controller_w_MTK_GPS_Module.html
[42]PX4FMU Autopilot / Flight Management Unit. (n.d.). In *PX4 Autopilot Platform*. Retrieved December 8, 2013, from https://pixhawk.ethz.ch/px4/modules/px4f
[43] UAVX-ARM32 Full Sensors. (n.d.). In *QUADROUFO*. Retrieved December 8, 2013, from http://www.quadroufo.com/product_info.php?products_id=88&osCsid=d7lf5mikok4tplcs0lr1814qo3
[44] Adafruit Ultimate GPS Breakout - 66 channel w/10 Hz updates. (2013). In adafruit. Retrieved April 21, 2014, from http://www.adafruit.com/products/746

[45] MultiWii PRO Flight Controller w/MTK GPS Module. (2013). In Hobbyking. Retrieved April 21, 2014, from http://www.hobbyking.com/hobbyking/store/__26588__MultiWii_PRO_Flight_Controller_w_MTK_GPS_Module.html

[46] Estimate Electric Motor and Prop Combo. (n.d.). In adamone . Retrieved December 8, 2013, from http://adamone.rchomepage.com/calc_motor

[47] NTM Prop Drive Series 28-26A 1200kv / 286w. (n.d.). In HobbyKing. Retrieved April 22, 2014, from http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=39435

[48] TURNIGY Plush 25amp Speed Controller. (n.d.). In HobbyKing. Retrieved April 22, 2014, from http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=2163

[49] LiPoBatteries (2013). In *Wikispaces*. Retrieved December 8, 2013, from http://quadcopter.wikispaces.com/LiPo+Batteries

[50] Robotics beta (2012, November). In *StackExchange*. Retrieved December 8, 2013, from http://robotics.stackexchange.com/questions/554/quadcopter-lipo-battery-weight-capacity-trade-off

[51] Turnigy 2450mAh 3S 30C Lipo Pack. (n.d.). In *HobbyKing*. Retrieved May 12, 2014, from http://hobbyking.com/hobbyking/store/__10274__Turnigy_2450mAh_3S_30C_Lipo_Pack.html

[52] Turnigy 3000mAh 3S 30C Lipo Pack. (n.d.). In *HobbyKing*. Retrieved May 12, 2014, from http://hobbyking.com/hobbyking/store/__9497__Turnigy_3000mAh_3S_30C_Lipo_Pack.html

[53] Turnigy 3600mAh 3S 30C Lipo Pack. (n.d.). In *HobbyKing*. Retrieved May 12, 2014, from http://hobbyking.com/hobbyking/store/__9505__Turnigy_3600mAh_3S_30C_Lipo_Pack.html

[54] Turnigy 5000mAh 3S 30C Lipo Pack. (n.d.). In *HobbyKing*. Retrieved May 12, 2014, from http://hobbyking.com/hobbyking/store/__9515__Turnigy_5000mAh_3S_30C_Lipo_Pack.html

[55] 2.4G CT6B 6-Channel Transmitter+Receiver (R6B). (n.d.). In *HobbyPartz*. Retrieved April 22, 2014, from http://www.hobbypartz.com/79p-ct6b-r6b-radiosystem.html

[56](n.d.). In *Raspberry Pi*. Retrieved December 8, 2013, from http://www.raspberrypi.org

[57] Products (n.d.). In *Arduino*. Retrieved December 8, 2013, from http://arduino.cc/en/Main/Products

[58] BeagleBone (n.d.). In *BeagleBone.org*. Retrieved December 9, 2013, from http://beagleboard.org/Products/BeagleBone

[61] (n.d.). In *PCB Quote Online*. Retrieved April 22, 2014, from http://www.bittele.com/pcb-quote-online_T.asp

[62] H.264/MPEG AVC. (n.d.). In *Wikipedia*. Retrieved December 8, 2013, from http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

[63] Real Time Streaming Protocol. (2014, May 4). In *Wikipedia*. Retrieved May 12, 2014, from http://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol

[64] CSharp. (2010, March 22). In *VideoLAN Wiki*. Retrieved May 12, 2014, from https://wiki.videolan.org/C_Sharp/

[65] (2014, April 20). In *GStreamer*. Retrieved April 23, 2014, from http://gstreamer.freedesktop.org/

[66] (2014, May 1). In *FFmpeg*. Retrieved May 12, 2014, from http://www.ffmpeg.org/

[67] Psips (n.d.). In *GitHub*. Retrieved April 23, 2014, from https://github.com/AndyA/psips

[68] Raspberry Pi Camera Board. (n.d.). In *Adafruit*. Retrieved May 12, 2014, from http://www.adafruit.com/products/1367?gclid=CJDshtzEp74CFckWMgodO1QADQ

[69] HERO3 White Edition. (2014). In *GoPro*. Retrieved May 13, 2014, from http://gopro.com/cameras/hd-hero3-white-edition?gclid=CLvpkaenqr4CFahaMgodp24AaQ

[70] Genius 3220032100 WideCam F100 USB 2.0 WebCam (n.d.). In *Newegg*. Retrieved December 8, 2013, from http://www.newegg.com/Product/Product.aspx?Item=N82E16826179091

[71] GlobalSat BU-353 USB GPS Navigation Receiver (n.d.). In *Amazon*. Retrieved December 8, 2013, from http://www.amazon.com/GlobalSat-BU-353-USB-Navigation-Receiver/dp/B000PKX2KA

[72] BU-353 GPS Receiver (n.d.). In *USGlobalSat*. Retrieved December 8, 2013, from http://www.usglobalsat.com/store/download/62/bu353_ds_ug.pdf

[73] 2Wire Gateway User Guide. (2008, June). In *Pace*. Retrieved May 12, 2014, from http://knowledgebase.pace.com/view/766/2700_User_Guide.pdf

[74] AirPort Express 802.11n (1st Generation). (n.d.). In *Apple*. Retrieved May 12, 2014, from http://support.apple.com/kb/sp1

[75] APM 2.6 Set (external compass) (n.d.). In *3DRobotics*. Retrieved December 8, 2013, from http://store.3drobotics.com/products/apm-2-6-kit-1

[76] TURNIGY Plush 25amp Speed Controller. (n.d.). In HobbyKing. Retrieved April 25, 2014, fromhttp://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=2163
[77] APC Propellers 10X47 Push-Pull Set. (n.d.). In 3DRobotics. Retrieved April 25, 2014, fromhttps://store.3drobotics.com/products/apc-propeller-set-10x47-sfp-style
[78] Raspberry Pi Model B 512MB RAM (n.d.). In *Adafruit*. Retrieved December 8, 2013, from http://www.adafruit.com/products/998
[79] Arduino Due - assembled - Due (n.d.). In *Adafruit*. Retrieved December 8, 2013, from http://www.adafruit.com/products/1076
[80] Raspberry Pi WIFI Adapter / Dongle - The Pi Hut (n.d.). In *Amazon*. Retrieved December 8, 2013, from http://www.amazon.com/Raspberry-Pi-WIFI-Adapter-Dongle/dp/B009FA2UYK/ref=sr_1_6?ie=UTF8&qid=1386475850&sr=8-6&keywords=wifi+dongle
[81] XBee 1mW Wire Antenna - Series 1 (802.15.4). (2010, December 13). In Amazon. Retrieved April 25, 2014, fromhttp://www.amazon.com/XBee-1mW-Wire-Antenna-802-15-4/dp/B004G4ZHK4/ref=pd_sim_pc_1?ie=UTF8&refRID=187364R9TRBMVJBBVSD4
[82] NTM Prop Drive Series 28-26A 1200kv / 286w (short shaft version) . (n.d.). In HobbyKing. Retrieved April 25, 2014, fromhttp://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=39435
[83] Turnigy 6X FHSS 2.4ghz Transmitter and Receiver (n.d.). In *Hobby King*. Retrieved December 8, 2013, from http://www.hobbyking.com/hobbyking/store/__24969__turnigy_6x_fhss_2_4ghz_transmitter_and_reciever_mode_2_.html
[84] Turnigy Talon Quadcopter (V2.0) Carbon Fiber Frame. (2014). In HobbyKing. Retrieved April 25, 2014, fromhttp://www.hobbyking.com/hobbyking/store/__22781__Turnigy_Talon_Quadcopter_V2_0_Carbon_Fiber_Frame_550mm.html
[85] Cables (n.d.). In *3D Robotics: UAV technology*. Retrieved December 8, 2013, from https://store.3drobotics.com/t/parts/cables
[86] Haldane, M. (2013, August 8). U.S. Slowly Opening up Commercial Drone Industry. In *Reuters*. Retrieved December 8, 2013, from http://www.reuters.com/article/2013/08/08/us-usa-drones-commercial-idUSBRE97715U20130808
[87] Projected Growth for Non-Military Use of Drones Presents New Business Opportunities (n.d.). In *Business Opportunities Journal*. Retrieved December 8, 2013, from http://www.boj.com/projected-growth-for-non-military-use-of-drones-presents-new-business-opportunities/
[88] Smithson, S. (2012, February 7). Drones over U.S. Get OK by Congress. In *Washington Times*. Retrieved December 8, 2013, from http://www.washingtontimes.com/news/2012/feb/7/coming-to-a-sky-near-you/?page=all
[89] Hex Airbot. (n.d.). In *Hex Airbot*. Retrieved December 8, 2013, from http://hexairbot.com/
[90] DJI | All Products. (n.d.). In *DJI*. Retrieved December 8, 2013, from http://www.dji.com/products/
[91] Micro Drone (n.d.). In *Firebox*. Retrieved December 8, 2013, from http://www.firebox.com/product/5565/Micro-Drone?aff=512
[92] Five Lessons for a Successful STEM Career. (n.d.) In *Lockheed Martin*. Retrieved December 8, 2013, from http://www.lockheedmartin.com/us/products/procerus/quad-vtol.html
[93] Unmanned Systems. (n.d.). In *Northrop Grumman*. Retrieved December 8, 2013, from http://www.northropgrumman.com/Capabilities/Unmannedsystems/Pages/default.aspx
[94] Unmanned Systems. (n.d.). In *AAI Corporation*. Retrieved December 8, 2013, from http://www.aaicorp.com/products/unmanned-systems
[95] Thermal Cameras and Video Analytics for Perimeter Security. (n.d.). In *SightLogix*. Retrieved December 8, 2013, from http://www.sightlogix.com/
[96] Drew, C. (2009, March 16). Drones Are Weapons of Choice in Fighting Qaeda. In *New York Times*. Retrieved December 8, 2013, from http://www.nytimes.com/2009/03/17/business/17uav.html?pagewanted=all&_r=0

APPENDIX

Appendix A : Test Documents 2
A.1 Battery 2
A.1.1 Battery Life Test 1 2
A.2 Communication 3
A.2.1 Communication Latency Test 1 3
A.2.2 Communication Latency Test 2 4
A.3 Field Test 5
A.3.1 Test 1 5
A.3.2 Test 2 6
A.3.3 Box Test 7
A.4 RC 12
A.4.1 Reactions for Following 12
A.4.2 Reactions for Following with Flight 13
A.5 Sensors 14
A.5.1 Barometer 14
A.5.2 Compass 20
A.5.3 GPS 46
A.6 Video 81
A.6.1 RTSP Latency 81
A.6.2 Summary 85

Appendix A: Test Documents

A.1 Battery

A.1.1 Battery Life Test 1

Name: Drew Brandsen, Jared Zoodsma
Date: 3/15/2014
Test Number: 1

Item under test: Battery Life
Testing Parameters:

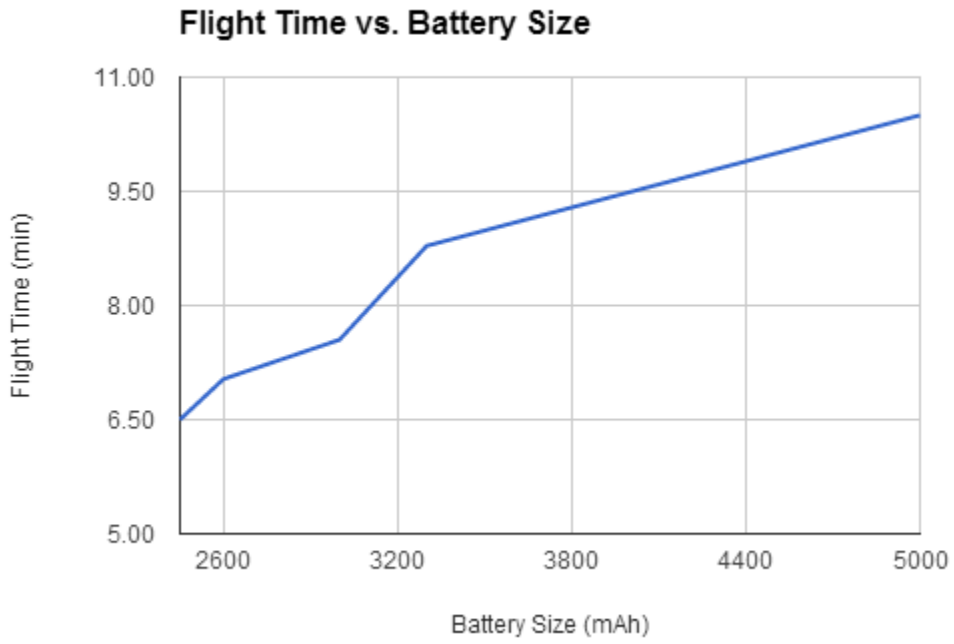
- 2450, 2650, 3000, 3300, 5000 mAh batteries were tested
- Raspberry Pi was powered during test
- No telemetry or video streaming was done during the test

Notes:

- Each battery was tested individually
- A fully charged battery was placed in quadcopter it was flown until the quadcopter could no longer be lifted due to low battery

---

Results:



- There was a fairly linear relation between battery size and flight time
- From this data, we can calculate that the system is drawing between 22 and 24 amps.

A.2 Communication

A.2.1 Communication Latency Test 1

Name: Jared Zoodsma
Date: 3/11/2014
Test Number: 1

Item under test: Communication Latency
Testing Parameters: (Ex. software version "xyz" used)

- GUI software used
- Arduino Code
- Pi Telemetry Code
- MultiWii 328P used

Notes:

- Both GPS modules were placed on west-facing overhead door in engineering building
- Test was done at 2:40 PM

Results:

- Took 10.4 seconds to get "caught up"
    - Once the system was "caught up" there still seemed to be a disparity between GUI system time and time reported from the GPS modules (more on this later)
- Although the clock on the computer was synced right before the test, there seems to be a discrepancy between times in the GUI and from the GPS modules. This can be observed because the difference between Quad time and GUI time is greater than the difference between Quad time and Arduino time (1.71 seconds vs. .80 seconds). This is unexpected because the information arrives at the GUI before it is passed to the Arduino.
    - At first glance this is attributed to unsynced clocks on the devices. Another explanation could be that the time is not updated in the code promptly.

A.2.2 Communication Latency Test 2

Name: Robert Hoff, Jared Zoodsma, Drew Brandsen
Date: 4/2/2014
Test Number: 1

Item under test: Communication Latency
Testing Parameters:

- GUI code
- Pi code
- Arduino code

Notes:

- Base Station (Arduino, GUI) were placed by west overhead door
- Pi with Multiwii-328P were placed by overhead door
- Wi-Fi router placed by overhead door

Results:

- Erroneous latitude differences were reported ~7.5% of the time
    - This data was reported to be ~50,000km latitude difference
    - This issue seemed to go away when GPS modules were moved farther away from the EB. This could mean GPS modules did not have very good fixes
    - A work around could be to have a threshold. If differences are greater than 1km then discard that data.
- Latencies bounced between 0.8 and 1.0 seconds
    - Average of 0.9 seconds

A.3 Field Test

A.3.1 Test 1

Name: Drew Brandsen, Robert Hoff, Spencer Olson, Jared Zoodsma
Date: 2/18/2014
Test Number: 1

Item under test: Communication from quadcopter to base station, and preliminary tracking algorithm and video streaming.

Testing Parameters:

- GUI version "2/19/2014" used on the Gateway
- RTSP video streaming protocol used (code in GUI)
- Arduino Code
- Telemetry Rev D with separate GPS serial port on Raspberry Pi
- Raspberry Pi used was one from the Calvin Lab (not team's Pi).
- Pi Camera
- Adafruit GPS was used on base station
- APM GPS was used on quadcopter
- MultiWii 328p Flight Controller
- Car power adapter used to power the Eagleye network.

Notes:

- Sunny weather with clear skies (GPS had good fix)
- Used Jared and Spencer's cars.
- Quadcopter in one car and base station in another.
- 3 LEDs were used to display close, medium, and far distances between base station and quadcopter in the direction of North, South, East, and West.
- One car remained stationary while another car moved to various locations
    - This was done with both the quadcopter remaining stationary and then again with the base station remaining stationary

---

Results:

- GUI struggled rendering full HD picture.
- Range of video stream was about 66 feet (rough estimate).
- When the quadcopter car was stationary, the tracking algorithm worked well, but when the quadcopter car was moving, there was latency in the data being received on the Arduino.
    - This may have been due to a GPS data delay in the raspberry pi (data was queued and so old data was being sent). This was amended by flushing the GPS serial port within the data loop. Upon restarting the communication this issue was mostly resolved. Data was then being sent within a second of real-time.

A.3.2 Test 2

Name: Spencer Olson, Jared Zoodsma, Drew Brandsen, Robert Hoff
Date: 3/03/2014
Test Number: 2

Item under test: Communication from quadcopter to base station, and preliminary tracking algorithm.

Testing Parameters:

- GUI version used on the Gateway
- RTSP video streaming protocol used (code in GUI)
- Arduino Code
- Telemetry Rev E on Raspberry Pi
- Raspberry Pi used was one from the Calvin Lab (not team's Pi).
- Adafruit GPS was used on base station
- MediaTek GPS was used on quadcopter (came with MultiWii)
- MultiWii Pro Flight Controller
- Bestek car power adapter used to power the Eagleye network and GUI computer

---

Notes:

- Cloudy Weather; GPS still had good fix when in the open
- Used Drew and Rob's Cars
- Quadcopter with Raspberry Pi and camera in Drew's Car. Base station with Wi-Fi Router in Rob's Car

---

Results:

- Found the Raspberry Pi was not receiving enough current to enable all feature (SSH) (when plugged into USB adapter in car)
- Determined a Serial flush between GUI and Arduino was needed once TCP connection had been established to decrease latencies
- Discovered reported distances (60m) were larger than actual distances (20m) when both cars were traveling (major latency issues)

A.3.3 Box Test

A.3.3.1 Summary

Name: *Drew Brandsen, Jared Zoodsma, Spencer Olson, Robert Hoff*
Date: 4/15/2014

Item under test: Arduino Orientation Tracking Algorithm
Testing Parameters:

- Arduino Software
- GUI Software
- Telemetry Software

Notes:

- Quadcopter was to fly a predetermined box pattern
  - Fly north for a few seconds, turn west and fly west for a few seconds, turn south and fly south for a few seconds.
- Initial Test used a +/-30 degree threshold window
  - if within +/-30 degrees of desired heading, begin pitching forward
- Later Tests used a +/-10 degree threshold window
- Expected heading data would be 0 → -90 → -180

Conclusion:

- Three out of four times the quadcopter correctly flew the pattern designated.
  - One time we ran out of room in the TNT and the quadcopter hit the divider between the tennis courts and the track.
- A smaller heading window should be explored to maximize accuracy of pitch movements.
- The tests were considered a success and will be integrated into GPS tracking to further improve our tracking algorithms.

