Project ID: FJL-BLMP
Project ID: GTH-AAA6

Towards the Realization of an Autonomous Blimp

Major Qualifying Project Report

Submitted to the Faculty
Of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree of Bachelor of Science
in Electrical and Computer Engineering
and Computer Science
by

Jeremy Colon (CS)
Melinda Race (ECE)
Darius Toussi (ECE)
Richard Walker (ECE)

Date Submitted:
April 25, 2013

Project Advisors:
George Heineman
Fred Looft

Abstract

This project continues an MQP from the 2011-12 academic year: Development of an Autonomous Blimp. The objective of our project was to modify the existing design by implementing autonomous flight capability using an Android phone, a power system capable of tracking energy usage, and a camera mission module for aerial photography. Through testing and flight tests, we demonstrated that we successfully implemented non-optimized autonomous flight, portions of the power system design, and a camera mission module.

Acknowledgements

We would like to thank our advisors Professor George Heineman and Professor Fred Looft for their continued support throughout this project. We would also like to thank the following individuals for their help with this project:

- Thomas Angelotti
- Danielle Beaulieu
- Torbjorn Bergstrom
- Professor Stephen Bitar
- Adam Blumenau
- Nicholas DeMarinis
- Cathy Emmerton
- Richard Gammon
- Mitchell Monserrate
- Patrick Morrison
- Earl Ziegler

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Introduction

The Autonomous Blimp project was started by a Major Qualifying Project (MQP) team during the 2011-2012 academic year. The team intended for the blimp to be capable of autonomous flight and perform multiple missions through the use of interchangeable mission modules. The blimp was also intended to sustain flight for an endurance flight time of 90 minutes while using only the on-board power system.

This project had an ambitious set of goals. The accomplishments of the team included the design and construction of the gondola, selection of the blimp's shell, creation of the drive system, development of the control system hardware and software, and the completion of two mission modules. The drive system was made up of the left, right, and tail motors and the servo. The control system consisted of the components required for manual and autonomous flight. The team did not achieve fully-functional autonomous flight capability or complete a video mission module, nor did they achieve the 90 minute endurance flight.

The team recognized the potential for future work on their project, and identified various areas in which the blimp could be improved.

1.2 Project Statement

Our project builds upon the blimp system designed and developed by the previous MQP team. The goals of our project included the implementation of autonomous flight using an Android phone as the flight processor, improvements to the existing power system to increase the flight time and monitor gondola power usage, and the development of a video mission module mounted in the gondola. Each of these goals was given equal priority, and all were seen to completion.

1.3 Summary

In this chapter we briefly introduced our project. In the remainder of this report, we will detail the development of our project. More details about the goals, accomplishments, and the problems encountered by the 2011-2012 project team are provided in Chapter 2. We present background research relevant to our project developments and a more detailed list of our project goals and objectives in Chapter 3. We discuss our methods and results for each goal in Chapters 5-8, and explain our overall results in Chapter 9. We provide recommendations for future work

in Chapter 10. For the remainder of this report, the 2011-2012 MQP team will be referred to as the “2011 AY team”.

2 Background

2.1 Autonomous Blimp Project Goals

The 2011 AY team intended to design an autonomous dirigible-based platform that had the capability to perform multiple missions through the use of interchangeable mission modules. To accomplish this goal, the team planned to design a blimp that would fulfill the following requirements:

A. Outdoor Operation
B. Autonomous Navigation
C. Performing Multiple Roles
D. Endurance Flights
E. Two-Way Communication with a Base Station

We describe each of these requirements in further detail below.

2.1.1 Outdoor Operation

Mission tasks and objectives were based on outdoor operation, thus the blimp needed to be maneuverable outdoors. To constrain the design goals, Standard Operating Conditions (SOC) were created and are shown in Table 1. These SOC were chosen because aircraft of similar size and purpose were designed to operate under similar conditions.

Conditions:	Limits:
Wind (m/s)	≤ 2.57
Temperature (°C)	-9.44 – 43.33
Precipitation Levels (mm/hour)	0.00 – 1.00
Altitude (meters)	3.05 – 30.48

Table 1: Standard Operation Conditions for outdoor operation of the blimp

2.1.2 Autonomous Operation

The 2011 AY team defined autonomous operation as the ability to navigate between given GPS (global positioning system) waypoints with minimal deviations from the intended path. Deviations can be defined as any time the blimp goes off of its course which includes altitude, heading, and speed corrections. The intended path is the shortest path to the next waypoint. Within the context of autonomous navigation, the blimp was considered to have

successfully reached a waypoint if it was within 4.6m (15ft.) horizontally and 1.8m (6ft.) vertically of its intended GPS coordinate.

2.1.3 Performing Multiple Roles

The original intent was to create a platform capable of fulfilling multiple roles through the use of interchangeable mission modules. The hardware for each module would vary depending on the tasks that needed to be accomplished in each mission. Examples of possible missions included, but were not limited to: search and rescue, range extension, delivery, fire spotting, advertising, communications relay, and swarm control.

2.1.4 Endurance Flights

One goal of the 2011 AY team was for the blimp to achieve a 90-minute endurance flight time using only the onboard batteries. The team defined an endurance flight as the ability for the blimp to maintain its position in a 2.6 m/s (5 knot) headwind. For comparison, similarly powered remote control (RC) fixed-wing aircrafts have endurance flight times of 20-30 minutes. The 2011 AY team planned for the blimp to be capable of maintaining flight times up to 3 times longer than fixed-wing counterparts [1]. It was planned that the blimp's flight time could be increased by adding more batteries as the scale and weight of the blimp increased.

2.1.5 Two-Way Communication with a Base Station

Lastly, the 2011 AY team required the blimp to be capable of two-way communication with the base station, which is located on the ground. This requirement was derived for two reasons.

The first reason was a mandate by the Federal Aviation Administration (FAA) that all autonomous aircraft have a two-way link to their ground control that overrides automatic controls for safety reasons. Having ground control would allow for increased safety during landing procedures, and the option to abort missions.

Secondly, the ground station would allow the user to send mission updates to the blimp based on changing conditions, the return of the blimp's telemetry information such as speed, position, and state, and the communication of mission specific data to the user.

2.1.6 System Requirements

To fulfill the goals listed in the previous sections, the 2011 AY team designed the blimp system to meet specific requirements. Each requirement was derived from the team's goals and

the goals related to requirement [A-E] are indicated in square brackets below. Distances were defined from a point within the center of the gondola. Please note that these requirements are stated here verbatim to represent the intentions of the 2011 AY team [1].

1. Can navigate to designated GPS coordinate to within 15 ft. horizontal, 6 ft. vertical displacement, outside the resolution of the onboard GPS, from the destination coordinate within Standard Operating Conditions (S.O.C.) using only the onboard GPS system, an inertial measurement unit, and a barometric altimeter [A, B]
2. Can maintain position at designated GPS coordinate to within 15 ft. horizontal, 6 ft. vertical displacement, using only the onboard Control system, within S.O.C. for a period of no less than 1 hour. It is allowable for the vehicle to drift from the designated area by no more than 30 ft. laterally and 15 ft. vertically, for no more than 5 minutes of the above hour, consecutively or non-consecutively. [A, B]
3. Interchangeable mission specific hardware up to 1.36 kg (3 pounds) which may be capable of two-way communication with the base station using serial communications. [C]
4. Can stay aloft with 3.09 m/s (6 knots) airspeed for a minimum of 90 minutes using only onboard battery and/or solar power under S.O.C. [D]
5. Can utilize two-way communication with the base station to sufficiently to communicate telemetry and instructions at a range of at least 200 m. [E]

2.2 Project Accomplishments

The accomplishments of the 2011 AY team included the creation of a hardware platform, an electronics control system, software for system control and navigation, and mission modules.

2.2.1 Gondola

The gondola was needed to hold the system hardware and electronics. It needed to be sturdy and lightweight, and able to attach to the shell for flight tests. The completed gondola, shown in Figure 1, was designed and constructed to contain the on-board electronics and the hardware for the control system and navigation. While Figure 1is provided to show what the actual gondola looks like, readers may be interested in Figure 2 which contains a Computer Aided Design (CAD) drawing of the gondola and provides a detailed explanation of all parts.

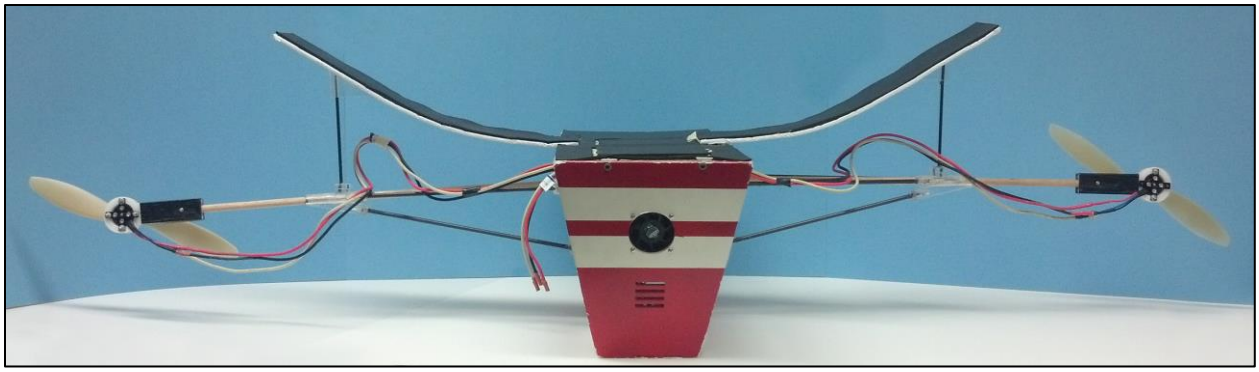


Figure 1: Gondola created by 2011 AY team

The 2011 AY team selected appropriate materials for the gondola body after conducting three-point bending and tensile strength tests on each of the possible materials. The team used foam core to construct the gondola body. As shown in Figure 2, the team used a CAD program to design the gondola and used this design to laser-cut the pieces of foam core. The final gondola body was made of laser-cut pieces of foam core attached with epoxy. The result was a sturdy lightweight gondola capable of holding all of the blimp materials. Separate compartments exist to house each major subsystem or component. The compartment holding the RC components and the processor is above the compartment for the power distribution boards. The battery compartment is directly below the aforementioned compartments. The mission module compartment is on the opposite end of the gondola and has two main sections, upper and lower.