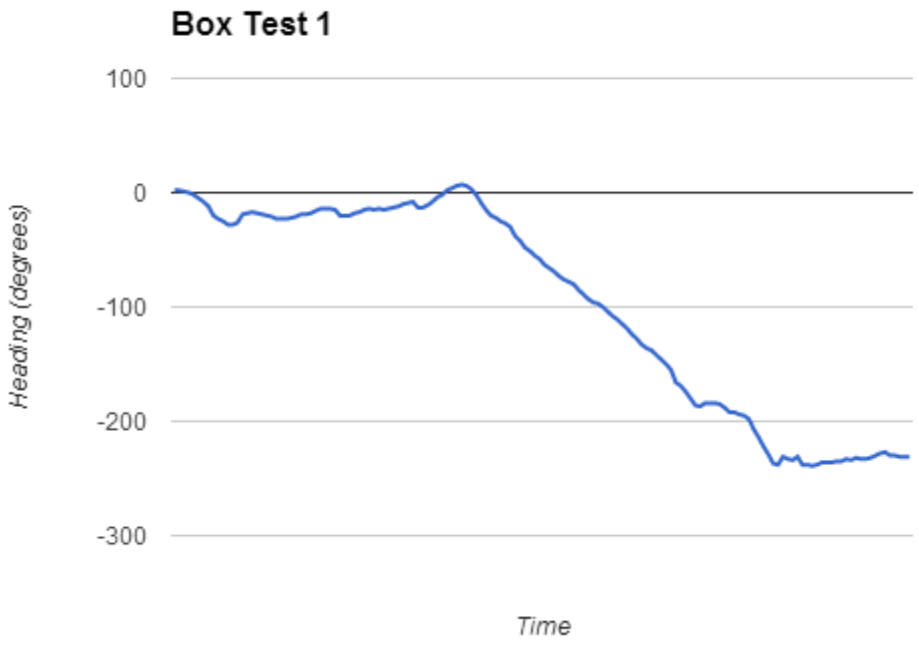
A.3.3.2 Box Test Number 1: +/-30 Degree Heading Window

Notes:

- First test of orientation tracking
- Quadcopter performed excellently
    - Pattern was more of a semi-circle than a box due to the large heading threshold window

---

Results:



Summary:

- Good data seen here - quad holds fairly straight, then consistently yaws towards destination heading. Once at desired heading, again holds steady
- There is a very gradual transition from North to South (during which the quadcopter is pitching slowly forward).
    - This is likely due to the large pitch threshold for the heading data (+/- 30 degrees)

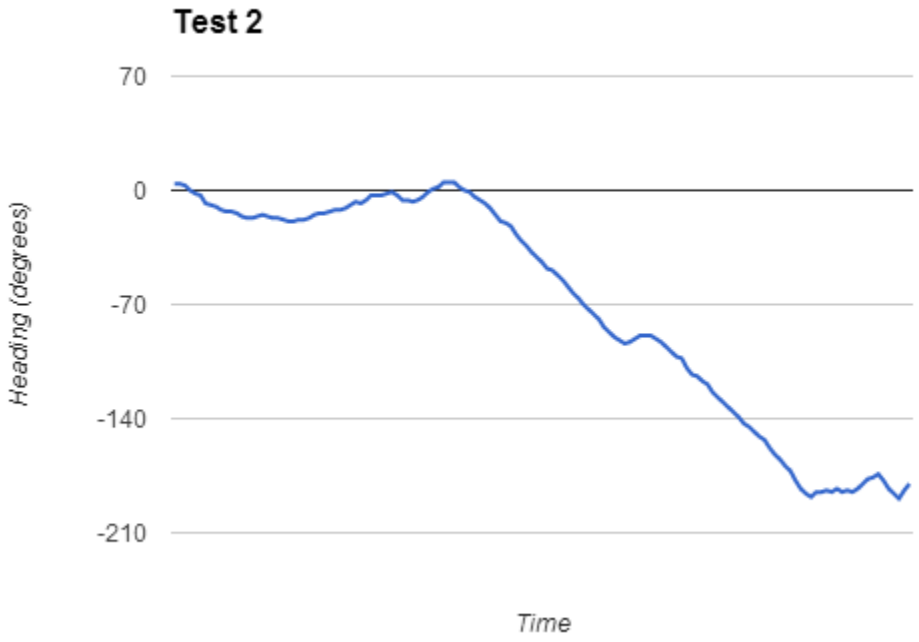
A.3.3.3 Box Test Number 2: +/-10 Degree Heading Window

Notes:

- Quadcopter performed great again
    - Pattern was much more a box due to the smaller heading threshold window

---

Results:



Summary:

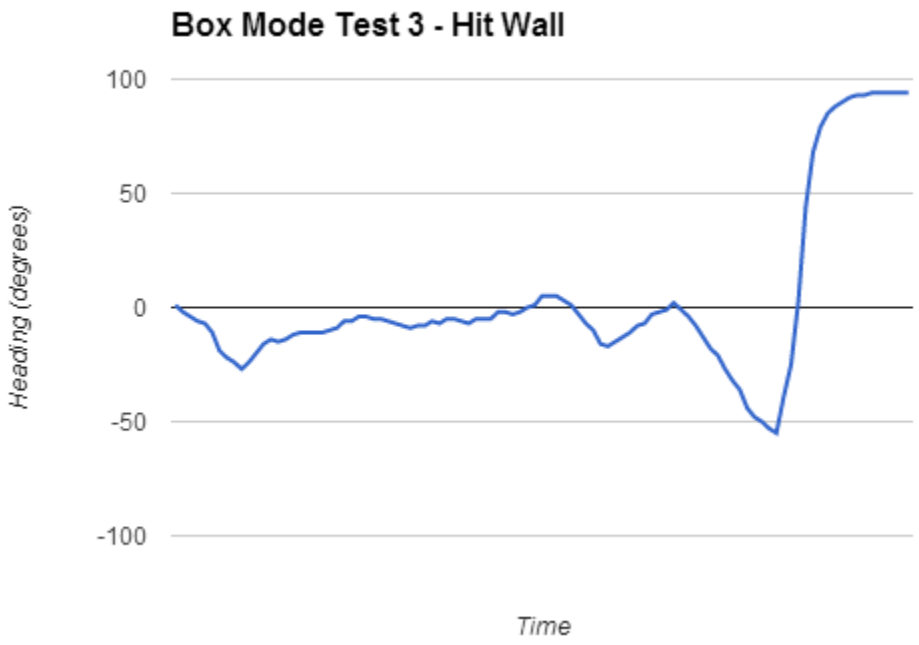
- Good data seen here - quad holds fairly straight, then consistently yaws towards destination heading. Once at desired heading, again holds steady
    - Initially, the quad heading is outside of 10 degree threshold so it corrects back using the feedback via telemetry information.
- Interesting that there aren't more data points in the West direction.
- The team liked these results better than the first one because of the more precise "box" pattern.

A.3.3.4 Box Test Number 3 - Crash into Divider

Notes:

- Quadcopter started off fairly well, but ended up driving more NE than N. Eventually we simply ran out of space and hit the divider wall before the turn had finished executing.

Results:



Summary:

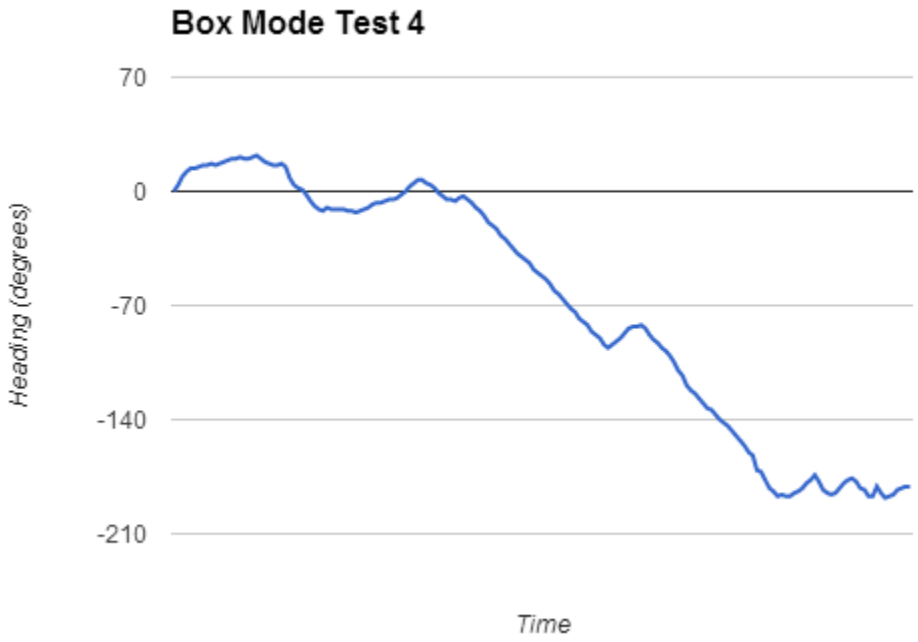
- Quad was consistently below 0, but on the border of the threshold
  - This caused the quad to slightly drift off course
- Quadcopter started pointing north-eastern. The momentum of this initial heading may have resulted in the crash.
  - We simply ran out of room to fly.
- However, you can see the quadcopter begin to execute its western turn when it makes contact with the divider, from which point it came falling down.

A.3.3.5 Test Number 4: +/-10 Degree Heading Window

Notes:

- The quadcopter was started much more northerly to help prevent the crash of the last test.
- The team was very happy with these results.

Results:



Summary:

- Good data seen here - quad holds fairly straight, then consistently yaws towards destination heading. Once at desired heading, again holds steady.
    - The orientation feedback correction can easily be seen in the initial hold of 0 degrees as well as the hold at -180 degrees at the end.
- The small spike in the middle could be a result of the quadcopter reaching -90 degrees, drifting beyond the window, and correcting back to -90 before moving on to the -180 heading.

A.4 RC

A.4.1 Reactions for Following

Name: Robert Hoff, Spencer Olson, Jared Zoodsma
Date: 4/8/2014
Test Number: 1

Item under test: RC values being written by Arduino
Testing Parameters:

- Arduino program Follow_test_4
- Arduino
- GUI
- Pi code
- MultiWii_Telemetry_RevG.py

Notes:

- Full communication test was run and quadcopter was plugged into open-sourced Multiwii GUI in order to view RC inputs
- Quadcopter was moved to various positions in relation to the base station in order to check whether the RC values sent corresponded to the desired action of the quadcopter in order to get it back to the base station
- Example, putting the quadcopter to the north east of the base station would make the pitch go down and the roll go to the left

---

Results:

- The RC values were recorded by screenshot along with associated latitude and longitude differences
- RC values corresponded as expected for all positions
- Screenshots are shown in the screenshots folder and are named their respective latitude and longitude differences

A.4.2 Reactions for Following with Flight

Name: Robert Hoff, Spencer Olson, Jared Zoodsma, Drew Brandsen
Date: 4/8/2014
Test Number: 2

Item under test: RC values being written by Arduino, and associated quadcopter flight
Testing Parameters:

- Arduino
- GUI
- Pi code

Notes:

- All telemetry data was sent in the communication loop
- code was modified to move the coordinates of the base station to ten meters north of the actual position in order to provide safety of those near the base station
- quadcopter was flown and the follow program was enabled, forcing the quadcopter to fly toward the base station

---

Results:

- Initial flights yielded unpredictable flight patterns, but code was modified
- Quadcopter behaved as expected, flying toward the base station from any direction
- Wind was too unpredictable for stable flight, so full program could not be run with moving base station

A.5 Sensors

A.5.1 Barometer

A.5.1.1 Positional Tests

- These tests are to determine best method for acquiring accurate barometer information
- The team is concerned that the prop wash will affect the barometer sensitivity. To counteract this, the team is evaluating using foam and mesh to protect the barometer as well as a control (no barometer protection)
- Initial "eyeball" test was done on 3/13/2014 and it appeared that no barometer cover resulted in the best performance (this test was done with Drew, Spencer, and Jared)

A.5.1.1.1 Positional Test 1

Name: Jared Zoodsma
Date: 3/14/2014
Test Number: 1

Item under test: Ground Test No Prop Wash
Testing Parameters: (Ex. software version "xyz" used)

- Foam sponge, mesh sponge were used to cover the barometer as well as control (no cover)
- Quadcopter was placed on the ground without props running
- information was gathered for 1 minute
- tests were performed in the engineering building
- Telemetry.py (RevF) used on quadcopter
- An altitude of 0.0m was expected

Notes:

1. Possible disturbances may have come from people entering and leaving the building (this could result in an overall pressure change)
2. All 3 options were tested with quadcopter on ground without props running
3. Used last 20 samples from 1 minute of data acquisition

Results:

	Control	**Foam**	**Mesh**
Average	0.17	0.05	-0.12
Median	0.2	0.08	-0.23
Standard Dev.	0.17	0.16	0.33

A.5.1.1.2 Positional Test 2

Name: Jared Zoodsma
Date: 3/14/2014
Test Number: 2

Item under test: Table Test No Prop Wash
Testing Parameters: (Ex. software version "xyz" used)

- Foam sponge, mesh sponge were used to cover the barometer as well as control (no cover)
- Quadcopter was placed on a table (3.25') without props running
- Information was gathered for 1 minute
- Tests were performed in the engineering building
- Telemetry.py (RevF) used on quadcopter

Notes:

- Possible disturbances may have come from people entering and leaving the building (this could result in an overall pressure change)
- All 3 options were tested with quadcopter on ground without props running
- Used last 20 samples from 1 minute of data acquisition
- Communication was initialized on the ground (so it was set as 0) then moved to the table where values were recorded
- An altitude of (1 meter = 3.25') was expected

Results:

	Control	**Foam**	**Mesh**
Average	0.98	1.27	0.37
Median	0.96	1.24	0.34
Standard Dev.	0.12	0.11	0.18

A.5.1.1.3 Positional Test 3

Name: Jared Zoodsma, Drew Brandsen
Date: 3/15/2014
Test Number: 3

Item under test: Ground Test with Prop Wash
Testing Parameters: (Ex. software version "xyz" used)

- Foam sponge, mesh sponge were used to cover the barometer as well as control (no cover)
- Quadcopter was placed on the ground with props running
- Information was gathered for 1 minute
- Tests were performed in the engineering building
- Telemetry.py (RevF) used on quadcopter

Notes:

- Possible disturbances may have come from people entering and leaving the building (this could result in an overall pressure change)
- All 3 options were tested with quadcopter on ground with props running @ 50%
- Used last 50 samples from 1 minute of data acquisition
- An altitude of 0.0m was expected as the quadcopter was initialized on the ground.

Results:

	Control	**Foam**	**Mesh**
Average	0.43	0.17	0.54
Median	0.43	0.16	0.59
Standard Dev.	0.16	0.17	0.27

- Mesh option was deemed a poor option since it showed poor performance in all three evaluation areas and was thrown out at this point

A.5.1.1.4 Positional Test 4

Name: Jared Zoodsma, Drew Brandsen
Date: 3/15/2014
Test Number: 4

Item under test: Table Test with Prop Wash
Testing Parameters: (Ex. software version "xyz" used)

- Foam sponge was used to cover the barometer as well as control (no cover)
- Quadcopter was placed on a table (3.25') with props running
- Information was gathered for 1 minute
- Tests were performed in the engineering building
- Telemetry.py (RevF) used on quadcopter

Notes:

- Possible disturbances may have come from people entering and leaving the building (this could result in an overall pressure change)
- Both options were tested with quadcopter on ground without props @ 50%
- Used last 50 samples from 1 minute of data acquisition
- Communication was initialized on the ground (so it was set as 0) then moved to the table and props were initialized and values were recorded
- An altitude of (1 meter = 3.25') was expected

Results:

	Control	**Foam**
Average	1.11	1.02
Median	1.15	1.04
Standard Dev.	0.29	0.28

A.5.1.1.5 Summary of Positional Tests

Summary:

- Mesh cover was thrown out due to low performance in all metrics

Overall Averages

	Control	**Foam**
Average	0.1725	0.1275
Median	0.1850	0.1300
Standard Dev.	0.1850	0.1800

- At first glance, there seems to be very little difference in performance, if any between the control (no cover) and the foam cover. However, after further inspection it appears that the foam had better performances when prop wash was introduced (see table below)

Overall Averages (w/ Prop Wash)

	Control	**Foam**
Average	0.270	0.095
Median	0.290	0.100
Standard Dev.	0.225	0.225

- Initially standard deviation was weighed the highest of all metrics, however there seemed to be an insignificant difference in standard deviation between control and foam.
- When prop wash was introduced, foam resulted in better averages and medians than the control. Therefore, Foam appears to be the best option for consistent and accurate altitude information

A.5.1.2 Flight Tests

Name: Jared Zoodsma, Drew Brandsen
Date: 3/15/2014
Test Number: 1

Item under test: Quadcopter Barometer - actual flying
Testing Parameters: (Ex. software version "xyz" used)

- Foam sponge was to cover the barometer as well as control (no cover)
- Quadcopter was initialized on the ground
- Information was gathered for 1 minute
- Tests were performed in the engineering building
- Telemetry.py (RevF) used on quadcopter
- An altitude of 2.0m was expected

Notes:

- Possible disturbances may have come from people entering and leaving the building (this could result in an overall pressure change)
- Used last 50 samples (not the last 2 samples) from 1 minute of data acquisition

---

Results:

	Control	**Foam**
Average	0.83	1.90
Median	0.78	1.94
Standard Dev.	0.45	0.31

- Foam has shown to be the better option after the flight test. Using foam results in better standard deviation and significantly better accuracy.