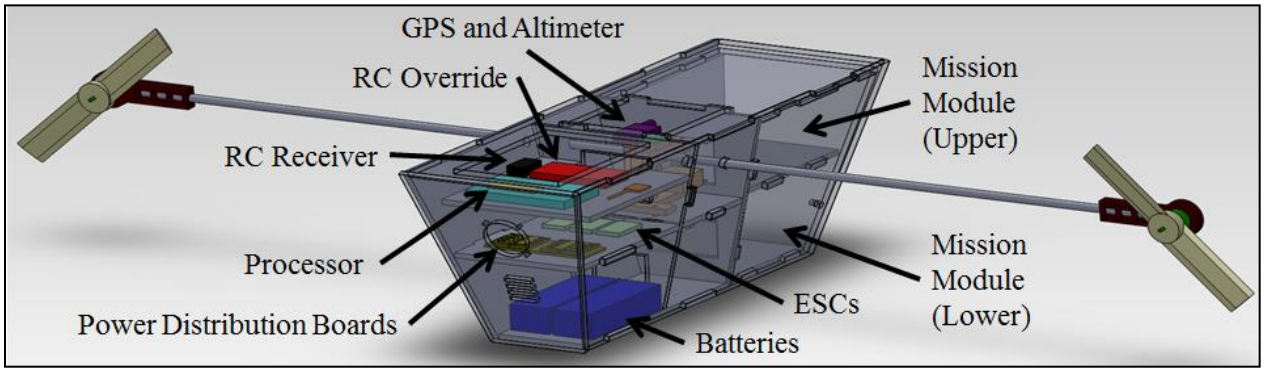


Figure 2: CAD Drawing of Gondola [1]

The hardware and drive system was designed to fit inside of the gondola, with the exception of the main axle holding two of the drive motors. Axle materials underwent two-point bending tests, and different materials were tested on the gondola. The final axle was a wooden dowel.

The gondola, hardware, and drive system proved effective under the SOC during ground and flight tests. During flight tests the system was capable of making turns, holding position, and maneuvering in 2.6 m/s (5 knot) winds.

2.2.2 Electronics

The 2011 AY team successfully designed and constructed an electronics control system for blimp operation, shown in Figure 3. This electronics system successfully integrated a Gumstix Overo Water processor and a Gumstix RoboVero expansion board [1]. The power distribution system to the altimeter and the GPS was fully functional, and experienced no failures during the course of the project. The separate power distribution boards for the 7.4 V and 11.1 V systems were also fully functional without failures. The electronics allowed the drive system to be overridden with an RC controller. This was accomplished through the use of a multiplexer, which selected signals from either the onboard processor or the RC controller.

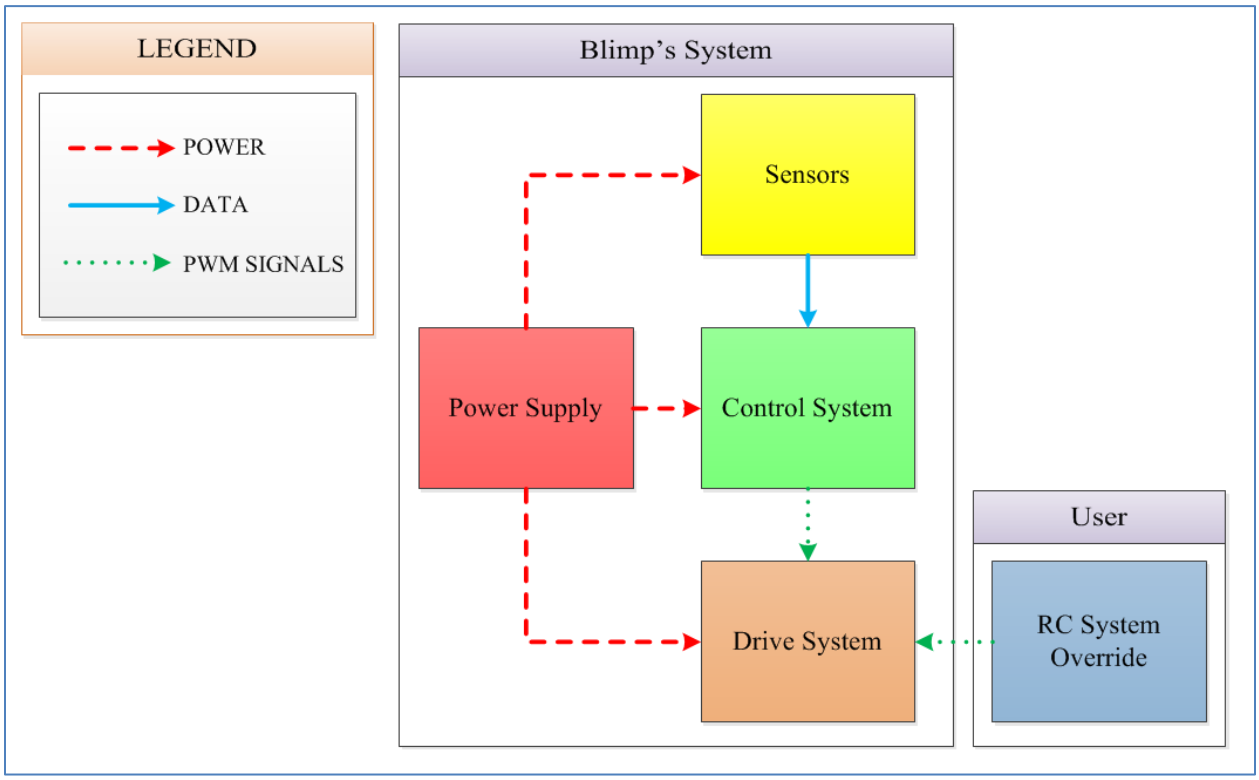


Figure 3: Top-Level System Diagram of 2011 AY team Electronics

2.2.3 Software

The 2011 AY team wrote Python code for the Gumstix Robovero to control the electronics and to implement autonomous flight. The high-level software was designed to call

upon the low-level software and was written to implement autonomous flight, but was not completed or fully tested. Their low-level software to control the drive system was fully tested and was successful in the following functions [1]:

1. Their control system was capable of driving each motor at the desired speed and reliably setting the angle of the servo.
2. Their software successfully communicated between the GPS module and the processor, so they could access the readings output by the GPS module.
3. The team was able to use the magnetometer to determine a compass heading accurate within 5 degrees.
4. The barometric altimeter was able to determine an altitude relative to the altimeter's starting altitude that was accurate within 2 feet.
5. The navigation system was able to successfully determine the range to a given GPS waypoint and determine the bearing that the gondola would need to reach that point.

The completed blimp is shown in Figure 4.

Figure 4: WPI Blimp (Shell and Gondola)

2.2.4 Mission Modules

One of the 2011 AY team's goals was to have interchangeable mission modules that would allow the blimp to perform multiple roles. The team successfully created two mission modules to perform specific tasks, and began work on other mission modules that were not completed or fully tested. The completed mission modules were the drop module and the range extension module. The drop module was capable of holding a small payload and releasing it when a switch on the RC transmitter was flipped. The range extension module contained two

extra batteries for the main system, and served to extend the possible flight time of the blimp. The 2011 AY team began work on a video module, but the module was not completed. The team also documented in their final report a list of possible tasks that future mission modules could fulfill, including advertising, disaster relief surveying, reconnaissance, and event photography or recording [1].

2.3 Problems

In this section we discuss the recurring problems encountered by the 2011 AY team. These problems included mechanical hardware failures, drive and power system efficiency problems, maintaining organized documentation, and the limited time available to complete the project.

2.3.1 Hardware Failures

The first recurring problem was a mechanical hardware issue with the axles of the motors. During motor and propeller testing, the shafts of the motors failed several times. As seen in Figure 5, the shaft or "axle" of the motor broke due to gyroscopic forces that exceeded the specification of the soft iron shafts. The 2011 AY team explained this failure as a poor choice of motors on their part. As a result, they purchased new motors with axles that could withstand the gyroscopic forces, but they were unable to verify that the new axles did not fail because of their limited time frame.

The second recurring hardware issue was an electrical hardware problem. The problem was that the electrical hardware had difficulty communicating with the ground station. This was caused by a faulty USB-to-serial adapter that was purchased and required replacement. This communication system was not critical to flight, so the team focused on other aspects of the project. As a result, the communication system was not fully implemented.

Figure 5: Axle Failure [1]

2.3.2 Drive and Power System Efficiency

Another recurring problem was meeting their goal for the efficiency of the drive system and power system. When constructing the gondola and mounting it onto the shell, they did not consider the benefits of utilizing the lifting body[1] of the blimp shell. As a result, they intended to use the thrust generated by the propellers to increase or decrease altitude as well as change the yaw and generate forward and rearward thrust. If the blimp's lifting body were used, turning and altitude adjustments would not require as much power from the propellers, and the system would have been more efficient. The second issue with the efficiency of the drive system and power system was that the power system of the blimp was not able to achieve the goal of a 90-minute endurance flight due to the fact that the batteries did not have the necessary capacity.

2.3.3 Documentation and Organization

One of the problems with the blimp's gondola was the organization of the wiring for the circuits and components. Figure 6 and Figure 7 below show the wiring inside the gondola. While both pictures show the significant amount of work that was done, it is clear that the wiring was cluttered and required organization. Unfortunately, the 2011 AY team did not fully document the entire inside of the gondola, specifically the Input-Output (I/O) ports of the circuit boards, the parts used and inventory of parts, circuit schematics, and datasheets for components purchased. As a result, many of the wires, I/O connections, and some of the other components could not be easily identified without stepping through and testing each part.

[1] Lifting body: the creation of aerodynamic lift "from the shape of the vehicles rather than from wings as on a normal aircraft" [2].

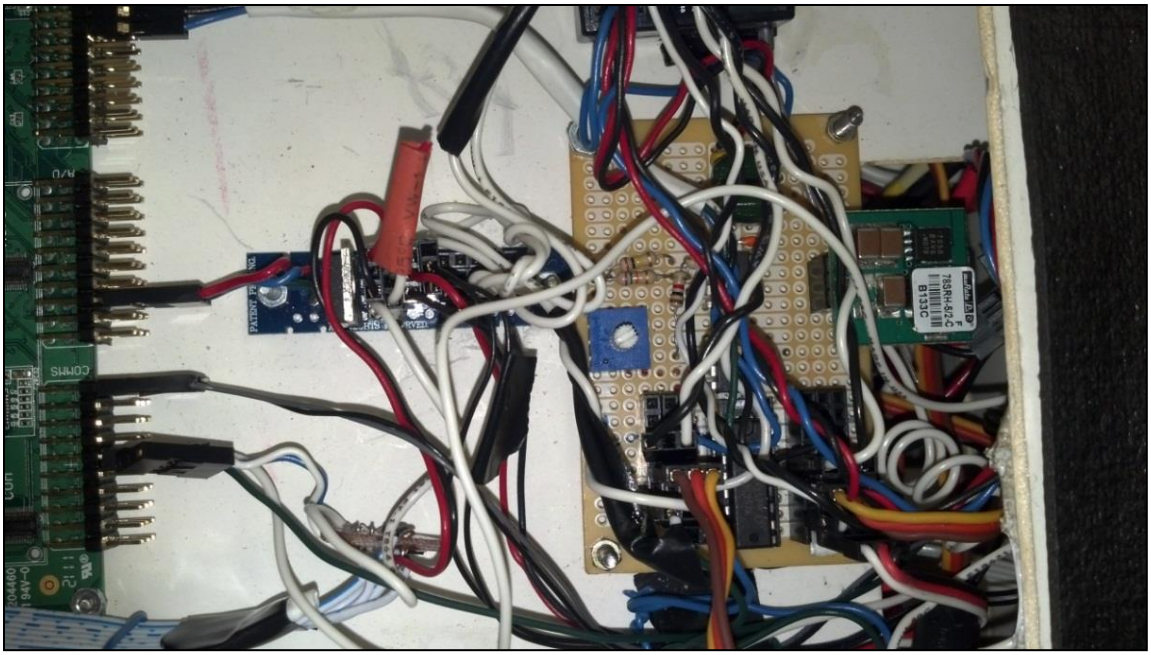


Figure 6: Gondola Circuitry

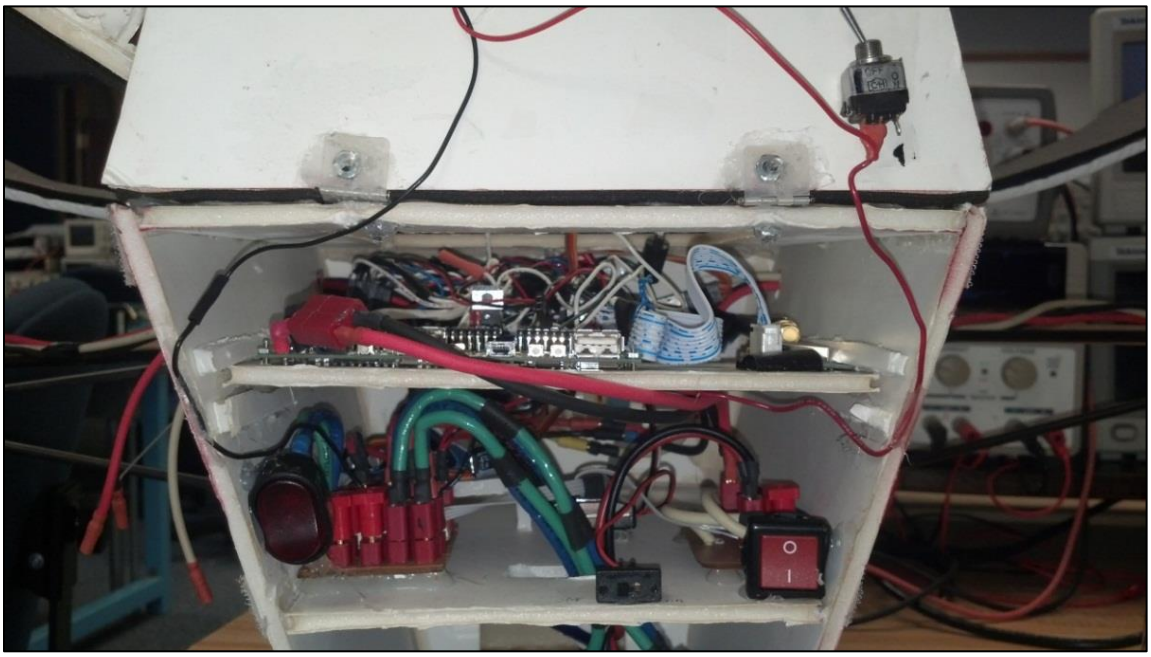


Figure 7: Wiring in the top and bottom layers of the gondola

2.3.4 Time Constraints

The final and most important problem that impacted the 2011 AY team was the limited time available to complete the project. Due to many issues encountered during troubleshooting, debugging, and waiting for ordered parts to arrive, some of their goals and milestones were not met [1]. These goals and milestones included the completion of a two-way communication system between the ground station and the blimp, software testing and implementation of the

navigation system for autonomous flight, thorough GPS testing and fine tuning, and the implementation of a working video module. While the team was unable to meet all of their desired goals, they laid a solid groundwork for our team to continue the project with a new set of goals.

2.4 Summary

In this chapter we discussed the goals of the 2011 AY team and the requirements that related to each goal. We discussed the 2011 AY team's accomplishments and detailed the challenges that the 2011 AY team faced.

3 Detailed Project Proposal

Topics and areas covered in this chapter include autonomous flight, the power system, and the camera mission module. After each aspect of our proposed blimp design is briefly discussed, we will follow with more details of the system design requirements in Section 3.4.

3.1 Hardware for Autonomous Flight

One of the problems identified early on was the limited processing capacity of the original blimp communications and control system. As a result of this and the inability of the 2011 AY team to implement autonomous flight, we decided to search for a new type of embedded computer system to meet our in-flight processing needs. Although details will be described in the Section 5, it is important to note that we strongly considered using an Android device because of the advanced technology available at the time.

To replace the hardware components with an Android device we first researched what was currently available on the market for Android devices so that we could determine what qualities our device should have. We wanted the Android device to contain all of the sensors that the 2011 AY team used in the gondola, which included the GPS, altimeter, accelerometer/gyroscope, and magnetometer. After researching numerous Android devices released between 2010 and 2012, we created a list of specifications that described the standard for Android devices [3]. The standard specifications for Android devices can be seen in Table 2.

Categories	Standard Android Device
Processor	1 GHz Dual-Core
Memory (RAM)	1 GB
Operating System	4.0 (Ice Cream Sandwich)
Storage	MicroSD
Wireless	Wi-Fi, Bluetooth
Connection	Micro-B USB, 3.5mm Audio Jack
Accelerometer/Gyroscope	+/- 250-2000 deg/sec
Magnetometer	Yes
Barometer	No
GPS	Yes
Battery	1780-1850 mAh
Weight	141.748 g

Table 2: 2012 Standards for Android Devices

3.2 Power System

Large commercial blimps are propelled by gas combustion engines, and smaller RC blimps are propelled with electric motors. The different mission profiles for the different types of blimps results in different flight times. While the average flight time for smaller blimps is about one hour, they can range from 40 minutes to 80 minutes, as shown in Table 3.

	Blimp Features			Flight Times		
Operation	**Length (m)**	**Number of Motors/Servos**	**Battery Capacity (mAHr)**	**Minimum (mins)**	**Maximum (mins)**	**Average (mins)**
Indoor	2.8	3	2500	40	60	50
	3.5	3	3200	50	80	65
	4.5	3	3200	60	80	70
Outdoor	5.0	3	9600	40	60	50
	7.0	7	17900	40	60	50
	10.0	7	17900	40	60	50

Table 3: Average flight times of indoor blimps versus outdoor blimps [4]

Electric motors are preferred for RC blimps because their speed can be controlled via electronic speed controllers [5]. The electric motors within the drive/propulsion system of a blimp require an electric power source.

The preferred electric power sources for RC blimps are rechargeable batteries. The most popular chemistries used in rechargeable batteries are lithium-ion (Li-ion) or lithium-polymer (LiPo) batteries. As shown in Table 4, lithium-based batteries are favorable for use in RC blimps because of their electrochemical potential[2] and high energy densities [6].

Characteristics	Battery Type					
	NiCd	NiMH	Lead Acid	Li-ion	Li-ion polymer	Reusable Alkaline
Gravimetric Energy Density(Wh/kg)	45-80	60-120	30-50	110-160	100-130	80 (initial)
Cycle Life (to 80% of initial capacity)	1500	300 to 500	200 to 300	500 to 1000	300 to 500	50
Fast Charge Time	1h typical	2-4h	8-16h	2-4h	2-4h	2-3h
Overcharge Tolerance	moderate	low	high	very low	low	moderate
Cell Voltage(nominal)	1.25V	1.25V	2V	3.6V	3.6V	1.5V

Table 4: Comparison of battery characteristics [7]

Power systems supplement and increase the efficiency of these drive systems by monitoring the status of the power source and regulating the current draw of other sub-systems. Power systems also reduce unnecessary power dissipations in some cases by replacing passive components with active components. Active components conduct electricity when needed, and are turned off when not in use, which results in lower power consumption. In contrast, passive components draw current continuously.

3.3 Mission Modules

Camera modules in blimps have been utilized for a wide variety of uses in academic projects, commercial purposes, and for use by non-profit organizations [8]. Companies produce blimps with commercial camera modules to be used for aerial photography and video recording, as well as event photography and security surveillance. Academic projects at universities in the United States, Germany, Japan, and Canada have designed camera modules for blimps for uses that include urban search and rescue, blimp navigation, monitoring gully erosion, coordinating ground robots, and surveillance systems. A non-profit organization in Spain uses a blimp with a camera module for aerial photography after disasters to assess the structural stability of buildings [8].

[2] Electrochemical potential: the nominal voltage across the battery.

3.4 Requirements

The requirements are tailored to our project goal for autonomous flight, the power system, and the mission modules. A detailed table of our proposed requirements can be found in Section 3.4.4.

3.4.1 Flight Requirements

Many of the components of the blimp system are classified as needed either for autonomous flight or controlled flight, or both. For example, the flight motors and control system are needed regardless of the type of operation the blimp is under, autonomous or controlled, whereas certain processing capability may only be needed for autonomous flight.

Due to the time limitations of our project, we intended to use the drive system components purchased by the 2011 AY team which include the servo, left motor, right motor, and tail motor. Since we planned on using these components, one of the requirements for the Android device that would control the drive system was that it must be able to control the servo and motor speeds through the use of pulse-width modulation (PWM) signals.

When researching and choosing components, we carefully considered the weight of each component because the blimp can only lift a maximum of 3629 grams (8 lbs). Our plan was to remove most of the previous hardware used in the control system, so we weighed the previous components, specifically the Gumstix Overo Water, Gumstix RoboVero, GPS, and altimeter, and found that they weighed a total of 162 grams (0.36 lbs). With this benchmark, we planned on selecting an Android device that weighed no more than 162 grams so as to not go over the previous total weight of the components used by the 2011 AY team.

We planned to write software that would allow the blimp to follow a non-optimized autonomous flight path. To accomplish this goal, one major requirement that we had for the software was that the code would log and update every reading of the blimp's current heading, GPS location, and altitude. We planned to implement code to control the motors and servo based on the data. We decided on a non-optimized autonomous flight path due to the time constraints of our project. From this we determined that a requirement for our project was that the blimp should be able to fly between a set of given GPS coordinates along a non-optimized path.