A.5.2 Compass

A.5.2.1 Compass Precision

A.5.2.1.1 Summary of Compass Precision Tests

Name: Drew Brandsen, Jared Zoodsma
Date: 4/13/2014

Item under test: MultiWii Compass
Testing Parameters:

- GUI
- Pi code

Notes:

- Tests were done outside near the garage door of the engineering building
- There was real-life disturbance such as wind
- Sensor data was streamed wirelessly to the GUI to be recorded
- Reported order: Latitude, Longitude, Altitude, Heading, Time

---

Conclusion:

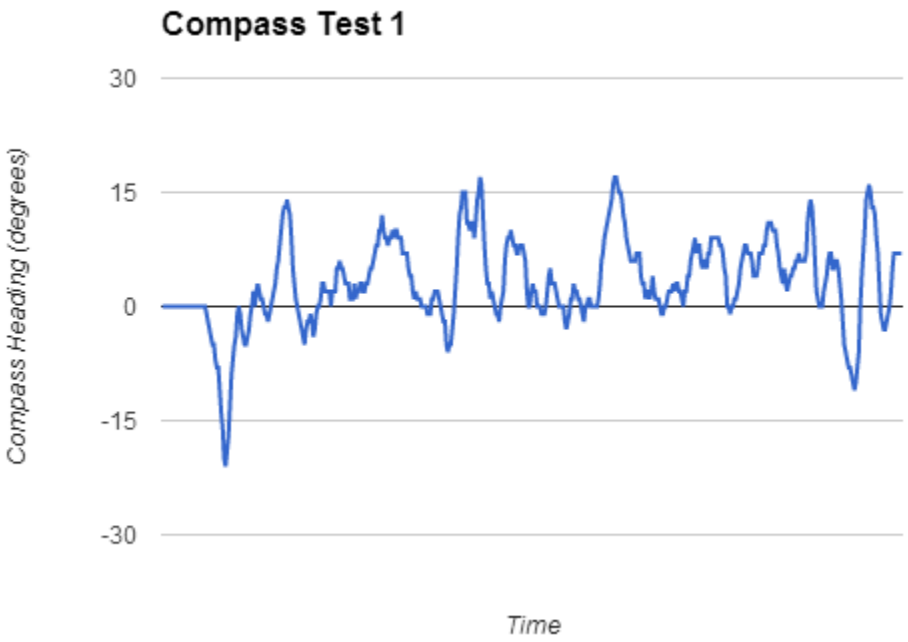
- The EMF problem is solved as the motors no longer have an effect on the compass value.
- The compass data for tests 1, 2, 5, and 6 show an immediate drop to start the test. For some reason, there was a 20-30 degree difference between a "0-heading" on the ground and a "0-heading" in the air. As a result, the quadcopter immediately turned to correct for this change.
    - It seems that the ground had some effect on the compass data which caused this problem.
- The quadcopter flew well with both barometer and heading modes activated. While the data shows that the flight modes are working, there is nothing in software that the team can change to improve the results (all onboard MultiWii software).
    - To improve the results the team would have to look to improved hardware

A.5.2.1.2 Compass Test Number 1 - Centered at Zero, Takeoff with Mag Mode On

Notes:

- The quadcopter was started facing a 0 degree heading before taking off. Mag Mode was always activated.

Results:



Average	3.2
Max	17.0
Min	-21.0
Standard Deviation	5.7

Summary:

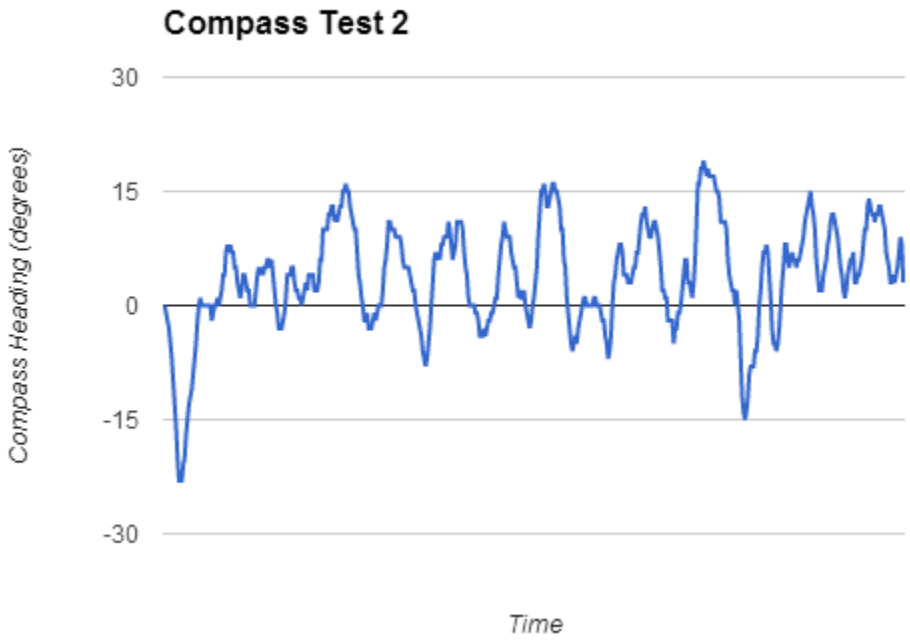
- Though the data varied a fair amount in either direction, the quadcopter always recovered to a "near-zero" heading
- The team was encouraged by the results; however, we would have liked to see extreme data points closer to 0

A.5.2.1.3 Compass Test Number 2 - Centered at Zero, Takeoff with Mag Mode On

Notes:

- The quadcopter was started facing a 0 degree heading before taking off. Mag Mode was always activated.

Results:



Average	3.8
Max	19.0
Min	-23.0
Standard Deviation	7.0

Summary:

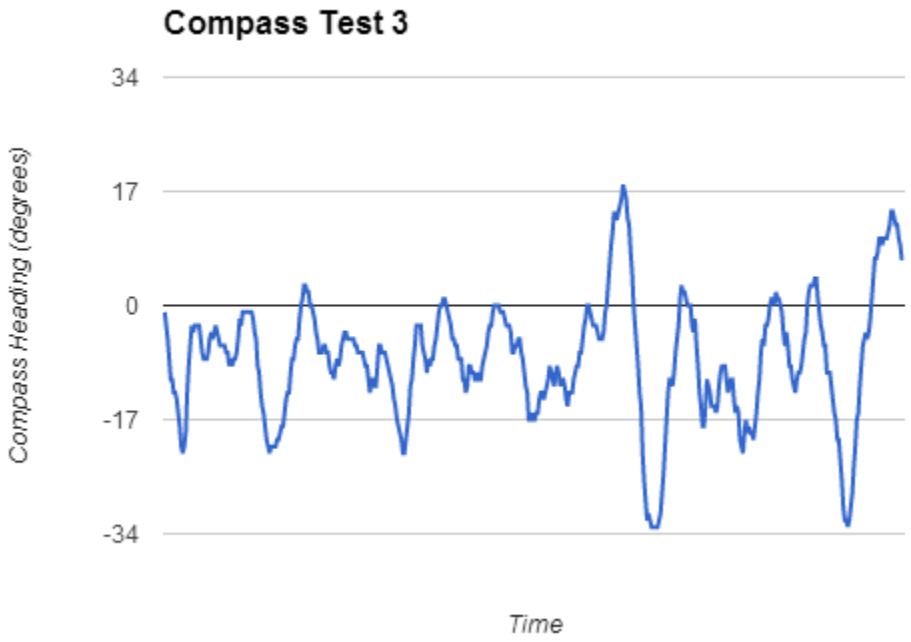
- Though the data varied a fair amount in either direction, the quadcopter always recovered to a "near-zero" heading
- The team was encouraged by the results; however, we would have liked to see extreme data points closer to 0.
- This test had a higher standard deviation than the previous test.

A.5.2.1.4 Compass Test Number 3 - Fly to Zero Heading, then turn on mag mode

Notes:

- The quadcopter took off and was "yaw-ed" to a 0 degree heading. At that point, mag mode was activated until landing.

Results:



Average	-7.9
Max	18.0
Min	-33.0
Standard Deviation	9.2

Summary:

- It appears that Test 3 had worse results than the previous two tests. One likely cause of this is that it was very hard to activate Mag Mode exactly when the heading was zero.
- This means that Mag Hold was likely activated around -5 degrees, which would yield a much better average heading.

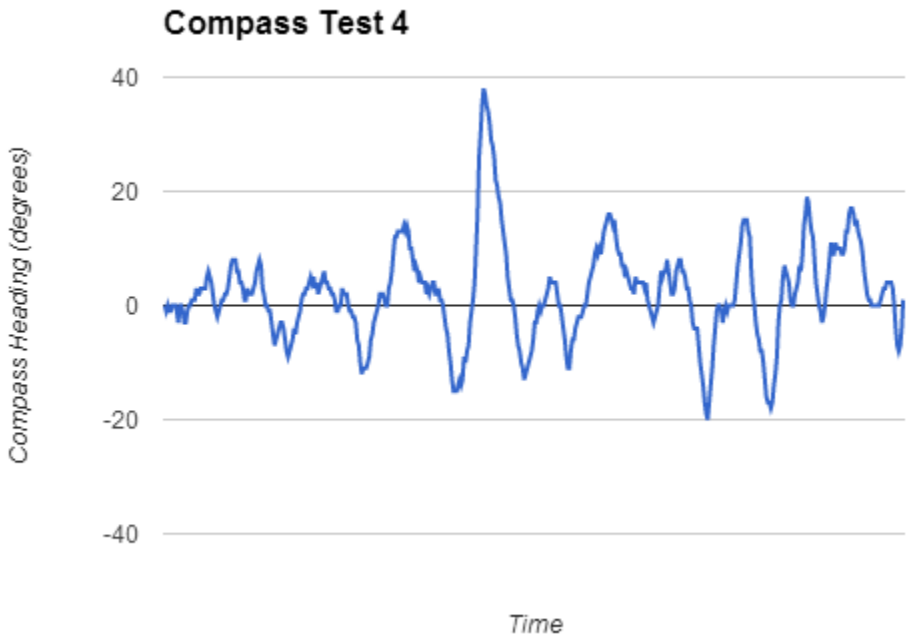
A.5.2.1.5 Test Number 4 - Fly to zero heading, then turn on mag mode

---

Notes:

- The quadcopter took off and was "yaw-ed" to a 0 degree heading. At that point, mag mode was activated until landing.

Results:



Average	2.2
Max	38.0
Min	-20.0
Standard Deviation	8.6

Summary:

- Again it appears that Test 4 had worse results than Tests 1 and 2. One likely cause of this is that it was very hard to activate Mag Mode exactly when the heading was zero.
- This means that Mag Hold was likely activated around 3 degrees, which would yield a much better average heading.
- This test also had one of the highest extreme values, but once again the quadcopter responded by bringing the heading back to near-zero.

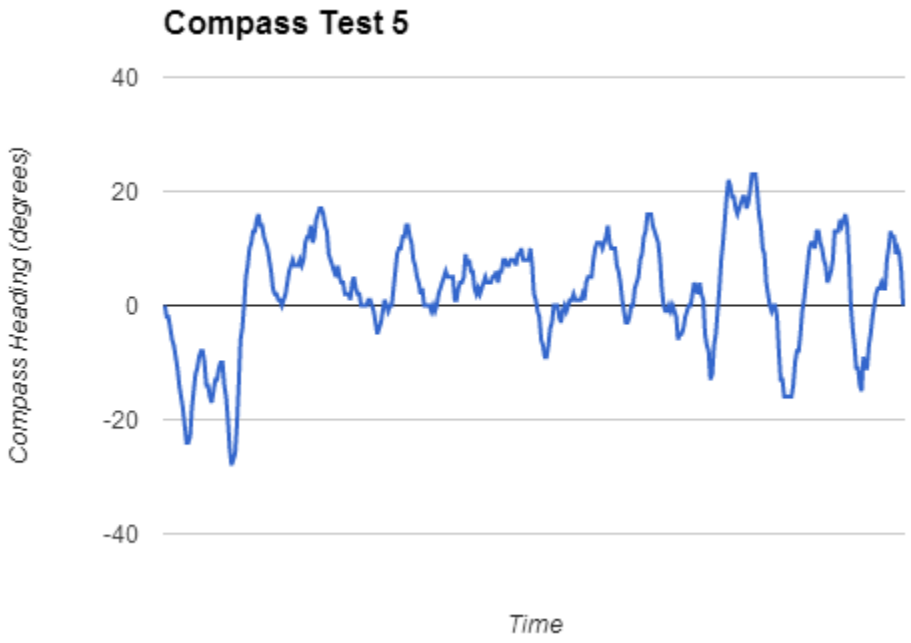
A.5.2.1.6 Compass Test Number 5

Notes:

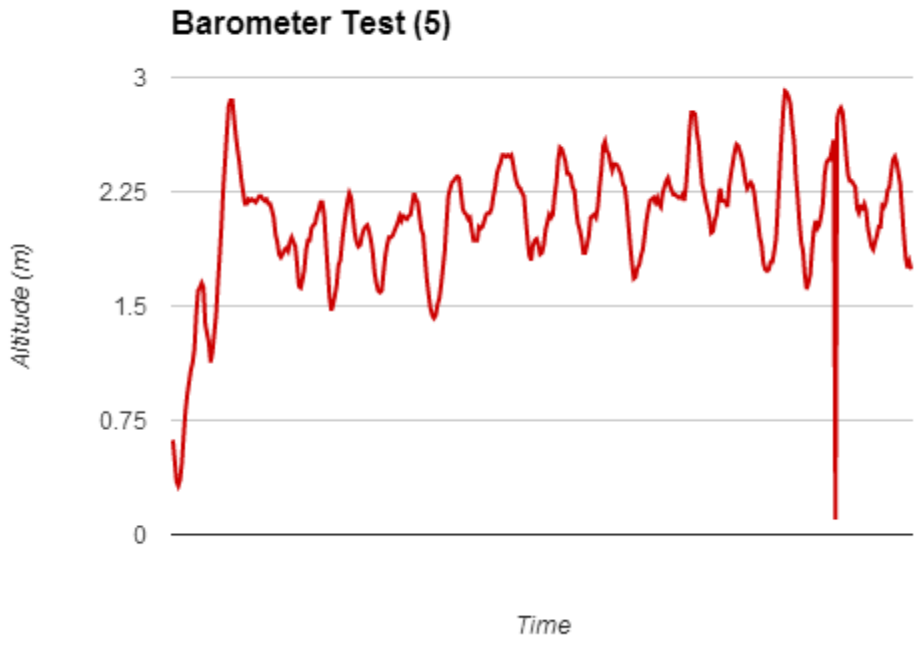
- The quadcopter was started facing a 0 degree heading before taking off. Mag Mode was always activated.
- Baro mode was activated at roughly 2 meters to see how well altitude was held while mag mode was also being held

---

Results:



Average	2.6
Max	23.0
Min	-28.0
Standard Deviation	9.4



Average	2.1
Max	2.9
Min	0.1
Standard Deviation	0.4

Summary:

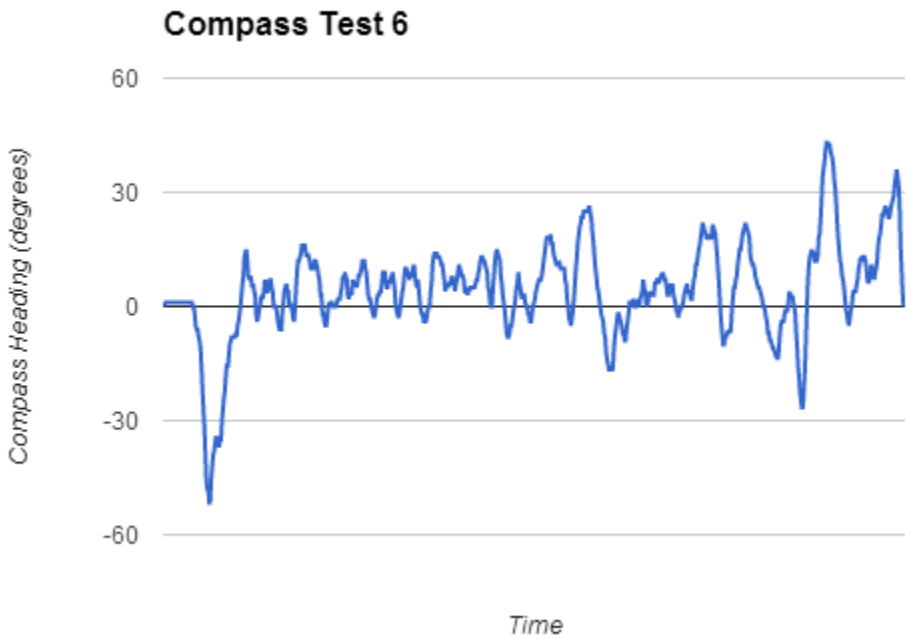
- Compass data was again similar to the previous 4 tests. There was a good average but again a poor standard deviation
- Barometer data looks very good with a standard deviation of 0.4. There is the one outlier near the end of the test that is bad.
    - The problem with barometer is that it is only relative to air pressure (which can change quickly), so what may look consistent in the data was actually much worse flight performance relative to the ground.

A.5.2.1.7 Compass Test Number 6

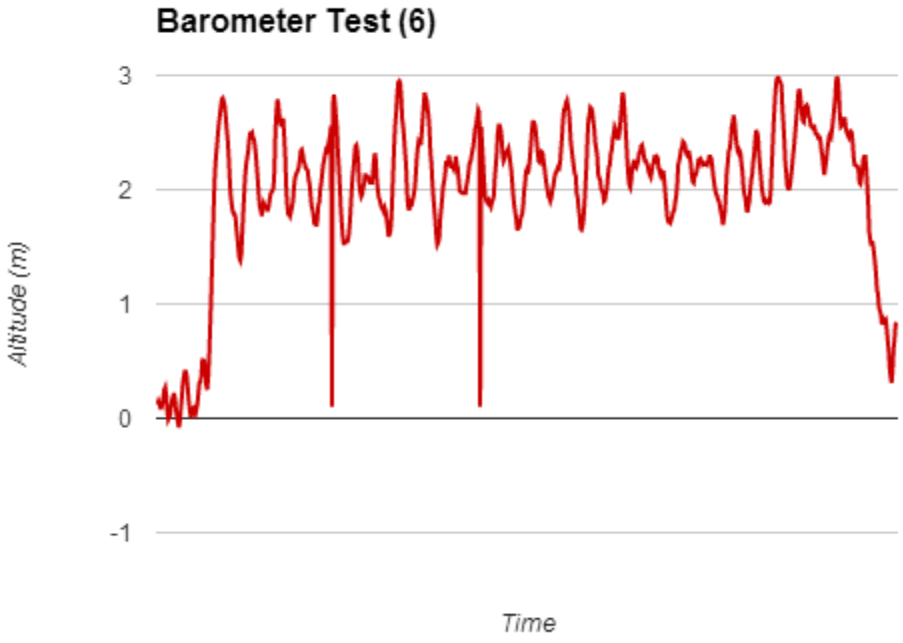
Notes:

- The quadcopter was started facing a 0 degree heading before taking off. Mag Mode was always activated.
- Baro mode was activated at roughly 2 meters to see how well altitude was held while mag mode was also being held

Results:



Average	4.3
Max	43.0
Min	-52.0
Standard Deviation	12.6



Average	2.0
Max	3.0
Min	-0.1
Standard Deviation	0.7

Summary:

- Compass data was again similar to the previous 4 tests. There was a good average but again a poor standard deviation. Only at the beginning was the data poor
- Barometer data looks good again with a standard deviation of 0.7. There are two outlier near the middle of the test that is bad.
    - The problem with barometer is that it is only relative to air pressure (which can change quickly), so what may look consistent in the data was actually much worse flight performance relative to the ground.

A.5.2.2 EMI Tests

The purpose of these tests is to determine the effectiveness of shielding the ESCs and how to shield the flight controller.

A.5.2.2.1 EMI Test 1

Name: Drew Brandsen, Jared Zoodsma
Date: 3/17/2014
Test Number: 1

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- ESCs (not wires) wrapped in aluminum foil
- ESCs "dangling" (not taped close to internals of quadcopter)
- Test was done at table by wind tunnel

Notes:

- Raspberry Pi was powered during these tests

Results:

- Compass was started at -100.0 degrees
- -103 degrees at full throttle
- -102 degrees at half throttle
- Compass was skewed 3 degrees when throttle was at full power
- Compass was skewed 2 degrees when throttle was at half power

A.5.2.2.2 EMI Test 2

Name: Drew Brandsen, Jared Zoodsma
Date: 3/17/2014
Test Number: 2

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- ESCs (not wires) wrapped in aluminum foil
- ESCs taped close to central part of quadcopter
- Test was done at table by wind tunnel

Notes:

- Raspberry Pi was powered during these tests

Results:

- Compass was started at -100.0 degrees
- -102 degrees at full throttle
- -102 degrees at half throttle
- Compass was skewed 2 degrees when throttle was at full power
- Compass was skewed 2 degrees when throttle was at half power

Initial Summary:

- Aluminum foil wrapping significantly reduces compass EMI to an acceptable skew level (2 degrees)

Further Summary

- After a few test flights, it was noticed that the magnetometer slowly degraded to an error of around 45 degrees (roughly the same as the control case).
- Additional foil was added and that seemed to correct the problem. But once again, test flights proved the degradation of the magnetometer again.

A.5.2.2.3 EMI Test 3

Name: Drew Brandsen
Date: 3/17/2014
Test Number: 3

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- ESCs not wrapped
- ESCs dangling
- Test was done at table by wind tunnel

Notes:

- Raspberry Pi was powered during these tests

Results:

- Compass was started at -180.0 degrees
- -163 degrees at full throttle
- -165 degrees at half throttle
- Compass was skewed 17 degrees when throttle was at full power
- Compass was skewed 15 degrees when throttle was at half power

A.5.2.2.4 EMI Test 4

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 4

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- mechanical connections loose
- Raspberry Pi case not on quadcopter

Notes:

- This is a control test

Results:

- Initial position: -10 degrees
- Full throttle: -55 degrees

A.5.2.2.5 EMI Test 5

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 5

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- mechanical connections loose
- Raspberry Pi case is on quadcopter

Results:

- Initial position: -10 degrees
- Full throttle: -55 degrees

A.5.2.2.6 EMI Test 6

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 6

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- mechanical connections loose
- Raspberry Pi case not on quadcopter

Notes:

- Flight Controller moved up ¾”

Results:

- initial position: -4 degrees
- full throttle: -14 degrees

A.5.2.2.7 EMI Test 7

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 7

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- Mechanical connections loose
- Raspberry Pi case not on quadcopter

Notes:

- 5.5X6.5” aluminum foil added directly below flight controller
  - aluminum foil is grounded
  - 4 screw holes located in the aluminum foil
- Flight control sitting on quadcopter (no spacers added)

Results:

- Initial position: -7 degrees
- Full throttle: -48 degrees

A.5.2.2.8 EMI Test 8

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 8

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- Mechanical connections loose
- Raspberry Pi case not on quadcopter

Notes:

- 5.5X6.5" aluminum foil added directly below flight controller
  - aluminum foil is grounded
- Flight control sitting on quadcopter (no spacers added)

Results:

- Initial position: -16 degrees
- Full throttle: -56 degrees

A.5.2.2.9 EMI Test 9

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 9

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- Mechanical connections loose
- Raspberry Pi case not on quadcopter

Notes:

- 5.5X6” grounded steel plate placed directly below flight controller
- Flight control sitting on quadcopter (no spacers added)

Results:

- Initial position: -165 degrees
- Full throttle: -169 degrees

A.5.2.2.10 EMI Test 10

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 10

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- Mechanical connections loose
- Raspberry Pi case not on quadcopter

Notes:

- 5.5X6” grounded thin steel sheet placed below flight controller
- Flight control sitting on quadcopter (no spacers added)

Results:

- Initial position: -16 degrees
- Full throttle: -18 degrees

Summary:

- It appears adding distance and steel (not aluminum) helps prevent compass skew.
- Note: These tests only examined initial skew and did not investigate the change in skew over time.

A.5.2.2.11 Effect of Power Supply Test 1

The purpose of these tests is to determine the source of compass skew.

Name: Drew Brandsen, Jared Zoodsma
Date: 3/17/2014
Test Number: 1

Item under test: Compass (Control Case)
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place

---

Notes:

- Raspberry Pi was powered during these tests

Results:

- Compass was started at -180 degrees
- -156 degrees at full throttle
- -160 degrees at half throttle
- Compass was skewed 34 degrees when throttle was at full power
- Compass was skewed 20 degrees when throttle was at half power

A.5.2.2.12 Effect of Power Supply Test 2

Name: Drew Brandsen, Jared Zoodsma
Date: 3/17/2014
Test Number: 2

Item under test: Compass
Testing Parameters:

- MultiWii is powered by a separate power supply
- Power distribution-sized ground plate is currently in place

---

Notes:

- Raspberry Pi was powered during these tests

Results:

- Compass was started at -180 degrees
- -155 degrees at full throttle
- -160 degrees at half throttle
- Compass was skewed 35 degrees when throttle was at full power
- Compass was skewed 20 degrees when throttle was at half power

A.5.2.2.13 Skew over Time Test 1

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 1

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- Mechanical connections loose

Notes:

- Raspberry Pi case placed under the flight controller
  - Flight Controller is 1" above quadcopter plate
- 4 degrees will be recorded
  - initial (no motors)
  - immediately after motors are turned on
  - 30 seconds after motors have been on
  - 1 minute after motors have been on

Results:

- Initial position: -17 degrees
- Full throttle: -21 degrees
- After 30 seconds: -21 degrees
- After 1 minute: -21 degrees

A.5.2.2.14 Skew over Time Test 2

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 2

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- mechanical connections loose

Notes:

- Flight Controller placed directly on a grounded steel plate

---

Results:

- Initial position: -24 degrees
- Full throttle: -27 degrees
- After 30 seconds: -27 degrees
- After 1 minute: -27 degrees
- -26 after 1:30
- -25 after 2:00
- -25 after 2:30
- -25 after 3:00
- -25 after 3:30

A.5.2.2.15 Skew over Time Test 3

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 3

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- mechanical connections loose

Notes:

- Flight Controller placed directly on an ungrounded steel plate

---

Results:

- initial position: -23 degrees
- full throttle: -25 degrees
- after 30 seconds: -26 degrees
- after 1 minute: -27 degrees
- -27 degrees after 1:30
- -27 degrees after 2:00
- -27 degrees after 2:30
- -27 degrees after 3:00
- -27 degrees after 3:30

A.5.2.2.16 Skew over Time Test 4

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 4

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- mechanical connections loose

Notes:

- Flight Controller placed directly on a grounded steel plate
- After Throttle is established, it will be varied throughout the test

---

Results:

- initial position: -29 degrees
- full throttle: -31 degrees
- after 30 seconds: -32 degrees
- after 1 minute: -32 degrees
- -32 degrees after 1:30

A.5.2.2.17 Skew over Time Test 5

Name: Drew Brandsen, Jared Zoodsma
Date: 3/18/2014
Test Number: 5

Item under test: Compass
Testing Parameters:

- MultiWii is powered by the power distribution board
- Power distribution-sized ground plate is currently in place
- mechanical connections loose

Notes:

- Flight Controller placed directly on an ungrounded steel plate
- After Throttle is established, it will be varied throughout the test

---

Results:

- initial position: -31 degrees
- full throttle: -32 degrees
- after 30 seconds: -33 degrees
- after 1 minute: -33 degrees
- -33 degrees after 1:30

A.5.3 GPS

A.5.3.1 1 Hz Precision

A.5.3.1.1 Summary of 1 Hz GPS Precision

Name: Drew Brandsen, Jared Zoodsma
Date: 4/2/2014

Item under test: GPS accuracy
Testing Parameters:

- GUI code
- Adafruit GPS on Arduino (no external antenna)
- MediaTek GPS on Pi

Notes:

- Arduino and Pi both placed in the same car (~1ft from each other)
- No antennas used on either GPS modules

A.5.3.1.2 1 Hz Precision Test 1

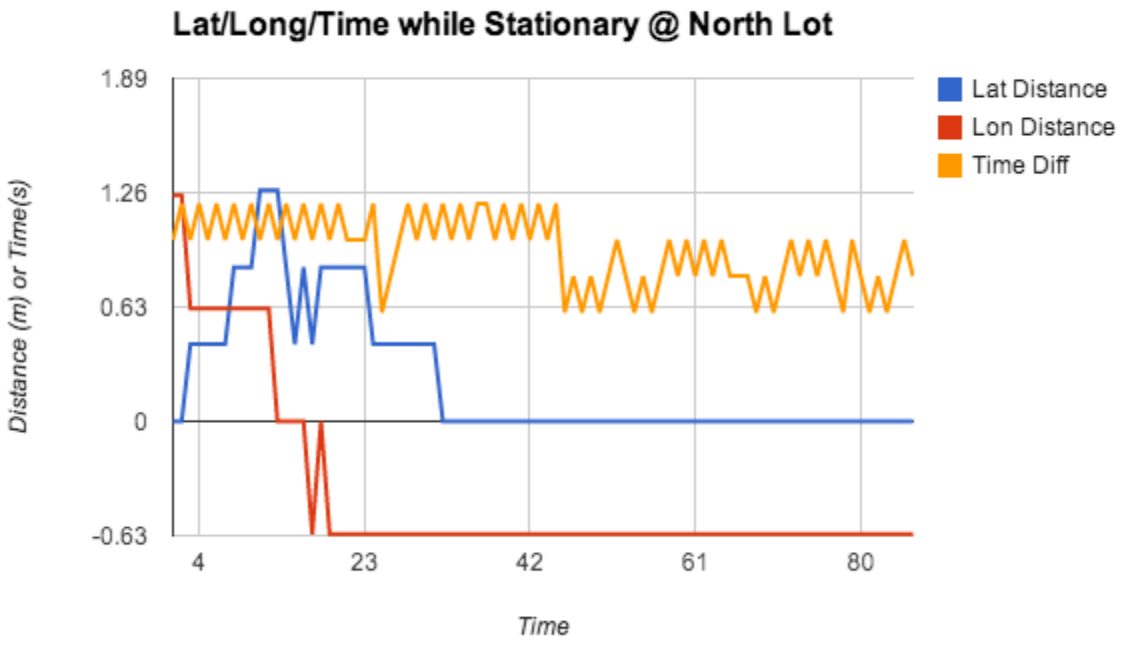
Test Number: 1

Notes:

- Car was stationary in the parking lot by the soccer fields
- Bad Data: 3 erroneous distances points, 1 erroneous time point

---

Results:



Average abs(Lat):	0.2196	**sd Lat:**	0.3563
Average abs(Long):	0.6006	**sd Long:**	0.4884
Average Time:	0.9459	**sd Time:**	0.1961

Summary:

- We were pleased with these results since distances were within GPS margin of error
- Time was never more than 1.26 seconds apart; average was 0.96s\
- We believe that the few bad distance data points are coming from the Adafruit GPS sending occasional blank data.
- We are unsure what is causing the few bad time data points seen throughout the tests.

A.5.3.1.3 1 Hz Precision Test 2

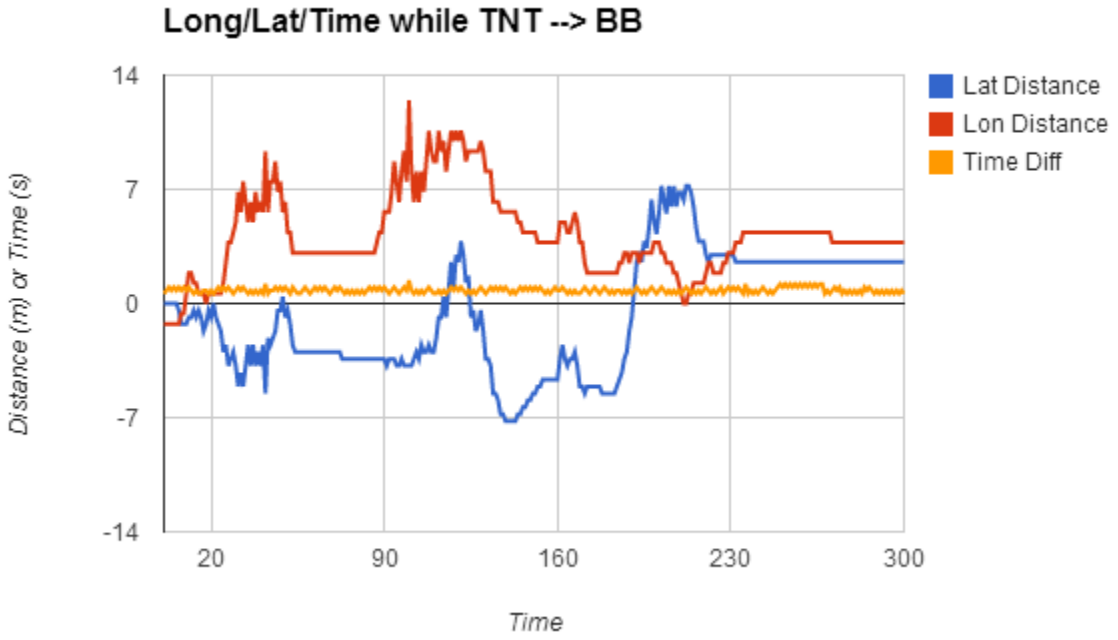
Test Number: 2

Notes:

- Car was moving from North Parking lot to BB counter-clockwise (~15mph)
- Bad Data: 11 erroneous distances points, 5 erroneous time points

---

Results:



Average abs(Lat):	3.2404	**sd Lat:**	3.5821
Average abs(Long):	4.2377	**sd Long:**	2.5401
Average Time:	0.8080	**sd Time:**	0.1666

Summary:

- The navigation path was along a curve, so we anticipated discrepancies along both lat and long.
- Time was still very consistent
- Location accuracy was not consistent, but neither was speed (varying from stop to 20 mph)
- Moving at 15mph (average), a 1 second delay could cause up to a 6.7 meters discrepancy.
    - Therefore, we are pleased the results were, on average, within this threshold
    - We wish that the gps data would have converged more at the end of the test when the vehicle was stopped, however.

A.5.3.1.4 1 Hz Precision Test 3

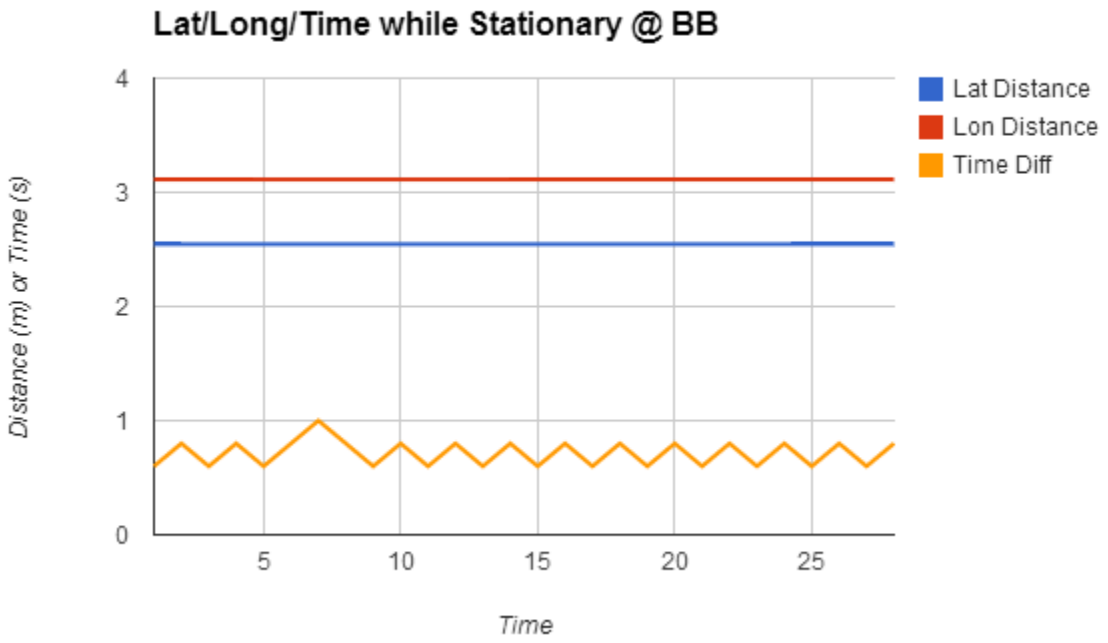
Test Number: 3

Notes:

- Car was stationary in BB parking lot
- Bad Data: 2 erroneous distances points, 0 erroneous time points

---

Results:



Average abs(Lat):	2.5459	**sd Lat:**	0.0000
Average abs(Long):	3.1129	**sd Long:**	0.0000
Average Time:	0.7143	**sd Time:**	0.1145

Summary:

- Car was located under trees and by a dorm
  - This could have resulted in poor signal strength
- GPS data was getting received (as indicated by changing time) but both GPS's held their exact location throughout the test.

A.5.3.1.5 1 Hz Precision Test 4

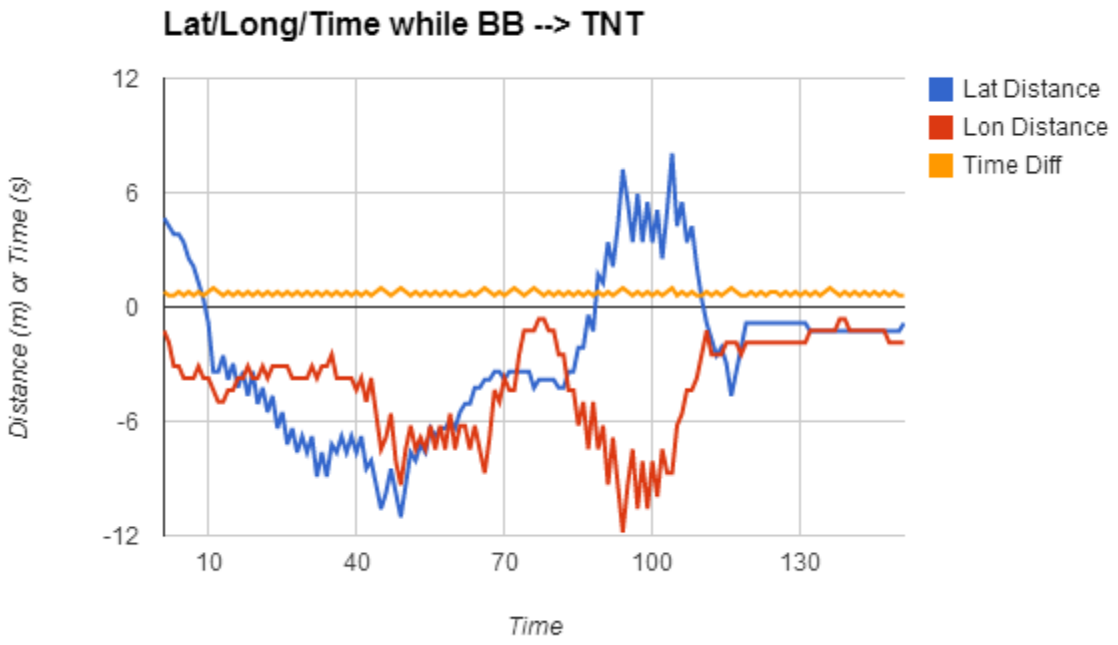
Test Number: 4

Notes:

- Car moved from BB to north parking lot clockwise (~15 mph)
- Bad Data: 4 erroneous distances points, 3 erroneous time points

---

Results:



Average abs(Lat):	4.0184	**sd Lat:**	4.0950
Average abs(Long):	4.1395	**sd Long:**	2.5752
Average Time:	0.7232	**sd Time:**	0.1219

Summary:

- Moving at 15mph, a 1 second delay could lead to 6.7meters discrepancy.
  - Therefore, we are pleased the results were, on average, within this threshold
  - However, there were several peaks well outside this range that could be improved.