3.4.2 Power System Requirements

We planned on improving the efficiency of the power system. The previous power system used diodes to prevent unwanted discharge of the parallel battery packs. However, the

diodes had an associated voltage drop, which resulted in lower voltage across the motors than the batteries' voltage. To accomplish the goal of improving the efficiency of the power system, we decided that the new power system must ensure that the voltage across the motors should be within 0.2V of the voltage across the batteries.

The previous power system powered the motors without relaying energy usage information to the user of the blimp. To improve the functionality of the power system, it was required that the new power system monitors and logs the energy usage data of the blimp throughout its flight.

3.4.3 Mission Modules

Our functional goals for the camera module were to allow the remote viewing of the camera's video feed while the blimp is in flight and to capture and save images and video files to external memory in the camera. We decided that the maximum allowed flight altitude for the blimp was to be 30.5 meters (100 feet), and all flight tests were to be performed on the WPI football field. We wanted the camera module to have a wireless range long enough to allow the user to control the camera if the blimp was on one corner of the football field at its maximum altitude and the user was on the opposing corner. The diagonal length of the football field is 120 meters, and the maximum blimp altitude was 30.5 meters, so we wanted the wireless range of the module to be at least 125 meters. The entire mission module, excluding the foam compartment, was allowed to weigh a maximum of 907 grams (2 lbs) as designated by the 2011 AY team, and we allocated 794 grams for the maximum camera weight. The remaining weight was allocated for the camera mount and any other components. Our camera module was designed to operate at a maximum distance of 30.5 meters from the ground, so the camera resolution needed to be standard high-definition (1280x720 pixels) or greater, to allow our mission module to capture quality photos from a distance. Also because of this maximum distance, the camera needed to have a zoom capability of 3x or greater in either optical or digital zoom. Our non-functional goals for the module are to make the module self-powered and fully encapsulated, which we have defined as having no wired connections to other sections of the gondola.

3.4.4 Requirement Chart

Table 5 shows the requirements of each subsystem derived from the goals defined above.

Autonomous Flight Requirements	
AF-01	The flight computer must be able to control the speed of the motors and rotation of the servo
AF-02	The motors and servo must be controlled through the use of PWM signals as used in standard model RC aircraft
AF-03	All control system hardware combined must weigh no more than 907 grams
AF-04	The system must update the blimp's position every second
AF-05	The system must update the blimp's altitude at least every second
AF-06	The system must update the blimp's heading at least every second
AF-07	The blimp must be able to autonomously navigate between two GPS waypoints
Power System Requirements	
PS-01	The power system batteries must have enough charge capacity to supply the blimp with energy for a minimum of 90 minutes with both motors operating at 50% speed
PS-02	The difference between the power supply's voltage and the voltage measured at the left, right, and tail motors should be no more than 0.2V
PS-03	The power system must measure the battery voltages between 0-13V ± 0.1V
PS-04	The power system must measure the current drawn by the left, right, and tail motors between 0-16A ± 0.1A
PS-05	The power system must calculate the average current draw of the left, right, and tail motors accurately to 0.1A during the blimp's flight time
PS-06	The power system must calculate the instantaneous power used by the left, right, and tail motors with a minimum resolution of 0.1W
PS-07	The power system must calculate energy used by the blimp by summing the instantaneous power over the flight time with a resolution of 0.1 Watt-Hours
PS-08	The power system must measure elapsed time with a minimum resolution of 1s
PS-09	The power system must measure the temperature inside the top shelf of the gondola accurately to 0.1°C
Mission Module Requirements	
MM-01	Mission module (excluding foam compartment) must weigh less than 907 grams
MM-02	Mission module must be independent from gondola (no wired connections to other compartments in the gondola)
MM-03	Mission module must have a minimum wireless range of 125 meters
MM-04	Camera must weigh less than 794 grams
MM-05	Camera must have a resolution of 1280x720 pixels (px) or greater
MM-06	Camera must have zoom capability of 3x or greater, either optical or digital zoom

Table 5: Autonomous Flight, Mission Module, and Power System Requirements

3.5 Summary

In this chapter we discussed our goals, needs, and requirements for autonomous flight, the power system, and the camera mission module. We discuss each of these aspects of the blimp's proposed design and the proposed requirements for each sub-system.

4 System Architecture

4.1 Introduction

This chapter describes the architecture of the blimp's systems. First we will discuss a high level overview of the gondola and its sub-systems. Then we will justify the need for each sub-system and the roles they play in the gondola as a whole. Lastly we will go into more detail about the internal architecture and design of each sub-system.

4.2 High Level Systems

The gondola contains multiple sub-systems working in conjunction with each other to propel the blimp, track energy information, and execute missions, as shown in Figure 8. The power supply is used to power each other sub-system with the exception of the mission module, which provides its own power for the device(s) in use. The power display unit includes an interface that displays the power consumption of the gondola. The sensors provide energy usage information to the power display unit. The drive system contains the three motors and the servo for providing propulsion to the blimp. The control system serves as the autonomous flight computer and takes in sensor readings from the environment and commands the motors and the servo. Motor control is the manual control for the motors and servo. The mission module can be different interchangeable mission modules to fulfill different purposes. Lastly, the base station serves as a remote manual control point with the RC controller, and a remote viewing and controlling point for the camera mission module.

4.3 Detailed Sub-Systems

Figure 8 shows a detailed system diagram of each sub-system block. Each of these sub-sections summarizes what is contained within each sub-system block.

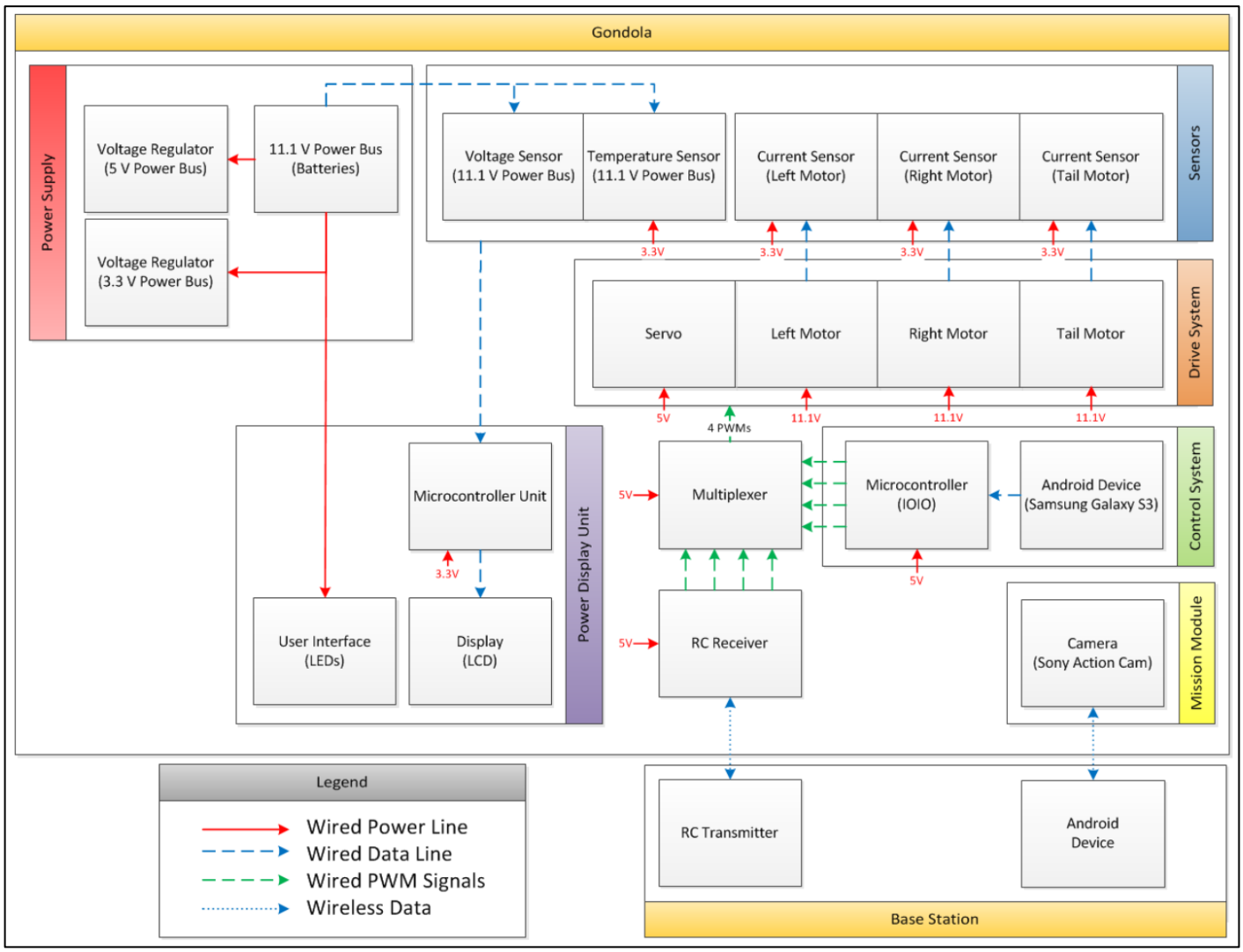


Figure 8: Detailed System Diagram

4.3.1 Power Supply

The power supply is responsible for providing energy to all of the components aboard the blimp, excluding the mission module. The power supply consists of batteries that provide power to the main power bus, and two regulation units that step downs the main voltage to lower levels, which can be used by additional electronics.

4.3.2 Sensors

The sensors are responsible for tracking the energy usage of the drive system. There are three types of sensors: voltage, current, and temperature. The three sensors work in conjunction to acquire data from the motors in the drive system, and then convey the data into form suitable for the power display unit.

4.3.3 Power Display Unit

The power display unit calculates the energy information from the data received from the sensors, while providing an interface for the user to view the energy information. It consists of a microcontroller, liquid crystal display (LCD), and light-emitting diodes (LED). The microcontroller performs the data calculations, the LCD allows the user to read the energy information, and the LEDs provide a quick and intuitive way for the user to assess the batteries' charge levels.

4.3.4 Drive System

For our project we evaluated the components selected for the drive system from the 2011 AY team and decided to leave the components as is for our project. The drive system consists of two brushless motors, a servo, and a brushless bidirectional motor. The two brushless motors are used for the left and right motors. The bidirectional motor is capable of turning in either direction and is used for the tail motor of the blimp to assist in turning. The servo is combined with the 2011 AY 4:1 gear ratio which maps the 0 to 90 degree rotation of the servo to 0 to 360 degrees of rotation. The servo is controlled directly by PWM signals. The drive system also contains three electronic speed controllers (ESC). These ESCs are used to control the amount of current the three motors can draw. The PWM signals sent to the ESCs determine the rotational speed for the two brushless motors, and the speed and direction the bidirectional tail motor turns.

4.3.5 Control System

The control system consists of the Samsung Galaxy S3 Android phone and the Android IOIO microcontroller board. The purpose of the S3 is to run the autonomous control software, to drive the blimp based on sensor readings received from the phone's sensors, and to log all data from the software onto plaintext files on the SD card of the phone.

The S3 connects to the IOIO board through a wired USB connection. The IOIO provides a Java API[3] that can be imported directly into the Java code of the autonomous control software. Using this API, the software running on the S3 is able to make calls to the IOIO and access the board's I/O pins. The autonomous control software reacts based on the sensor readings received from the sensors in the phone, and uses the IOIO to send PWM signals to the drive system motors and servo to drive the blimp.

[3] Application Programming Interface

4.3.6 Manual Override

The manual override is made possible through the use of the multiplexer and the RC receiver. The RC receiver maintains a wireless link to the RC controller that is used to manually control the drive system from the ground. The multiplexer is an 8-to-4 multiplexer which sends the drive system the signals being received from either the RC receiver or the IOIO board. The RC receiver is used for manual control and the IOIO board is used for autonomous control. To switch between manual or autonomous control modes, there is a switch on the RC controller that sends a signal to the multiplexer to change which signals are being sent to the drive system.

4.3.7 Mission Module

The mission module is an extra compartment in the gondola where extras such as a camera or extra batteries can be held. For our project, the focus of the mission module compartment was to have an aerial photography mission module. For this we used a Sony Action Cam with Wi-Fi so that we could take photos and video from the air. Since the Sony Action Cam has Wi-Fi capabilities, we were also able to remotely view and control a live video feed from the ground.

4.3.8 Base Station

The base station contains hardware that is used to wirelessly control the blimp. These components are the RC controller and an Android device. The RC controller is used with the manual override of the control system to allow manual control of the drive system. The Android device is a separate device from the control system, and it is used to wirelessly view and control the video feed from the Sony Action Cam in the mission module.

4.4 Summary

In this chapter we gave a high-level overview of the architecture of the blimp system. We showed a diagram of and described each section of the system.

5 Control System

5.1 Introduction

This chapter covers the design, implementation, and testing of the hardware used for the control system of the blimp. First we discuss the overall design for the control system. Next we discuss our needs and requirements for the control system. After we discuss our hardware and system requirements we explain our options for our control system. Lastly, we explain our final design and the results.

5.2 System Design

The control system is the hardware that controls the drive system of the blimp. Figure 9 shows the high-level design of the control system, including the 2011 AY team's multiplexer and RC receiver, and how the control system interacts with the drive system. The drive system of the blimp consists of four parts: a left and right motor, a bi-directional tail motor, and a servo. The left and right motors combined with the tail motor enable the blimp to turn. Additionally, combining the left and right motors with the servo allows the blimp to increase or decrease in altitude. To implement an autonomous control system for the blimp, we needed logic that would take commands generated from a program and generate proper signals to control the drive system.

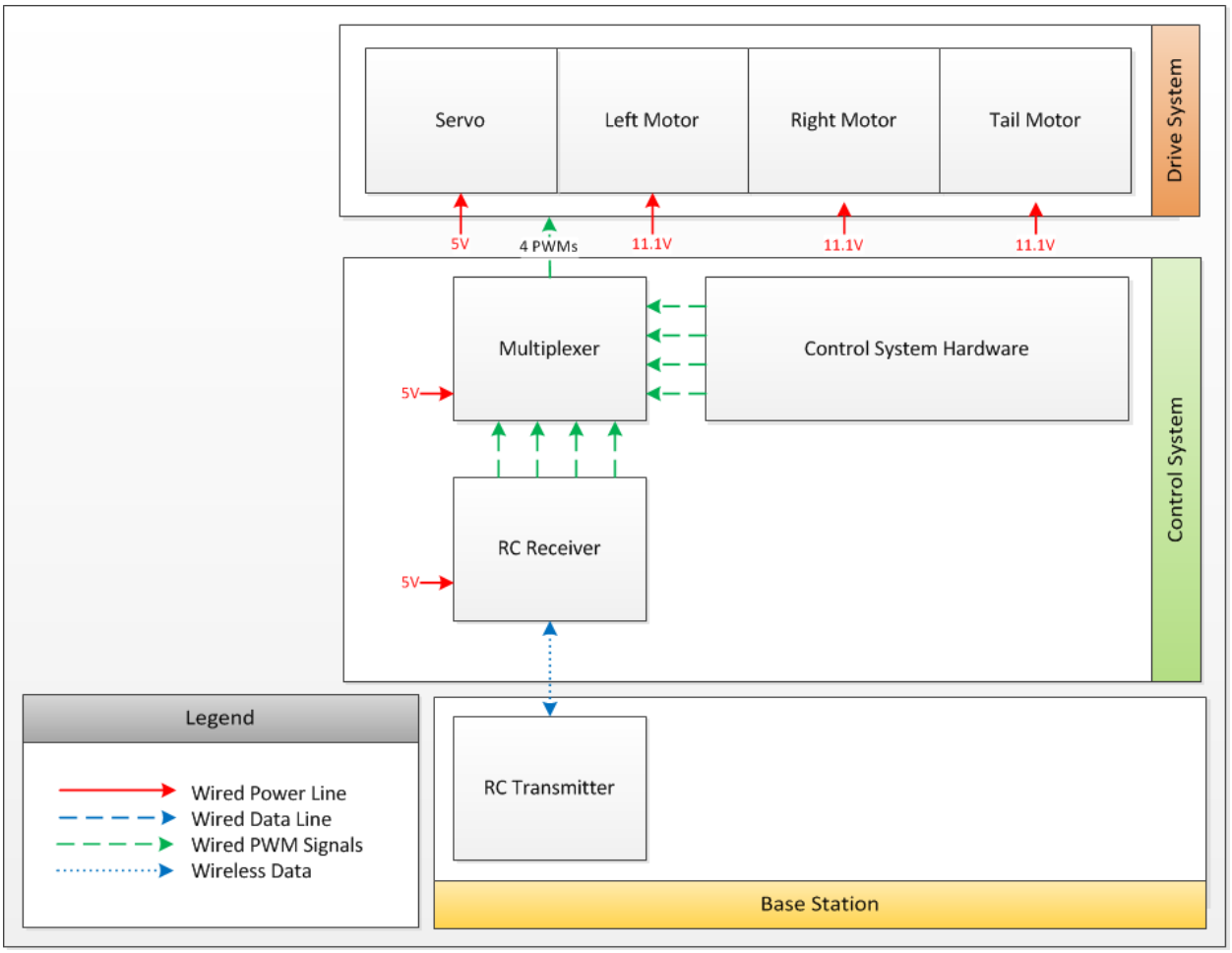


Figure 9: High Level Control System Diagram

For our project we modified the control system by removing the 2011 AY team's control system hardware and external sensor components. The control system contains hardware to generate and send PWM signals to an 8-to-4 multiplexer. The multiplexer determines which signals to send to the drive system – either the 4 PWM signals generated from the control system hardware, or the 4 PWM signals generated from the RC receiver.

5.2.1 System Needs

Before we were able to determine our hardware constraints, we first had to evaluate the hardware used by the 2011 AY team for the control system. A detailed system diagram for the control and drive system are shown in Figure 10. The hardware and circuits used for the control system consisted of an 8-to-4 multiplexer, Gumstix RoboVero and Overo Water, a 6-Channel RC receiver, and several external sensors which included a GPS and an altimeter.

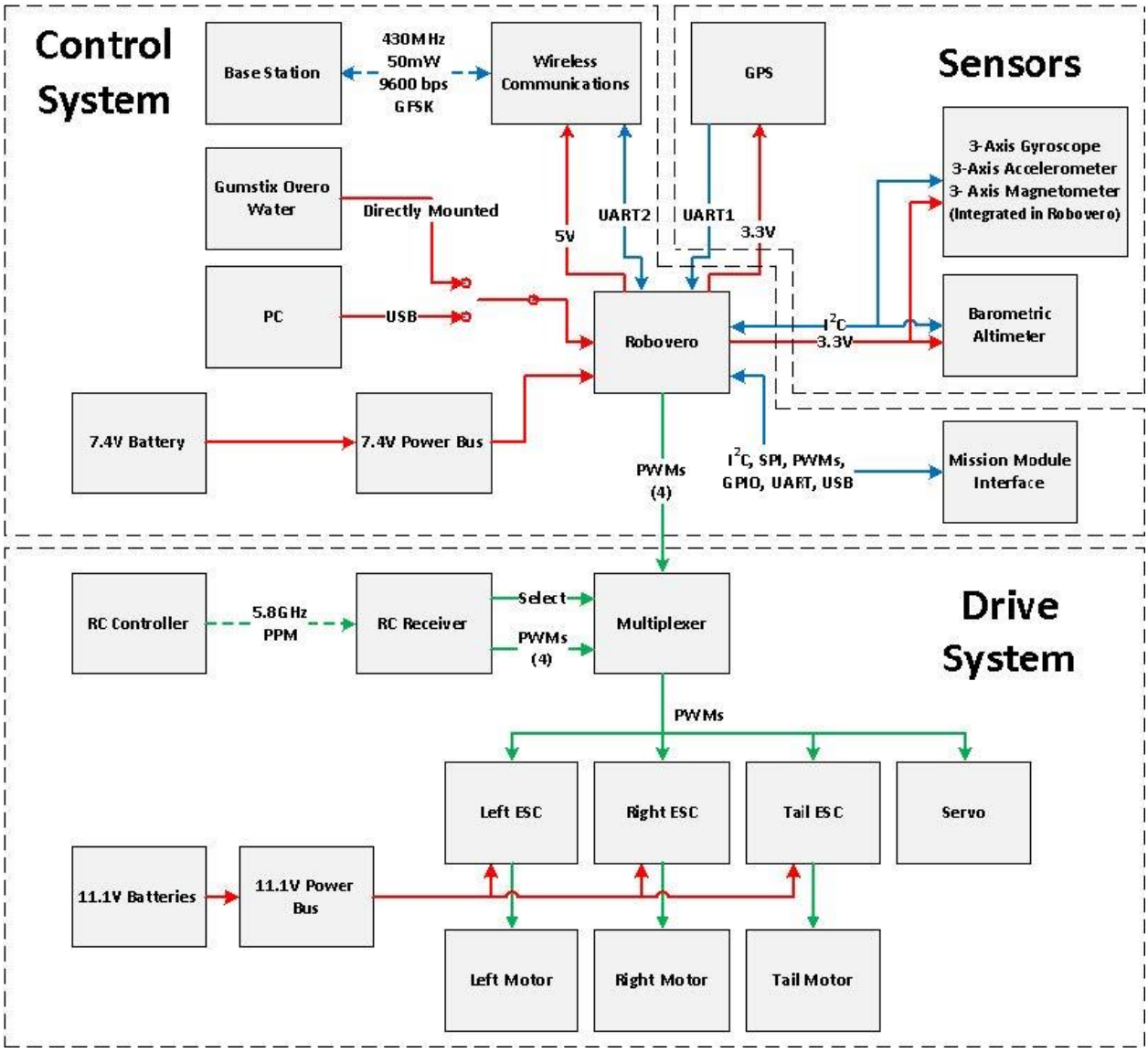


Figure 10: 2011 AY Control and Drive System Design [1]

The Gumstix RoboVero is a hardware component whose primary purpose was to run the autonomous program designed by the 2011 AY team. Additionally, the Gumstix RoboVero contained header pins to send PWM signals that the program generated to control the blimp's drive system. The Gumstix Overo Water was an expansion board for the Gumstix RoboVero that contained the accelerometer, gyroscope, and magnetometer sensors. The control system also contained an external GPS component which was used to determine the location of the blimp. While the GPS component could provide a calculated altitude value, the 2011 AY team used a barometric altimeter to get a more accurate altitude reading for the blimp. Both the GPS and barometric altimeter components sent their readings to the Gumstix Overo Water.