A.5.3.1.6 1 Hz Precision Test 5

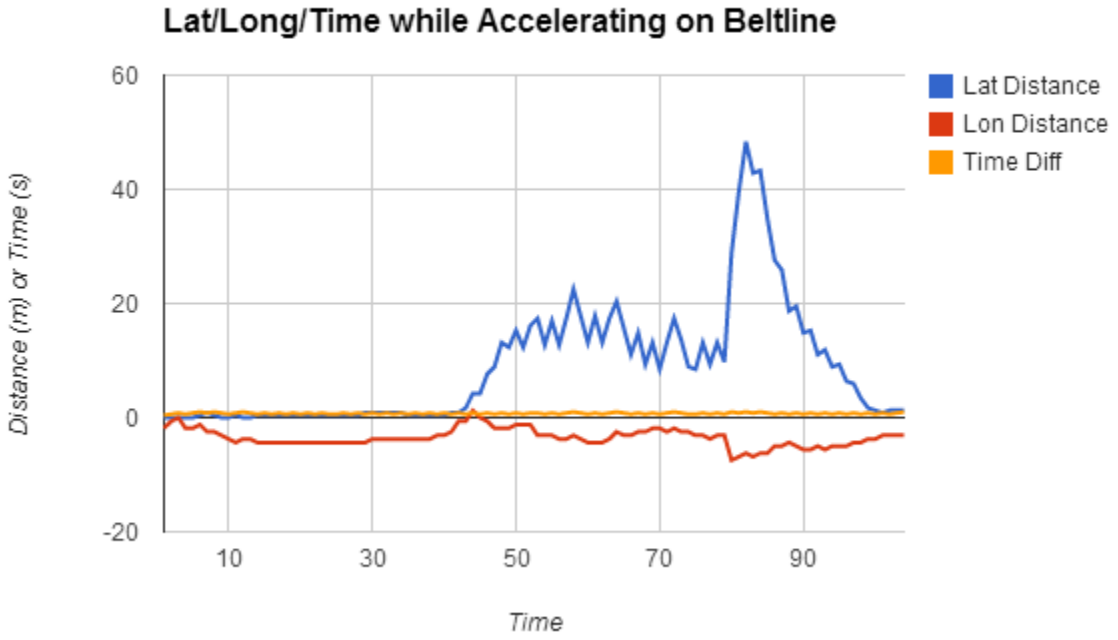
Test Number: 5

Notes:

- Car drove Lake St to Burton on the E. Beltline with acceleration (max speed ~50mph)
- Bad Data: 3 erroneous distances points, 1 erroneous time point

---

Results:



Average abs(Lat):	8.8128	**sd Lat:**	10.5910
Average abs(Long):	3.4960	**sd Long:**	1.5365
Average Time:	0.7327	**sd Time:**	0.1325

Summary:

- Moving at 45 mph, a 1 second delay could lead to 20.1meters discrepancy.
- The beltline is a North-South road, so only Latitude should see much change, which is roughly what we see above.
- On average, the data was well within the possible discrepancy
    - It's possible that as the car went under the Calvin crossing that this is what gave us the spike in data near time 80.
- The data also converged well at the end of the test.

A.5.3.1.7 1 Hz Precision Test 6

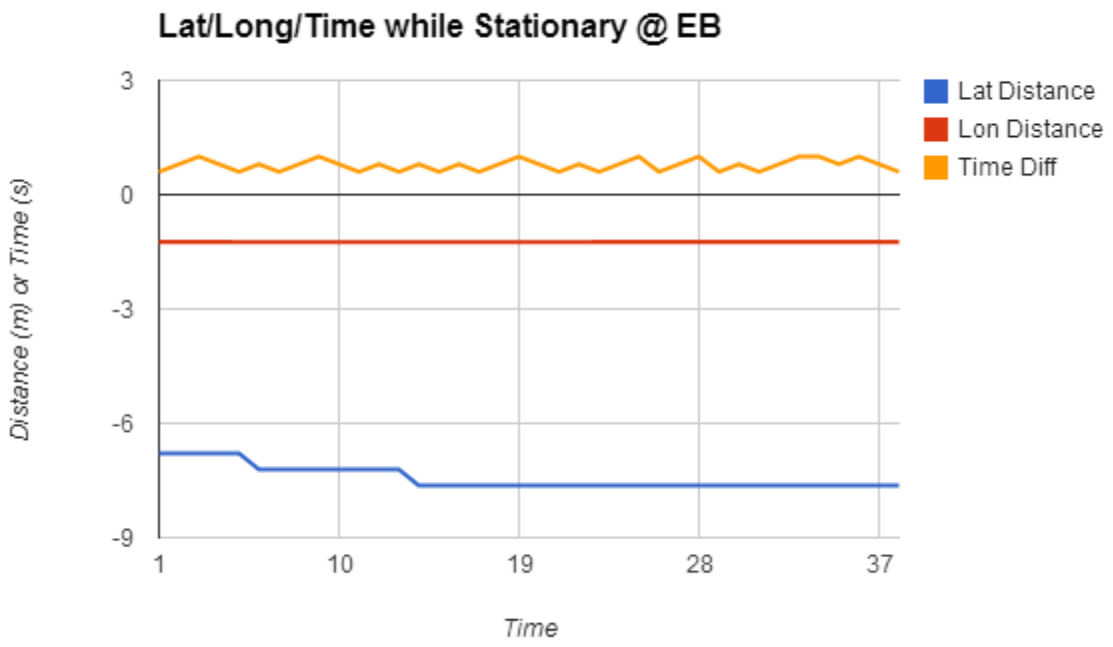
Test Number: 6

Notes:

- Car was stationary in EB parking lot
- Bad Data: 2 erroneous distances points, 1 erroneous time point

---

Results:



Average abs(Lat):	7.4368	**sd Lat:**	0.3078
Average abs(Long):	1.2452	**sd Long:**	0.0000
Average Time:	0.7737	**sd Time:**	0.1483

Summary:

- Time was very consistent and averaged 0.77 seconds.
- Latitude resulted in poor accuracy.
    - However, we were located near the engineering building. This could have resulted in poor signal fidelity.

A.5.3.2 10 Hz Precision

A.5.3.2.1 10 Hz Precision Summary

Name: Drew Brandsen, Jared Zoodsma
Date: 4/8/2014

Item under test: GPS Precision
Testing Parameters:

- Telemetry Code
- 10 Hz GPS refresh rate
- GUI code running on Gateway
- Arduino code

Notes:

- Arduino and Pi placed in same car (~6 inches apart)
- Short GPS antenna used on base station
- **Thanks to **Spencer Olson** for helping with post-test analysis in Excel

---

Conclusion:

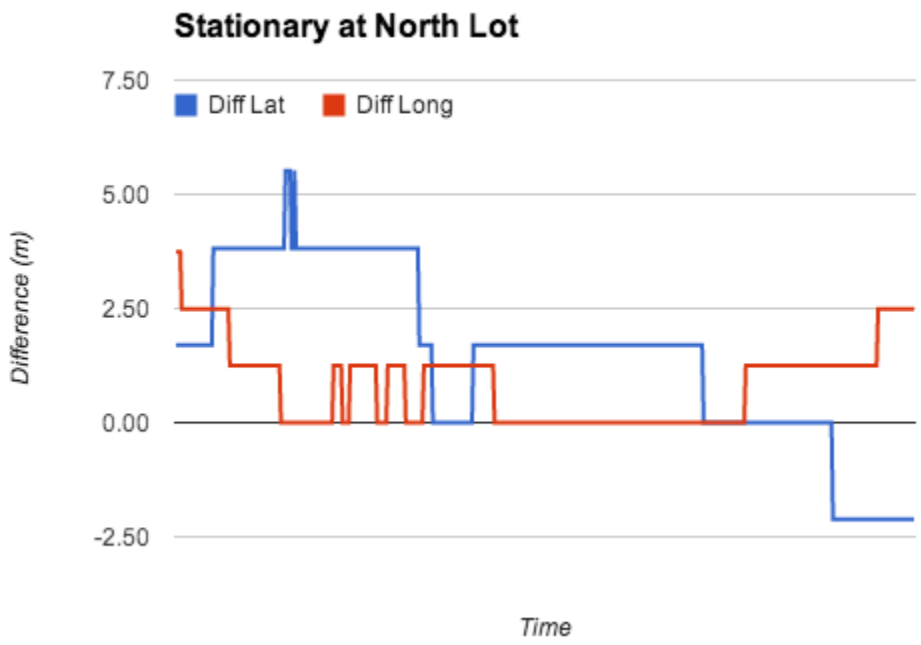
- GPS precision is far improved with increased refresh rates.
- The randomly bad cases of Latitude data need to be investigated.
- The poor data around the BB parking lot may be due to the number of trees and buildings in that area.
- It seems that precision increases with the speed that the car with the GPS modules is moving.

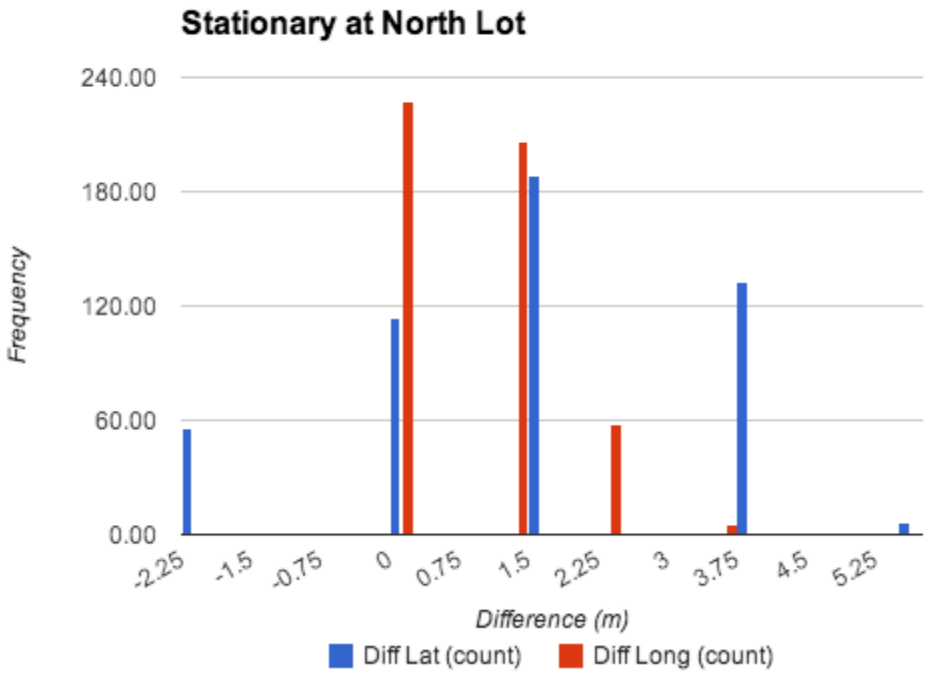
A.5.3.2.2 10Hz GPS Test Number 1 - Stationary at North Lot

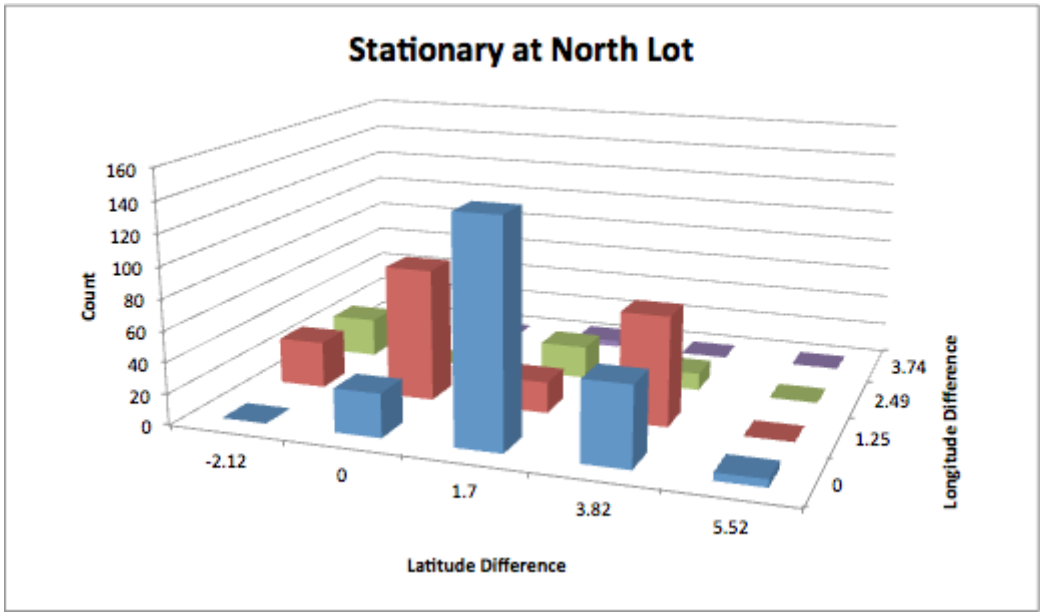
Notes:

- Sitting at north parking lot by track
- started at 10:40

Results:







	Time	Latitude	Longitude
Average	0.20	1.96	0.84
Standard Deviation	0.04	1.40	0.88

Data within 7.2 meter radius: 100%

Summary:

- We were pleased with the stationary results (100% within 6m radius)

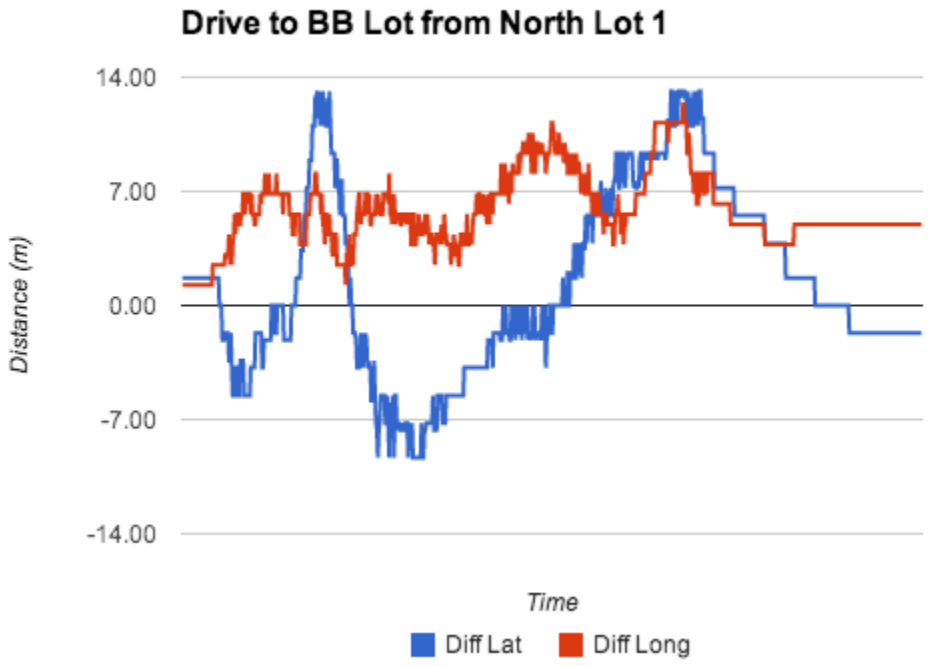
Results from this test confirmed our suspicion that GPS locations were being reported in discrete positional locations

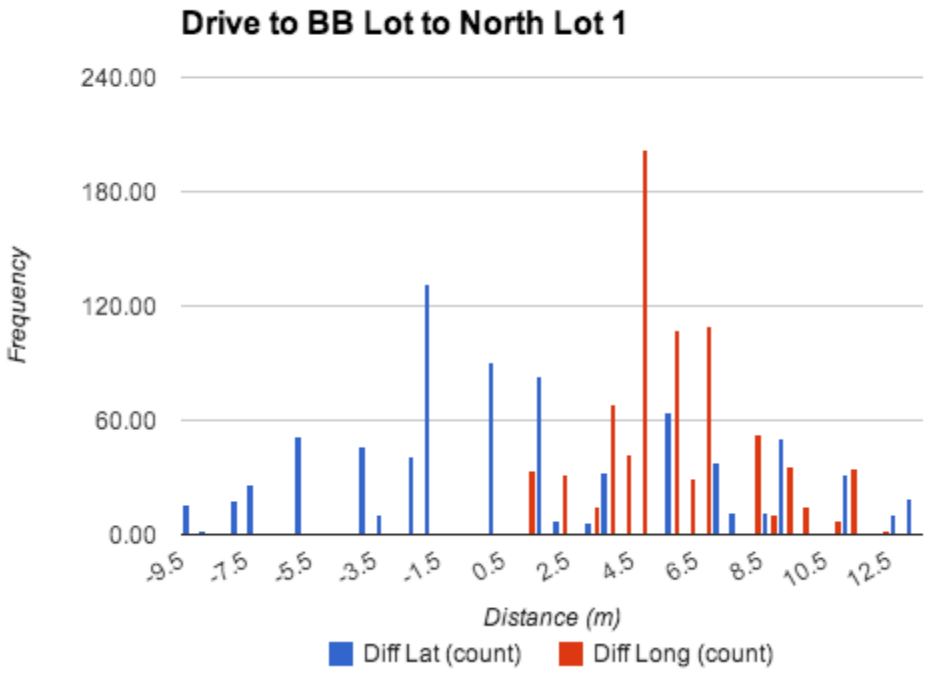
A.5.3.2.3 **Test Number** 2 - Drive to BB Lot from North Lot 1

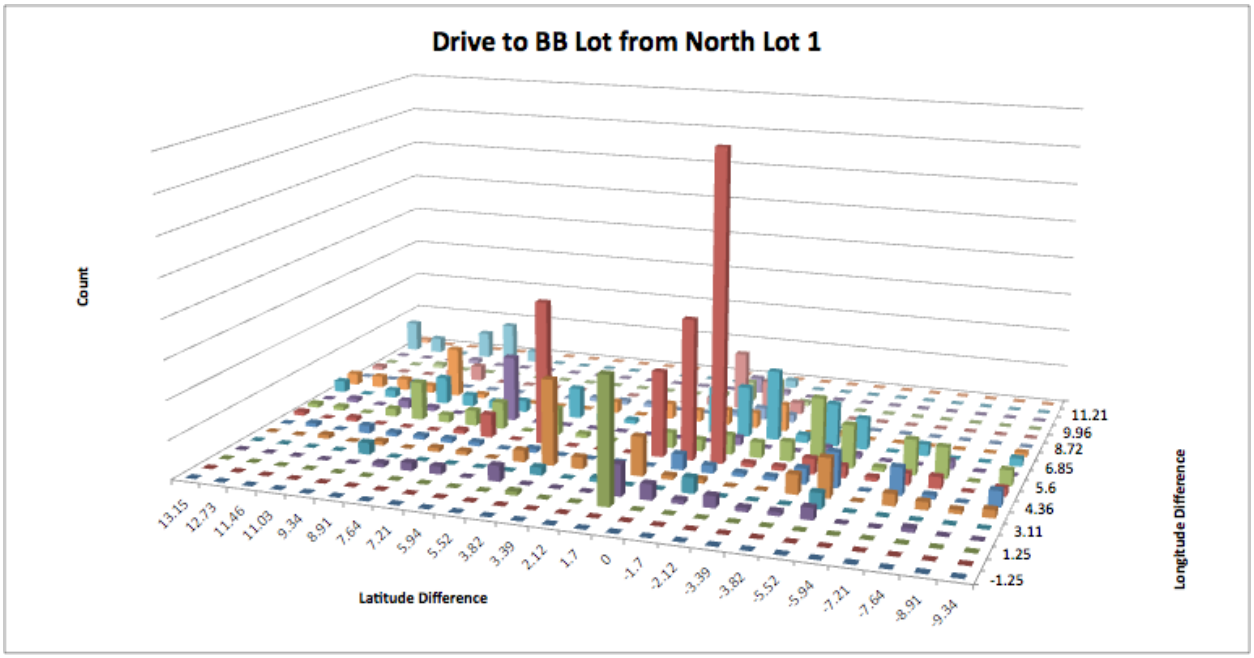
Notes:

- Drive to BB Lot from North Lot 1
- Test started at 10:45
- Speed around 20 mph
- Car stopped around 10:47:40

Results:







	Time	Latitude	Longitude
Average	0.2	4.52	5.80
Standard Deviation	0.05	3.55	2.26

Data within 7.2 meter radius: 60%

Summary:

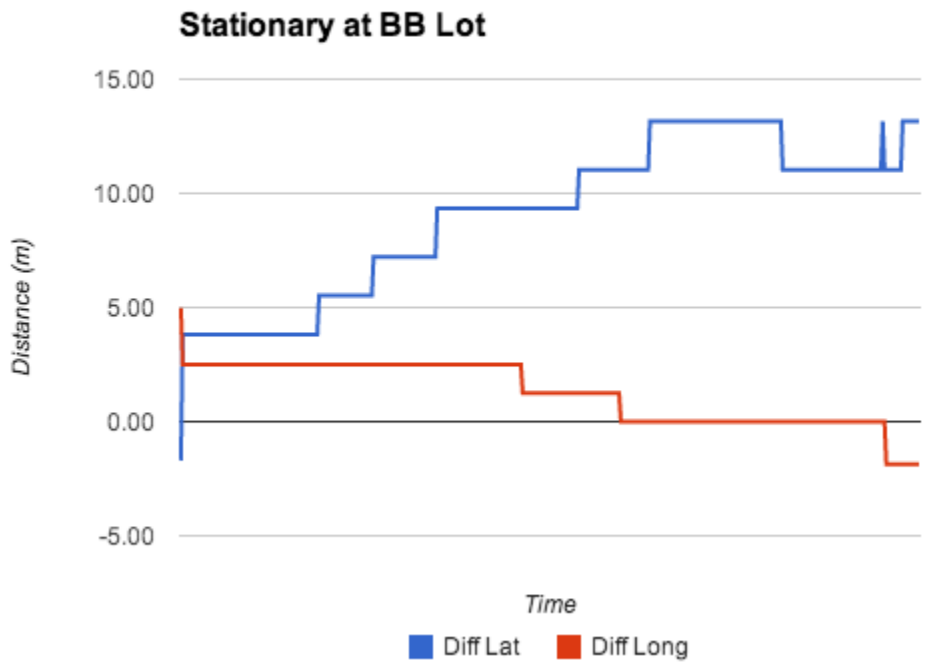
- Compared with the first test, GPS standard deviations increased and precision decreased
- 60% of data was within the expected area.

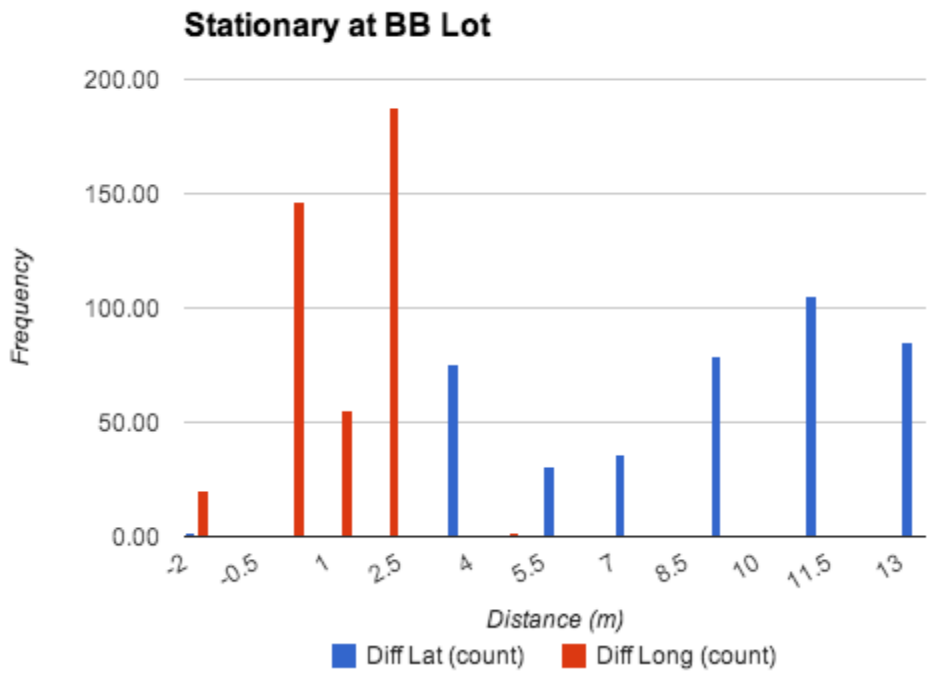
A.5.3.2.4 Test Number 3 - Stationary at BB Lot

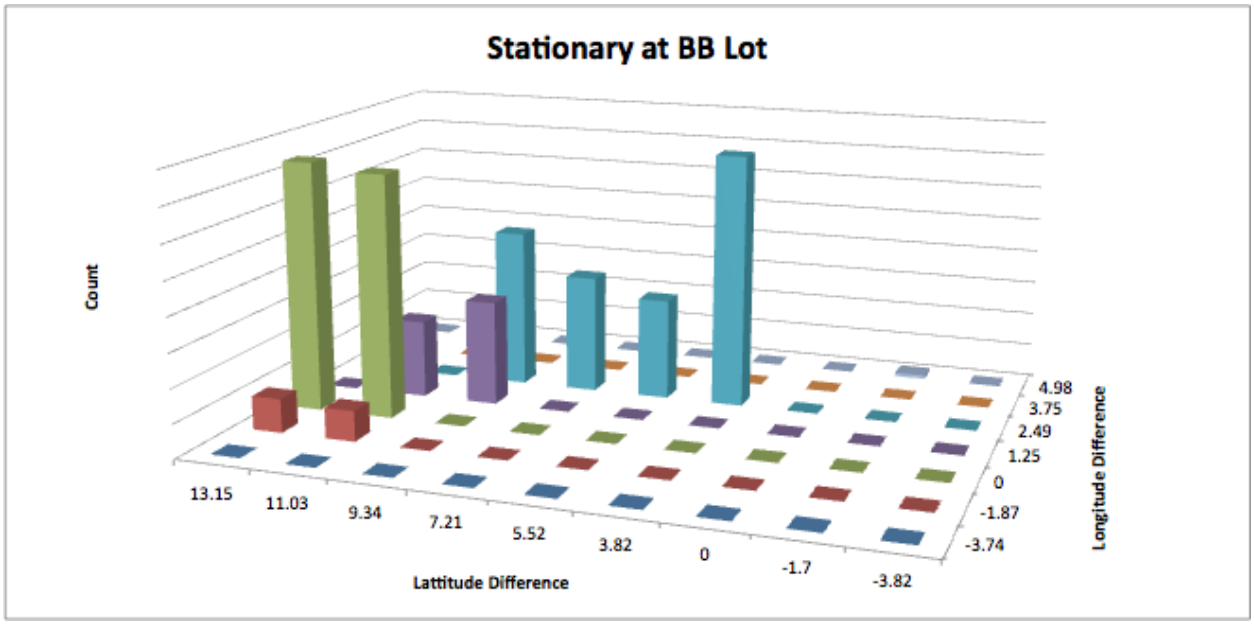
Notes:

- Stationary at BB parking lot
- started at 10:49

Results:







	Time	Latitude	Longitude
Average	0.2	9.07	1.40
Standard Deviation	0.05	3.28	1.13

Data within 7.2 meter radius: 26%

Summary:

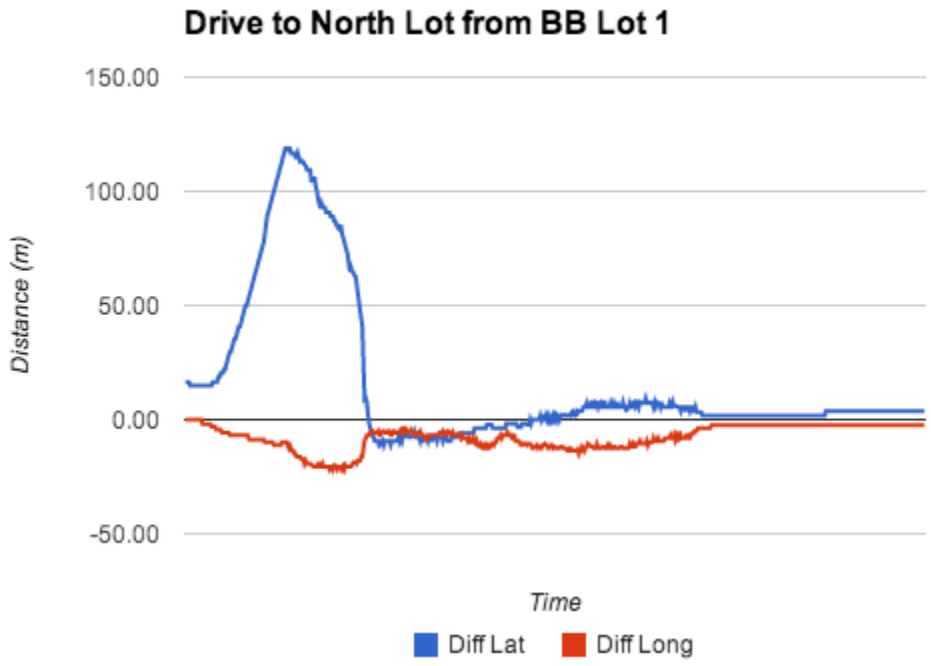
- Compared with the North Lot stationary, this test was a whole lot worse.
- This particular location seems to be very bad for GPS as other bad data was taken here on 4/2/14
- Only 26% of the data was within the 6m radius, which is really poor.

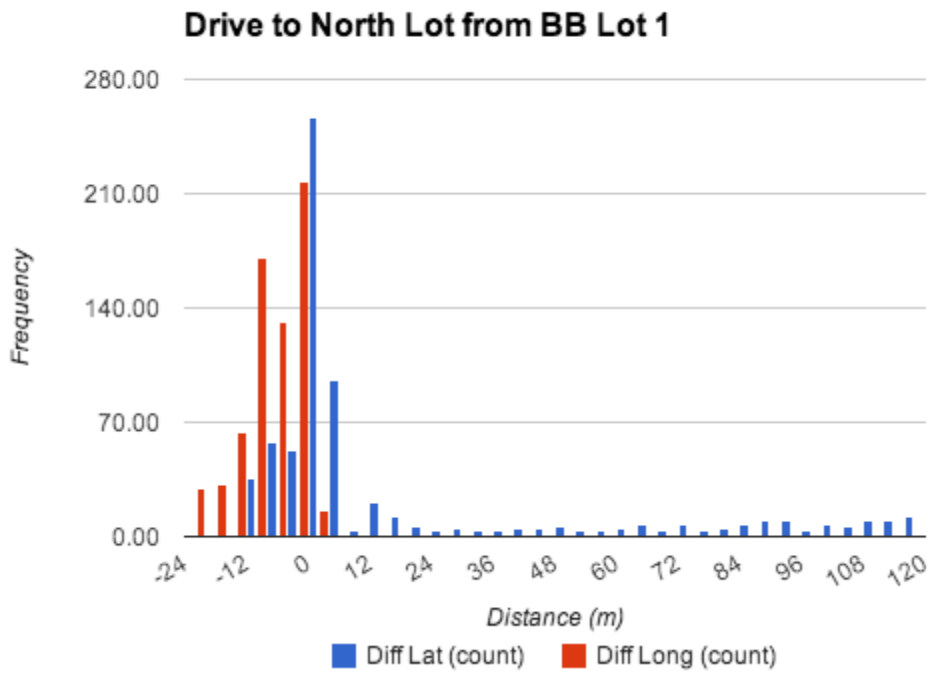
A.5.3.2.5 **Test Number** 4 - Drive to North Lot from BB Lot 1

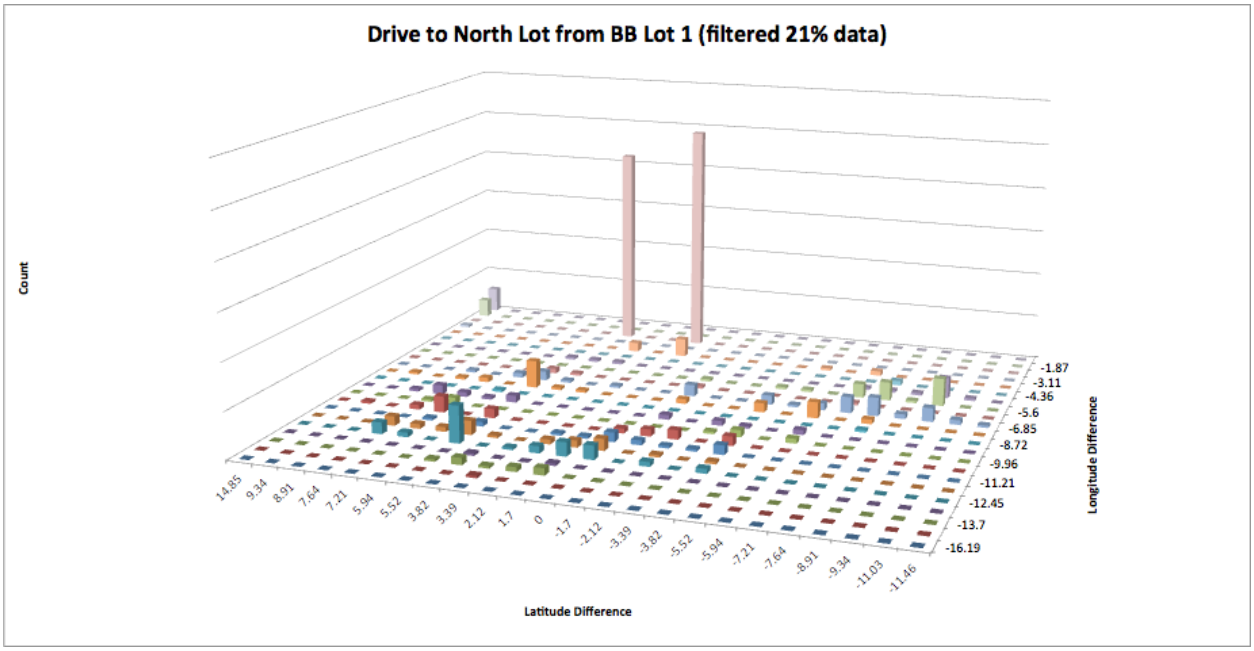
Notes:

- Drive to North Lot from BB Lot 1
- started at 10:51
- Speed around 20 mph
- stopped car at 10:52:45a

Results:







	Time	Latitude	Longitude
Average	0.2	19.33	7.68
Standard Deviation	0.05	32.17	5.30

Data within 7.2 meter radius (All): 36%

Data within 7.2 meter Radius (21% omitted): 45%

Summary:

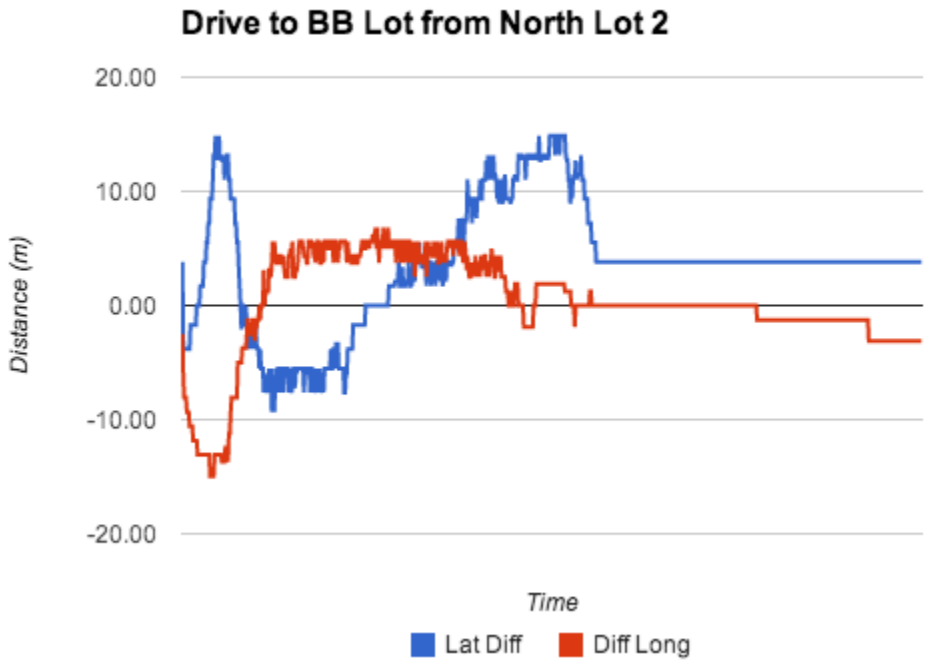
- For an unknown reason, Latitude data became exorbitantly terrible. We are not sure if there was a queuing issue within one of the GPS units that the comparison kept happening to an old position even as the car moved away. Over time, the latitude data did correct itself, but not after being over 100 meters off.
- When this bad latitude data was taken off there was still only 45% of the data within the expected range.