The 2011 AY team also used an RC receiver in their control system. The RC receiver was used so that if necessary, the team members could manually take control of the blimp's drive system while mid-flight. To utilize the signals sent from the RC receiver, and to implement a control mechanism, the 2011 AY team designed a circuit for an 8-to-4 multiplexer to control which set of signals were sent directly to the drive system, either the signals generated from the Gumstix Overo Water, or the RC receiver.

After evaluating the 2011 AY team's components, we determined the specific system needs for the control system. Any changes to the 2011 AY team's control system would still need to have the following functionalities:

- a sensor or component to detect altitude
- a component or sensor that could determine the blimp's global position
- software or hardware component to process sensor information
- hardware capable of generating PWM signals to directly control the drive system
- an RC receiver (for manual control/override)
- a multiplexer component to control which signals are sent to the drive system

In addition to technical needs for the new control system, the weight for the hardware had to be equal to or less than the 2011 AY team's control system hardware.

5.2.2 System Requirements

After reviewing the needs for our control system, we determined the requirements for the control system. Table 6 shows a summary of the specifications regarding the 2011 AY team's hardware used for the control system. Based on the combined weight of all of the previous components, one requirement was that the new control system weighs no more than 162.2 g (0.36 lbs.). The Gumstix RoboVero had a microSD card for memory storage and a 1x720MHz core processor with 512MB of RAM, so our new processor must have equal or greater processing power, on-board memory, and external storage capacity. The Gumstix Overo Water not only contained the gyroscope, accelerometer, and magnetometer sensors, but also was capable of wireless signal transmission, specifically Wi-Fi and Bluetooth. As a result, one of our requirements was to have wireless transmission capability (IEEE 802.11 wireless local area network), and we needed to have gyroscope, accelerometer, and magnetometer sensors that at least met, if not exceeded, the accuracy of the previous sensors.

To implement an autonomous algorithm capable of navigating the blimp between successive waypoints, we needed a GPS and an altimeter component, preferably one that equaled or exceeded the accuracy of the 2011 AY team's GPS and altimeter components. Another requirement was that the control system must be independently powered and be able to sustain operation for a minimum of 90 minutes, our endurance flight time goal. Lastly, the Gumstix RoboVero had header pins that were used to send PWM signals to the drive system at a frequency of 45Hz, so one control system requirement was that we needed a hardware component capable of producing identical PWM signals [1].

Categories	Control System Specifications
Processor	1 x 720MHz
Memory (RAM)	512MB
Storage	MicroSD
Wi-Fi	Yes
Bluetooth	Yes
USB	Yes
Pin I/O Layout	Header Pins
Accelerometer/Gyroscope	+/- 250-2000 deg/sec
Magnetometer	Yes
Altimeter	Yes
GPS	Tracking Sensitivity: -160dBm Cold Start Sensitivity: -143dBm Startup Time: 29 sec Chip Rate: 1.023MHz
Battery Life	1.43 hours
Weight	162.2 g

Table 6: 2011 AY Control System Hardware Specifications [1]

From the system needs stated above, the following requirements for the control system were derived. Each need was broken down into quantifiable details, and is shown in Table 7.

Control System Requirements	
CS-01	The control system components in total must weigh no more than 907 g (2 lbs.).
CS-02	The control system must have a GPS receiver capable of determining the blimp's location accurately within 4 meters.
CS-03	The control system must be capable of determining the blimp's altitude accurately within 1 meter.
CS-04	The control system must have an inertial measurement unit (IMU) with a minimum accuracy of +/- 250-2000 degree/sec.
CS-05	The control system must have a processor with a minimum speed of 720MHz.
CS-06	The control system must have a hardware component capable of generating a minimum of 4 PWM signals simultaneously based on standard RC aircraft PWM signal specifications
CS-07	The control system must be able to select which signals are sent to the drive system, either from the RC receiver (manual control), or from the processor device (autonomous control).
CS-09	The control system must have at least 512MB of memory (RAM).
CS-10	The control system must have external storage to store logged information.

Table 7: Control System Requirements

5.3 System Architecture and Trade Studies

Once we determined our control system requirements, we researched various hardware components to determine what would meet our requirements. Based on current smartphone specifications, we determined that using an Android smartphone would meet most of our weight, power, and sensor requirements. Additionally, Android OS is open-source, which would allow for more convenient programming because Google provides APIs to access sensor information. As of writing this report Android version 2.3, "Gingerbread", of the Android operating system (OS) comprises about 57.2% of the market for Android devices. Android 4.0, "Ice Cream Sandwich", (ICS) is projected to be the next dominant Android OS version, and currently makes up 28.6% of the market [9]. Also, ICS has the functionality to be a USB host. This means that the Android device is able to power and enumerate all connected USB devices [10]. Gingerbread lacks this functionality.

To replace the hardware components with an Android device, we first researched what was currently available on the market for Android devices so that we could determine what capabilities our device could have. We wanted the Android device to contain all of the sensors used by the 2011 AY team, which included the GPS, altimeter, accelerometer/gyroscope, and magnetometer. After researching numerous Android devices released from 2010 to 2012, we

created a list of specifications that described the standard for Android devices. The standard hardware specifications for Android devices can be seen in Table 8.

Categories	Standard Android Device
Processor	1 GHz Dual-Core
Memory (RAM)	1 GB
Operating System	4.0 (Ice Cream Sandwich)
Storage	MicroSD
Wireless	Wi-Fi, Bluetooth
Connection	Micro-B USB, 3.5mm Audio Jack
Accelerometer/Gyroscope	+/- 250-2000 deg/sec
Magnetometer	Yes
Barometer	No
GPS	Yes
Battery	1780-1850 mAh
Weight	141.7 g

Table 8: 2012 Standards for Android Devices [3]

To meet hardware requirements determined in Section 5.2.2 for sensor readings, processor speed, and internal and external memory storage, the two Android devices we considered were the HTC Evo 4G LTE (Evo) and the Samsung Galaxy S3 (S3). Table 9 shows a direct hardware specification comparison between the 2011 AY team's control system hardware, the Evo, and the S3. In terms of hardware specifications, both the Evo and the S3 excel compared to the previous control system hardware. In comparison to each other, the Evo and S3 specifications do not differ greatly. Both the Evo and S3 have dual-core processors which run at 1.5GHz, a microSD card slot, 16 GB of internal memory, Wi-Fi, Bluetooth, micro-B USB connection, magnetometer, and the same sensor accuracy for the accelerometer/gyroscope and the GPS.

Categories	Previous Control System [1]	HTC Evo 4G LTE [11]	Samsung Galaxy S3 [12]
Processor	1 x 720MHz	2 x 1.5 GHz	2 x 1.5GHz
Memory (RAM)	512MB	1GB	2GB
Storage	MicroSD	MicroSD, 16GB internal	MicroSD, 16GB internal
Wi-Fi	Yes	Yes	Yes
Bluetooth	Yes	Yes	Yes
USB	Yes	Yes (micro-B USB)	Yes (micro-B USB)
Pin I/O Layout	Header Pins	None	None
Accelerometer/Gyroscope	+/- 250-2000 deg/sec	+/- 250-2000 deg/sec	+/- 250-2000 deg/sec
Magnetometer	Yes	Yes	Yes
Altimeter	Yes	No	Yes
GPS	Tracking Sensitivity: -160dBm Cold Start Sensitivity: -143dBm Startup Time: 29 sec Chip Rate: 1.023MHz	Tracking Sensitivity: -154dBm Cold Start Sensitivity: -130dBm Startup Time: 32 sec Chip Rate: 20MHz	Tracking Sensitivity: -154dBm Cold Start Sensitivity: -130dBm Startup Time: 32 sec Chip Rate: 20MHz
Battery Life	1.4 hours	1.5 hours	1.6 hours
Weight	162.2 grams	134 grams	133 grams

Table 9: Specifications Comparison between Previous Control System, Evo, and S3

The HTC Evo 4G LTE and the Samsung Galaxy S3 do not have header connection pins and are not capable of generating PWM signals, so we needed to research possible MCUs that are capable of communicating with the smartphone and generating PWM signals to control the drive system. For the MCU we considered an Arduino Leonardo and the Android IOIO. Both the Arduino Leonardo and the IOIO are capable of generating PWM signals that could be sent to the drive system. One difference between these MCUs is that the Arduino Leonardo could not receive direct instructions from an Android device through the USB port, and instead would have to use an alternative method to control what PWM signals the Arduino Leonardo generated. The Android IOIO, however, came with a Java API that would allow an Android smartphone to communicate directly with the IOIO board through the USB port.

5.4 Final System Design

After comparing the HTC Evo 4G LTE to the Samsung Galaxy S3, we selected the Samsung Galaxy S3, shown in Figure 11, for our control system processor. There were numerous reasons for choosing the S3 over the Evo. One reason was that the S3 has a longer battery life, which would result in a longer possible flight time for the blimp. Additionally, the S3 has 2GB of memory, while the Evo only has 1GB of memory. One advantage of having more memory is that it could be useful for future teams to utilize the phone's capabilities. While the sensor accuracy was the same for all of the sensors that the Evo and S3 had in common, the S3 proved to be the more suitable choice because it had a barometer, which can be used to calculate altitude.

Figure 11: Samsung Galaxy S3

After comparing the Arduino Leonardo and Android IOIO, we decided to use the Android IOIO, shown in Figure 12. While both MCUs have header pins and are capable of generating PWM signals, one important reason for choosing the IOIO was that it could communicate directly to the smartphone through the USB port. Being able to utilize the Java API

that came with the IOIO allowed our team to create a program for our autonomous algorithm so that we could have the S3 send commands to the IOIO, and the IOIO could then produce appropriate PWM signals which can control the drive system.

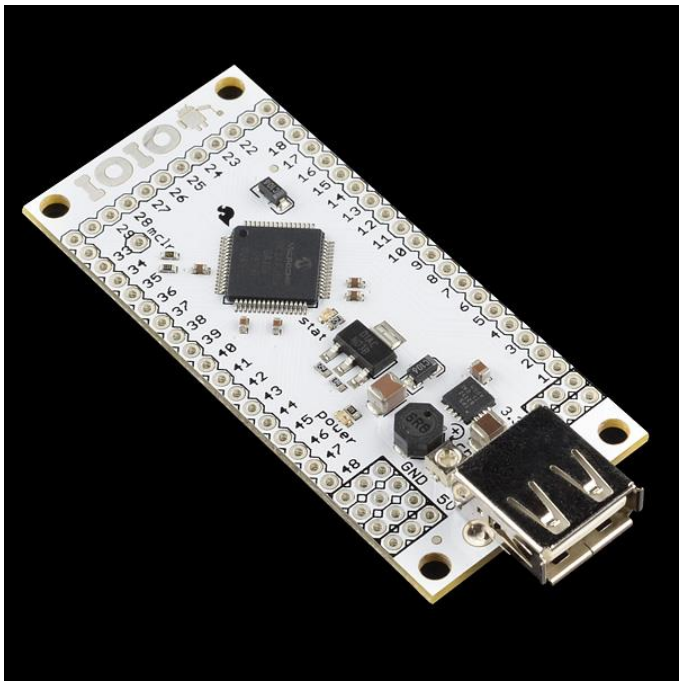


Figure 12: Android IOIO

Since our team needed to include a manual override system, we determined that it would not be necessary to change or remove the RC receiver used by the 2011 AY team. Because we intended to use the previous RC receiver, we also determined that we could use the 2011 AY team's 8-to-4 multiplexer to select which signals are sent to the drive system, either the signals from the RC receiver, or the PWM signals generated from the Android IOIO.

5.5 Detailed System Design

Once we determined that we were going to use a Samsung Galaxy S3 for sensor reading information and the Android IOIO to connect to the S3 to generate PWM signals, we needed to determine how to combine these two components. To do this we needed to incorporate the multiplexer and the RC receiver. All of these components combined form the control system for the blimp for both autonomous and manual control.

5.5.1 Detailed System Design Architecture

Figure 13 shows the detailed system design for the control system. One important component of the control system is the Samsung Galaxy S3. The S3 contains a Qualcomm Snapdragon 1.5GHz dual-core processor, which also contains the GPS receiver. In addition to the GPS receiver, the S3 has a barometer, 3-axis gyroscope, accelerometer, and magnetometer. With these sensors and the GPS receiver the S3 processor reads in sensor readings and then processes this information to determine what signals should be sent to the drive system to control the blimp autonomously. To interface with the phone we use the touch screen display to manually enter in GPS coordinates, as well as to observe sensor readings. All components in the S3 are powered directly from a 2100 mAHr lithium-ion battery.

After the S3 has taken in sensor readings and processed them, the S3 communicates with the Android IOIO to generate PWM signals for the drive system. The S3 connects to the Android IOIO via a USB cable. Once the S3 has a connection to the IOIO, the IOIO is then capable of generating appropriate PWM signals. We originally intended for the IOIO to be powered by the S3, but the IOIO had to be powered independently of the S3 because the IOIO cannot be powered directly by the S3. Instead, the IOIO is powered by the 5V rail on the multiplexer PCB, which we discuss in Section 5.6.3. The IOIO is powered by the 5V rail, which results in the PWM signals having a peak value of 5V.

The IOIO uses I/O pin numbers 3, 5, 7, and 10 for PWM output signals. These I/O pin connections correspond to Throttle, Rudder, Aileron, and Elevator respectively. These signals control the servo, tail motor, left motor, and right motor, respectively. To simulate the PWM signals sent from the RC receiver, the PWM signals generated by the IOIO have a pulse width that ranges from 1ms to 2ms and are generated at a frequency of 45 Hz. The pulse width of the PWM signal the IOIO sends to either the servo or the ESCs for the motors determines what the drive system does. The servo can be rotated within the range of 0 to 90 degrees. Since the drive system from the 2011 AY team used a 4:1 gear ratio, this mapped the 0 to 90 degree range to a 0 to 360 degree range. The tail motor is capable of spinning in either direction depending on what signal the tail motor ESC is given. Both the left and right motors are independently capable of being set to a speed ranging from off to their maximum rotation speed depending on the PWM signals sent to their respective ESCs.

Since both the RC receiver and the Android IOIO are capable of generating four similar PWM signals to control the drive system, we needed to implement a multiplexer to determine whether the blimp was in manual mode (RC Receiver) or autonomous mode (S3). To do this we implemented an 8-to-4 multiplexer that takes in the eight PWM signals generated by the RC receiver and the IOIO, four from each component. To determine whether the blimp was under manual control or autonomous control, an additional switching signal is sent to the multiplexer to select which set of four signals are sent to the drive system. This switching signal, the Gear signal, is generated from the RC receiver. If the Gear switch on the RC receiver is flipped, the blimp changes modes.

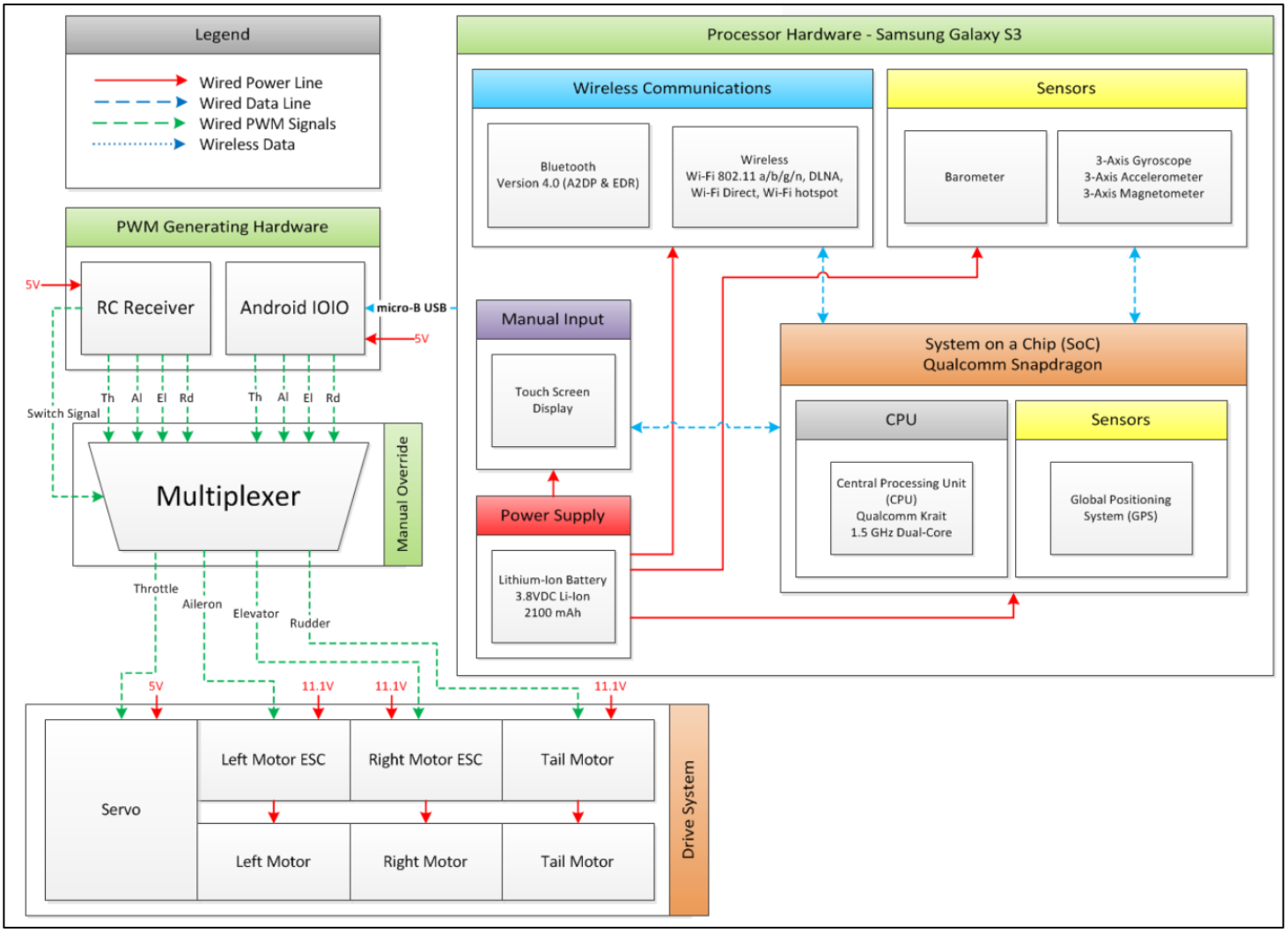


Figure 13: Detailed Control System Diagram

5.5.2 Detailed Manual Override System

Figure 14 shows the high level block design for the multiplexer portion of the control system. While a majority of the multiplexer schematic follows closely to what the 2011 AY team designed, we altered some components. We modified the gain stage from a gain of seven to six. The multiplexer system can be broken down into four main functional blocks: Low-Pass Filter, Gain Stage, Comparator, and Multiplexer.