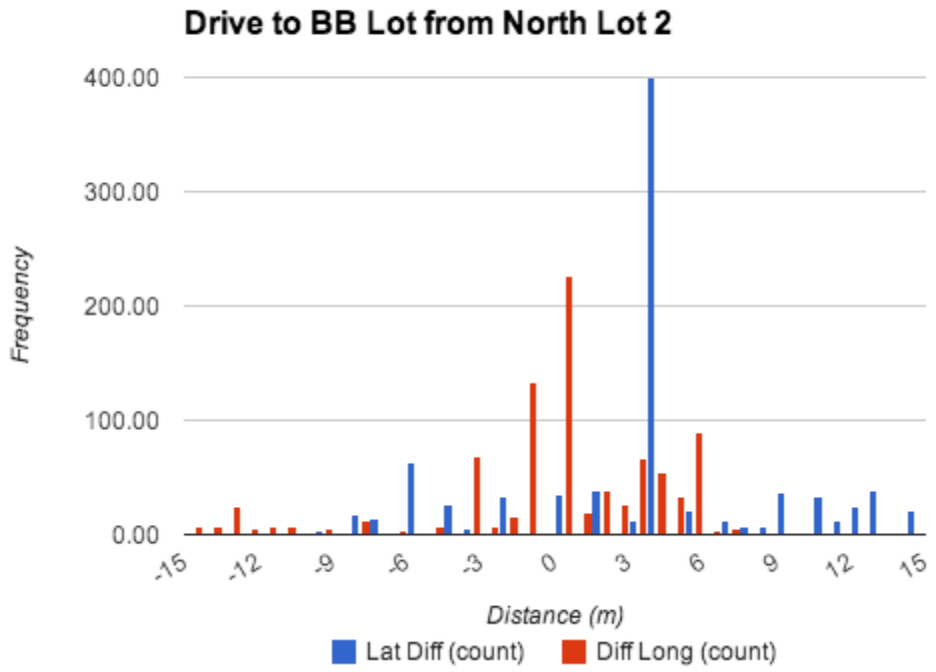
A.5.3.2.6 **Test Number** 5 - Drive to BB Lot from North Lot 2

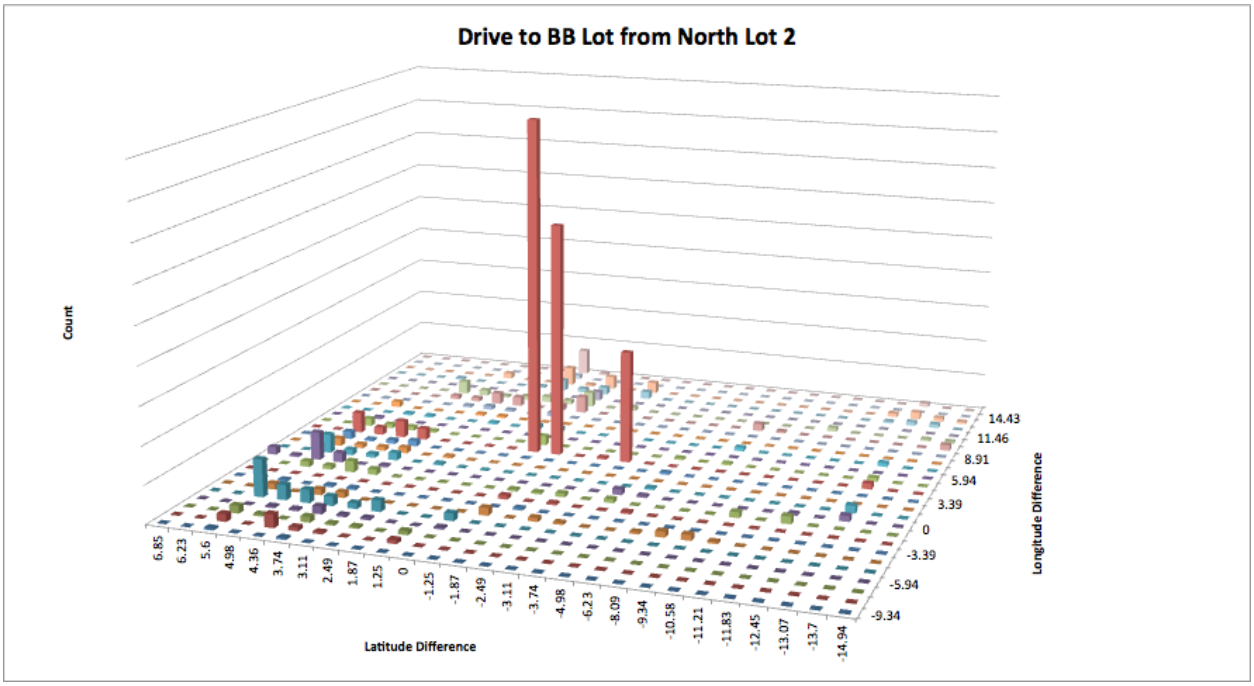
Notes:

- Drive to BB Lot from North Lot 2
- started at 10:55
- Speed around 20 mph
- car stopped around 10:56:47

Results:




Drive to BB Lot from North Lot 2
400.00
300.00
200.00
100.00
0.00
Frequency
-15
-12
-9
-6
-3
0
3
6
9
12
15
Distance (m)
Lat Diff (count)
Diff Long (count)


Drive to BB Lot from North Lot 2
Count
Latitude Difference
6.85
6.23
5.6
4.98
4.36
3.74
3.11
2.49
1.87
1.25
0
-1.25
-1.87
-2.49
-3.11
-3.74
-4.98
-6.23
-8.09
-9.34
-10.58
-11.21
-11.83
-12.45
-13.07
-13.7
-14.94
Longitude Difference
14.43
11.46
8.91
5.94
3.39
0
-3.39
-5.94
-9.34

	Time	Latitude	Longitude
Average	0.2	5.42	2.96
Standard Deviation	0.05	3.53	3.20

Data within 7.2 meter radius: 72%

Summary:

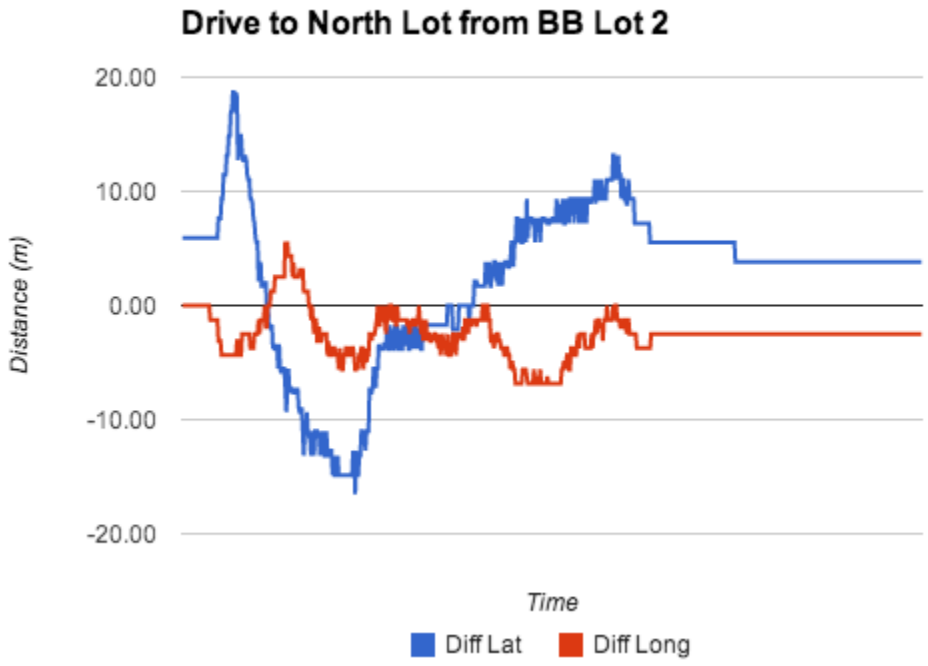
- The second test to BB Lot from North Lot proved a little bit better with 72% of the data being within range. However, this still isn't great data.

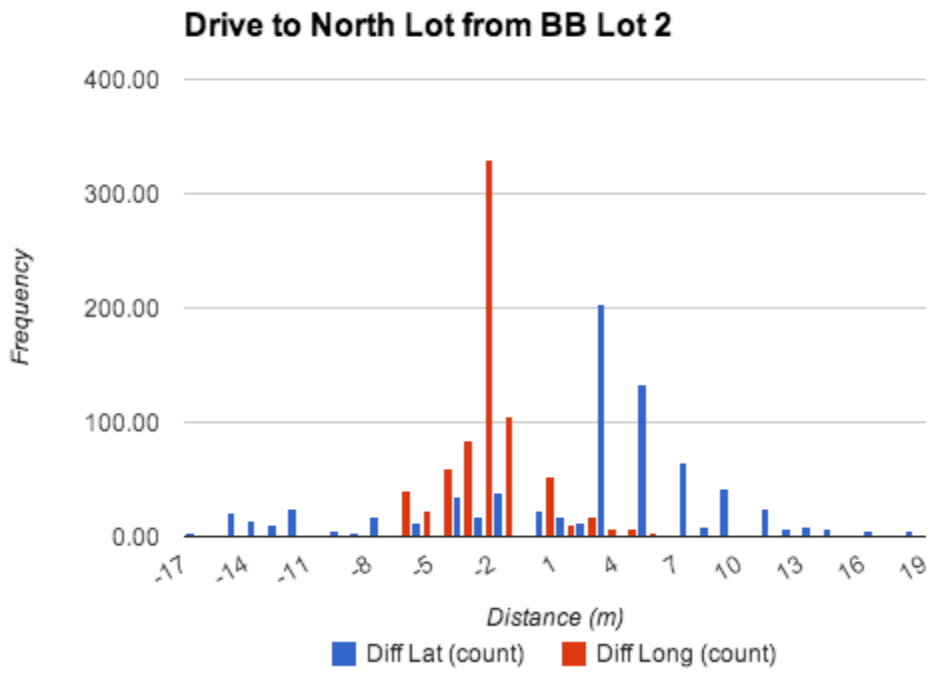
A.5.3.2.7 **Test Number** 6 - Drive to North Lot from BB Lot 2

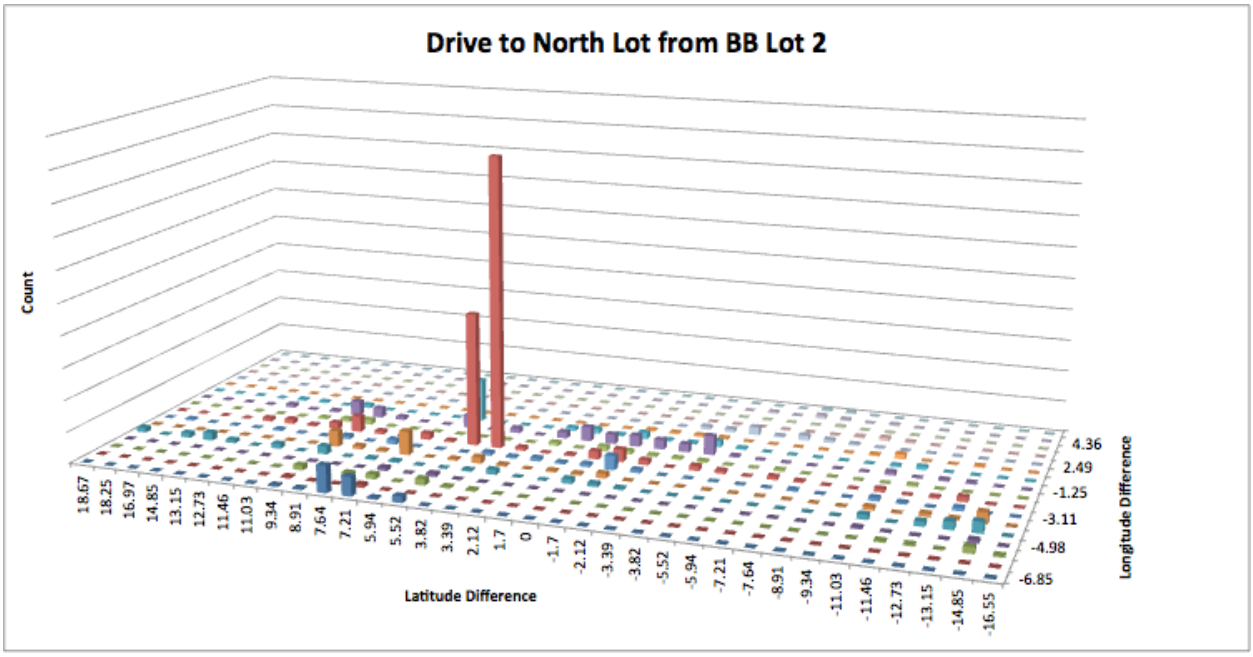
Notes:

- Drive to North Lot from BB Lot 2
- Speed around 20 mph
- started at 10:59
- stopped car at 11:00:40

Results:




Drive to North Lot from BB Lot 2
400.00
300.00
200.00
100.00
0.00
Frequency
-17
-14
-11
-8
-5
-2
1
4
7
10
13
16
19
Distance (m)
Diff Lat (count)
Diff Long (count)


Drive to North Lot from BB Lot 2
Count
18.67
18.25
16.97
14.85
13.15
12.73
11.46
11.03
9.34
8.91
7.64
7.21
5.94
5.52
3.82
3.39
2.12
1.7
0
-1.7
-2.12
-3.39
-3.82
-5.52
-5.94
-7.21
-7.64
-8.91
-9.34
-11.03
-11.46
-12.73
-13.15
-14.85
-16.55
Latitude Difference
4.36
2.49
-1.25
-3.11
-4.98
-6.85
Longitude Difference

	Time	Latitude	Longitude
Average	0.2	6.07	2.76
Standard Deviation	0.05	3.70	1.54

Data within 7.2 meter radius: 66%

Summary:

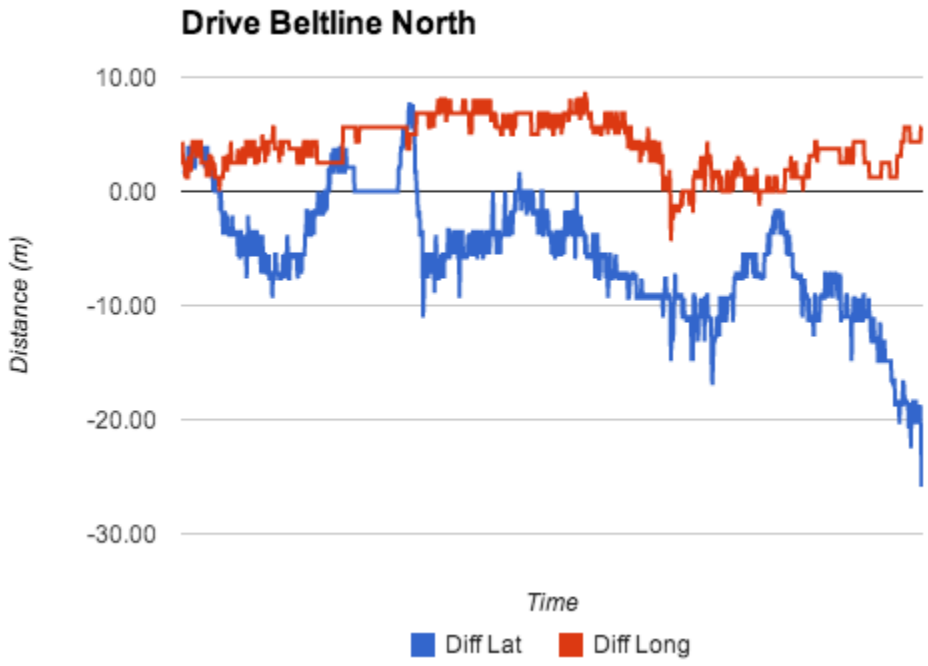
- Just like the previous test, the second round of BB Lot to North Lot proved more precise with 66% of the data being within the expected range.
- Nevertheless, this data is still isn't great.

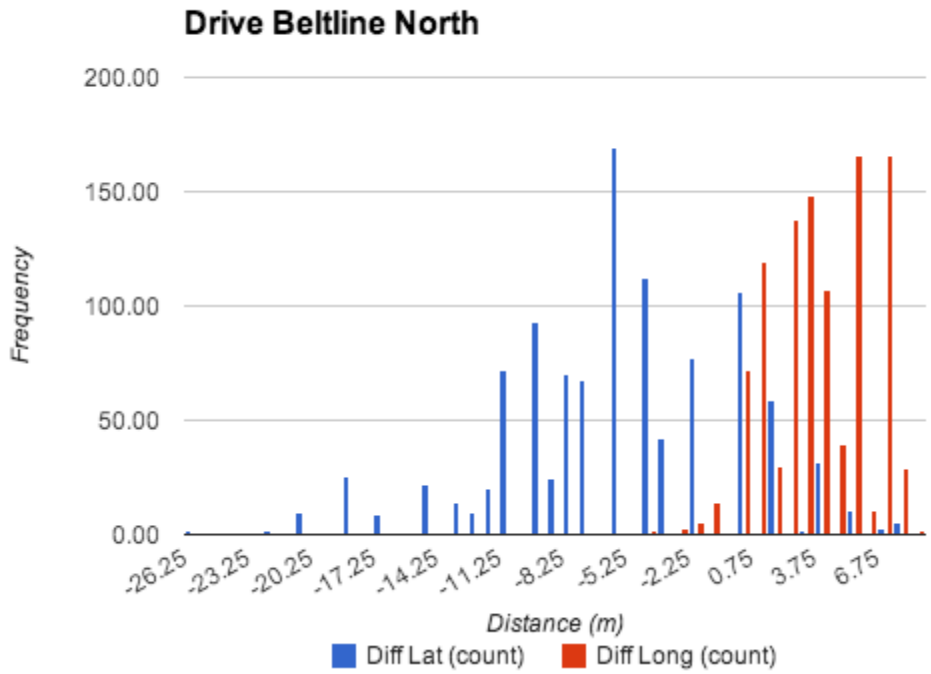
A.5.3.2.8 **Test Number** 7 - Drive Beltline North

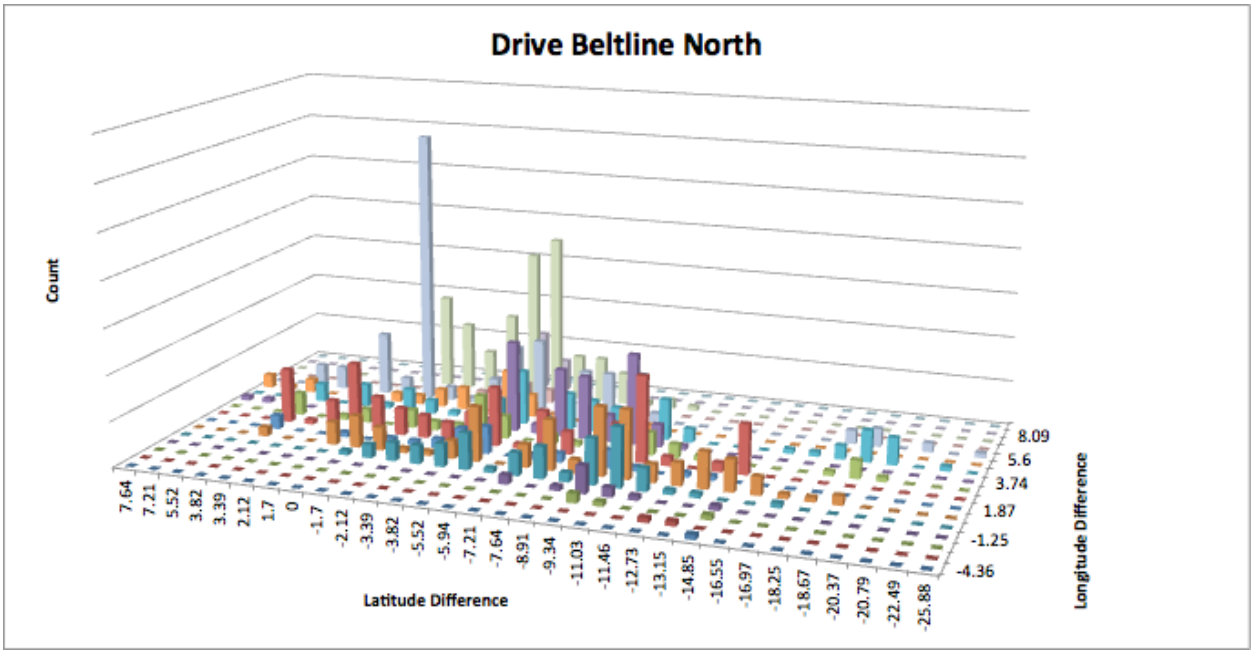
Notes:

- Drive on Beltline North
- Hard acceleration to 45 mph
- Average speed around 45 mph
- stoplight at: 11:05:53
- green 11:06:13
- stoplight at: 11:08:06a
- green 11:08:15a

Results:







	Time	Latitude	Longitude
Average	0.2	6.27	3.95
Standard Deviation	0.04	4.51	2.20

Data within 7.2 meter radius: 56%

Summary:

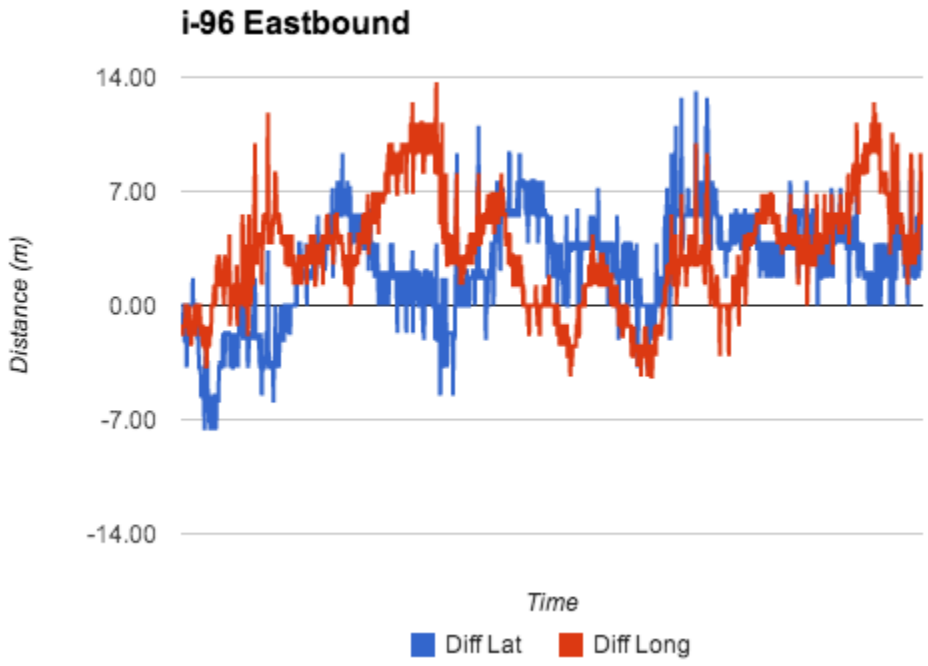
- The data gathered was actually quite good until the end when the Latitude data continued to plummet farther and farther away. Possibly another queuing issue of data?
- 56% of the data landed within the expected radius.

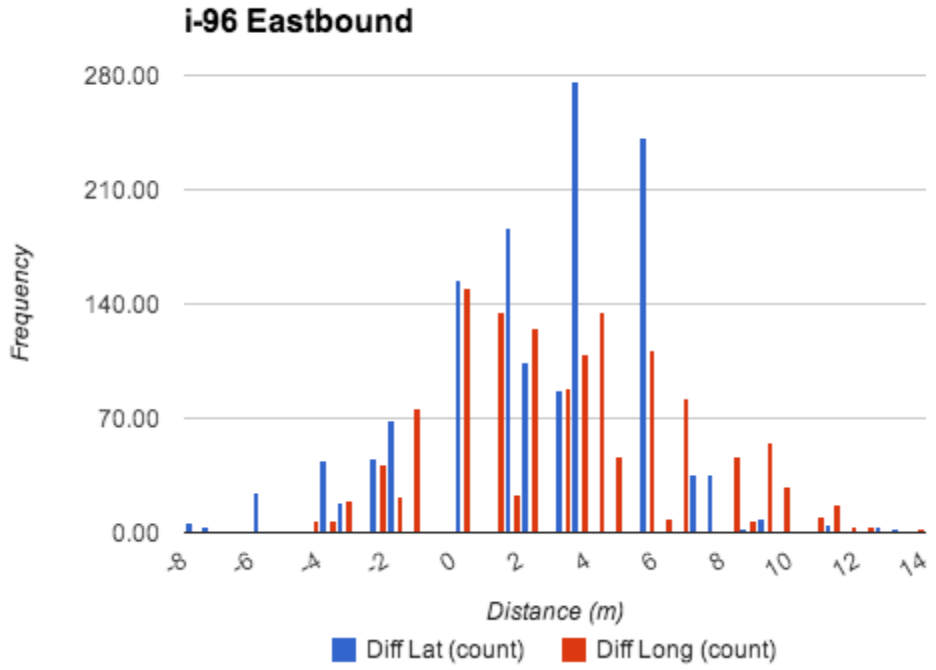
A.5.3.2.9 Test Number 8 - i96 Eastbound

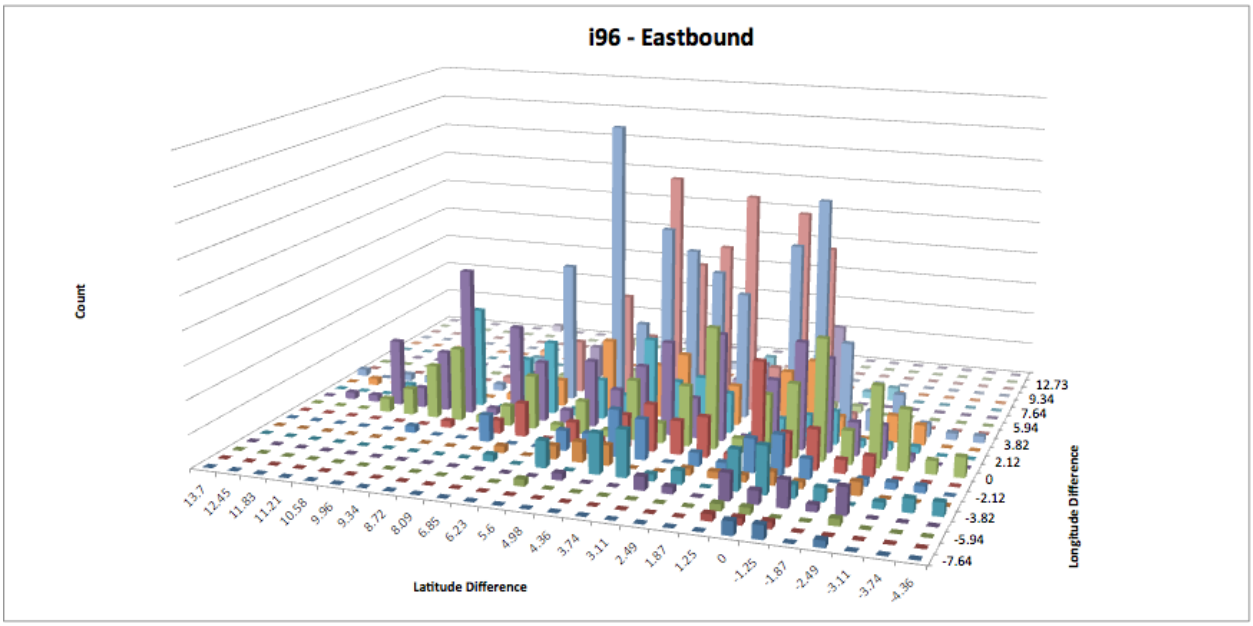
Notes:

- Drive on i-96 eastbound
- started at 11:11
- Up to speed at 11:12:30
- Average Speed around: 70mph

Results:







	Time	Latitude	Longitude
Average	0.2	3.38	3.83
Standard Deviation	0.05	2.14	2.83

Data within 7.2meter radius: 81%

Summary:

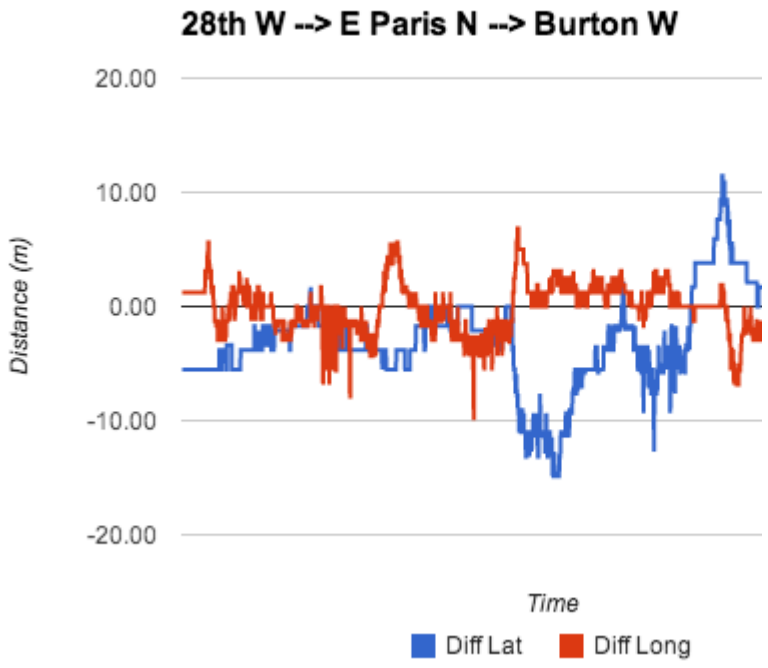
- This test gathered very good data with 81% of the data points landing within the expected range. It seems that with higher speed (20 mph -> 45 mph -> 70 mph) that the GPS precision also increased.

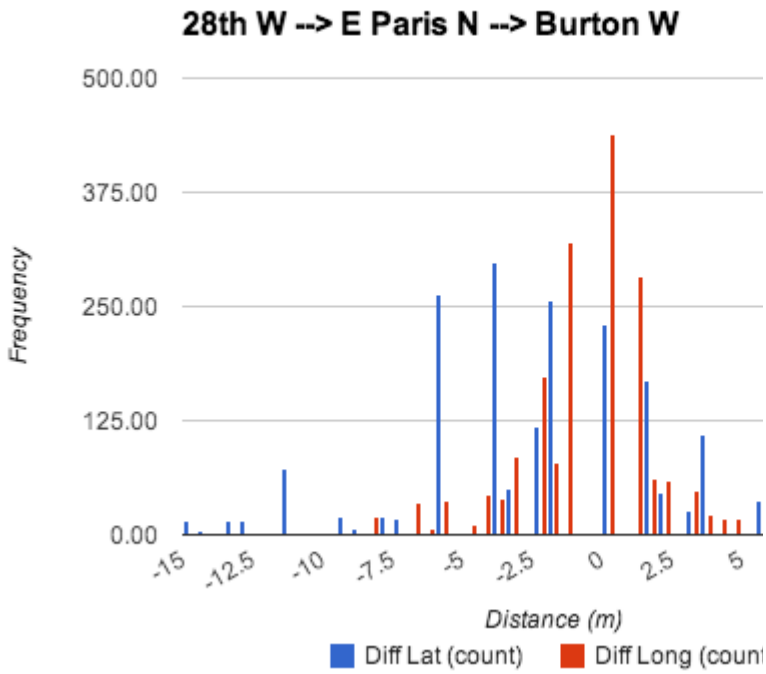
A.5.3.2.10 Test Number 9 - 28th W → E Paris N → Burton W

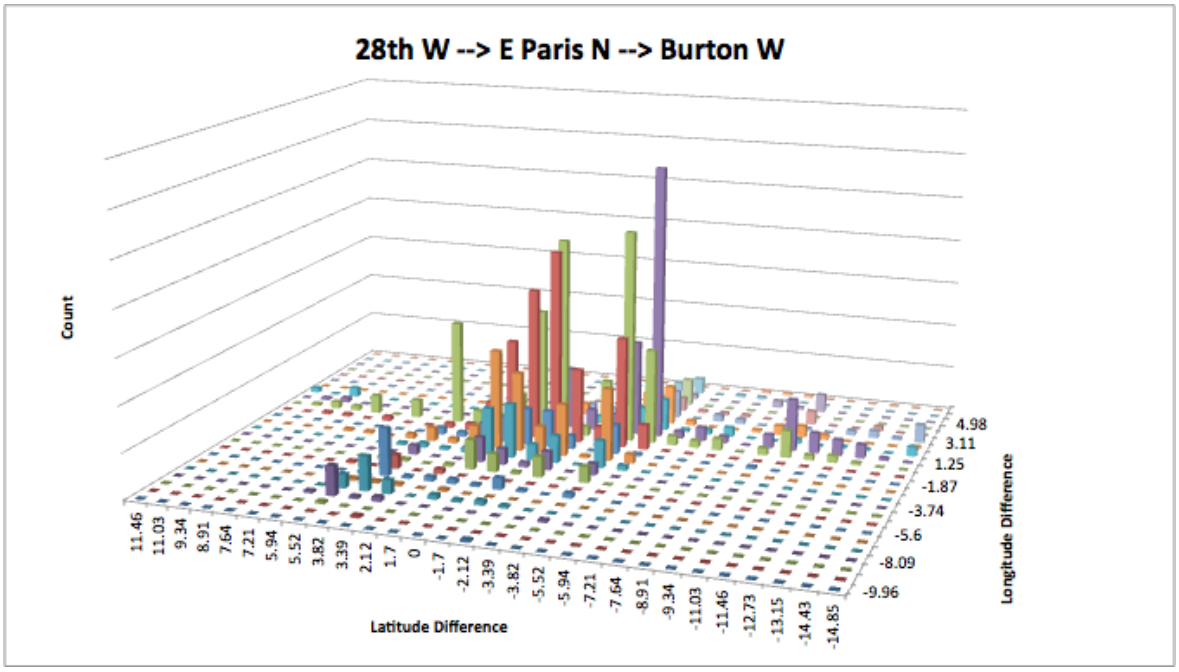
Notes:

- Drive 28th W, E Paris N, Burton W: many stoplights, real life
- started at 11:17
- Top speed around 45 mph
- normal speed around 38mph
- Turn on East Paris north 11:20:15
- full stop at light 11:21:54
- Turn on Burton west 11:22:05

Results:







	Time	Latitude	Longitude
Average	0.2	3.68	1.84
Standard Deviation	0.05	3.01	1.73

Data within 7.2 meter radius: 81%

Summary:

- It is interesting that with this most real-life situation, the data gathered was far better than that taken on Calvin's Campus. The car traveled varying speeds (from stops at lights to 45 mph) and also various directions.
- 81% of the data landed within the expected radius.

A.6 Video

A.6.1 RTSP Latency

A.6.1.1 RTSP Latency Test 1

Name: Spencer Olson, Jared Zoodsma
Date: 2/10/2014
Test Number: 1

Item under test: Video Latency
Testing Parameters:

- GUI Version: “TCP->GUI->Arduino->GUI_RevA”
- Gateway used to run GUI
- Video Parameters: 480 x 852, 24 frames per second
- RTSP used for video protocol

Notes:

Results: 1.8 second latency

A.6.1.2 RTSP Latency Test 2

Name: Spencer Olson, Jared Zoodsma
Date: 2/10/2014
Test Number: 2

Item under test: Video Latency
Testing Parameters:

- GUI Version: “TCP->GUI->Arduino->GUI_RevA”
- Gateway used to run GUI
- Video Parameters: 480 x 852, 10 frames per second
- RTSP used for video protocol

Notes: Video quality seemed very inconsistent and began to skip

---

Results: 2.3 second latency

A.6.1.3 RTSP Latency Test 3

Name: Spencer Olson, Jared Zoodsma
Date: 2/10/2014
Test Number: 3

Item under test: Video Latency
Testing Parameters:

- GUI Version: “TCP->GUI->Arduino->GUI_RevA”
- Gateway used to run GUI
- Video Parameters: 1080x1920, 24 frames per second
- RTSP used for video protocol

Notes: GUI struggled rendering full HD picture

Results: 2 second latency

A.6.1.4 RTSP Latency Test 4

Name: Spencer Olson, Jared Zoodsma
Date: 2/10/2014
Test Number: 4

Item under test: Video Latency
Testing Parameters:

- GUI Version: “TCP->GUI->Arduino->GUI_RevA”
- Gateway used to run GUI
- Video Parameters: 1080x1920, 10 frames per second
- RTSP used for video protocol

Notes: GUI struggled rendering full HD picture

Results: 2.4 second latency

A.6.2 Summary

Name: Drew Brandsen, Spencer Olson, Jared Zoodsma
Date: 4/18/2014

Item under test: Video Latency and Video Quality
Testing Parameters:

- Telemetry data was sent while video streaming
- Quad was flying (RC present) while video streaming.

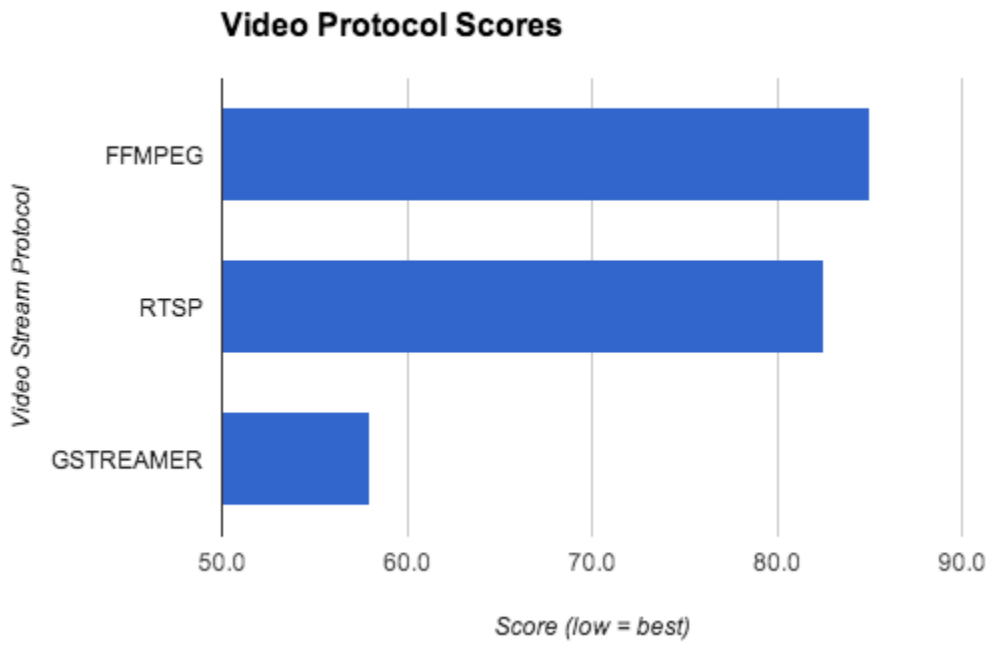
Notes:

- Weather was favorable with little wind.
- Flights lasted around 2 minutes
- Tests were done outside the Engineering Building garage door.

Conclusion:

- Although GStreamer produced more video jitter than the other protocols used, it proved to be the best protocol due to significantly lower video latency
- The GStreamer video jitter was reduced by using higher frame rates
- After preliminary tests, GStreamer with 480p resolution and 60 fps provided the best results.
    - Further tests can be performed to optimize video stream specifications.
    - Graphs of the results are seen on the next page.

Graphs:
<u>Results of Video Stream Protocol:</u>



Results of Video Stream Specifications