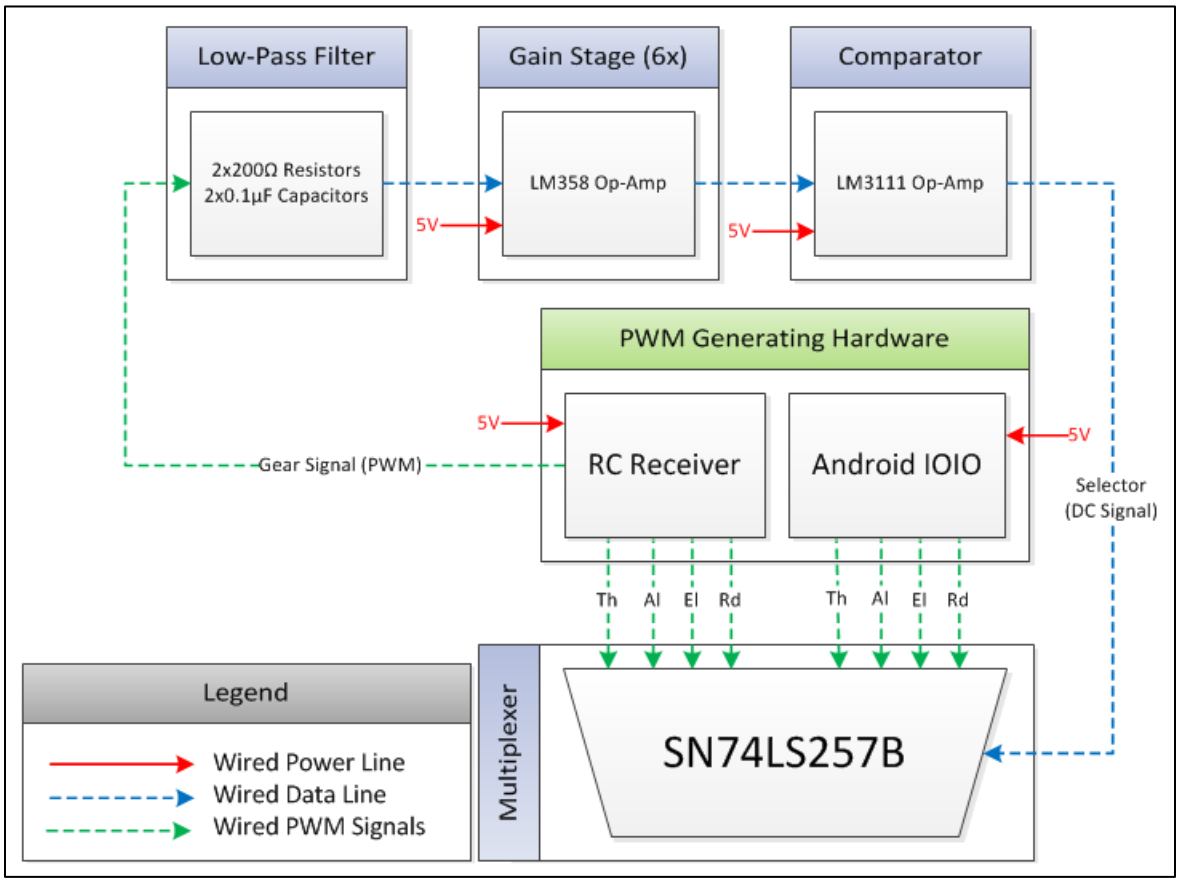


Figure 14: Block Diagram for Multiplexer System

Figure 15 shows the schematic configuration for the Low-Pass Filter portion of the multiplexer system. The Low-Pass Filter takes in the Gear signal from the RC receiver. The purpose of the Low-Pass Filter is to slow the response of the PWM signal to produce a DC signal. The DC signal is then sent to the next block of the multiplexer system, the Gain Stage.

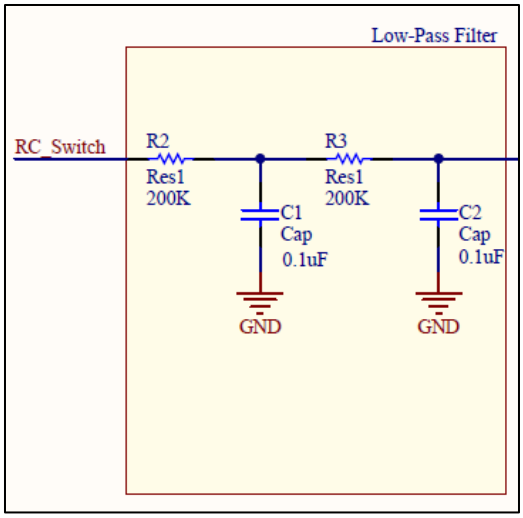


Figure 15: Schematic for Low-Pass Filter Block

Figure 16 shows the schematic configuration for the Gain Stage of the multiplexer system. Once the Gear signal is in the form of a DC signal, we pass the signal into an LM358 op-amp. By using a 5.1kΩ resistor with a 1kΩ resistor to form a voltage divider, we configure the negative feedback for the op-amp in such a way that we get a gain of approximately six. Using this op-amp configuration we amplify the DC signal by a magnitude of six for easier comparison later on in the comparator block.

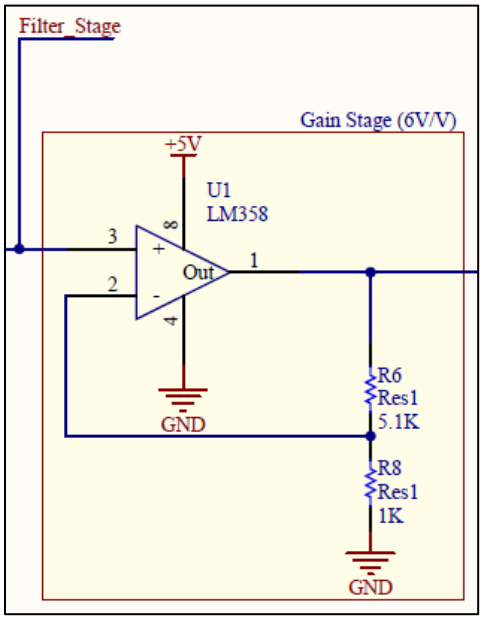


Figure 16: Schematic for Gain Stage

Figure 17 shows the schematic configuration for the Comparator portion of the multiplexer system. Once the signal has been amplified we pass the signal into an LM311 op-amp, which is configured to act as a comparator. By configuring the op-amp to use a

potentiometer we can specifically determine what voltage value we are comparing the amplified DC signal to. If the DC signal exceeds our set voltage reference for the comparator, then the output of the comparator is pulled high and is set to 5 volts, which is also the switching signal. If the DC signal is less than the set voltage reference the output of the comparator, the switching signal, is set to 0 volts. By configuring the op-amp to have a positive feedback using the 1kΩ resistor and 30kΩ resistor we create a threshold range where the output of the op-amp won't instantaneously change if the input to the op-amp (from the gain stage) is greater or less than the voltage reference by an insignificant value. This error threshold range was included to account for any jitter or noise that might be sent into the op-amp that would not accurately indicate if the control system changed from manual to autonomous control or vice versa.

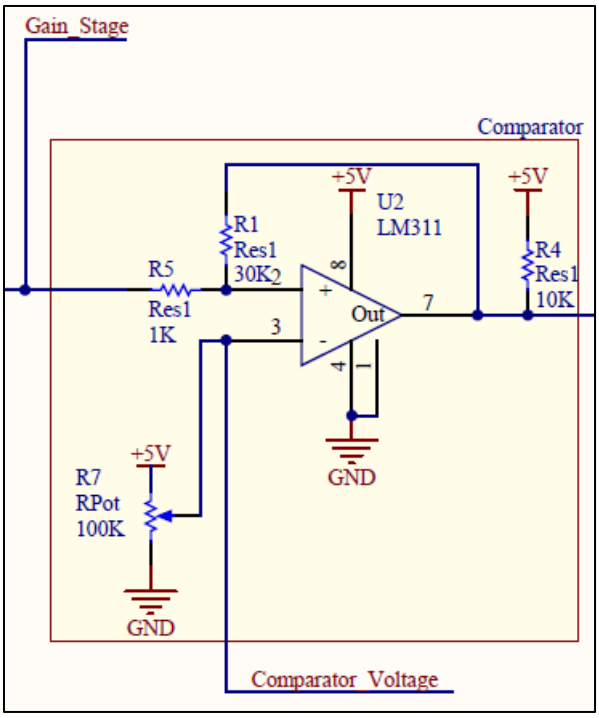


Figure 17: Schematic for Comparator Block

Figure 18 shows the schematic layout for the multiplexer component for the multiplexer system. For this portion of the system an 8-to-4 SN74LS257B multiplexer was used. Once the switching signal has been determined to be either a high (5 volts) or a low (0 volts), the switching signal is then passed into the multiplexer. If the switching signal is high, then the multiplexer will allow the PWM signals generated from the Android IOIO to be sent to the drive system components. If the switching signal is low, then the PWM signals generated from the RC

receiver will be allowed to pass through the multiplexer and to the drive system components. The switching signal is low by default when the system is initialized, regardless of the Gear signal.

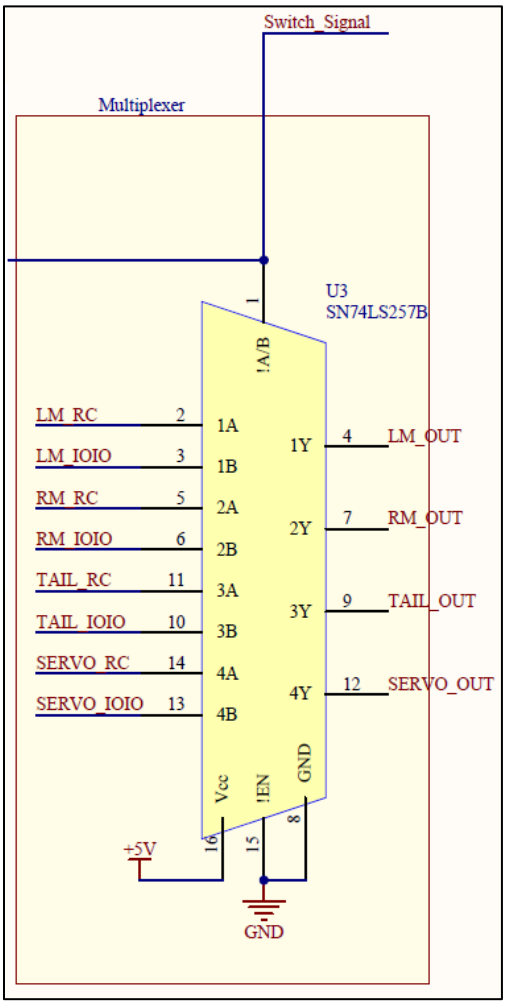


Figure 18: Schematic for Multiplexer Block

All of these blocks combined will allow the Gear signal from the RC receiver to determine if the PWM signals sent to the drive system are generated from manual control (RC receiver) or from autonomous control (Android IOIO). We converted the four blocks of the multiplexer system into a Printed Circuit Board (PCB), shown in Figure 19. The multiplexer PCB provides rails for both ground and 5 volts, which are provided by the power board, which is discussed in a later section of this report. Since the Android IOIO, RC receiver, the servo, and some components in the multiplexer system need 5 volts to operate, the PCB has 5 volt rails to power these components.

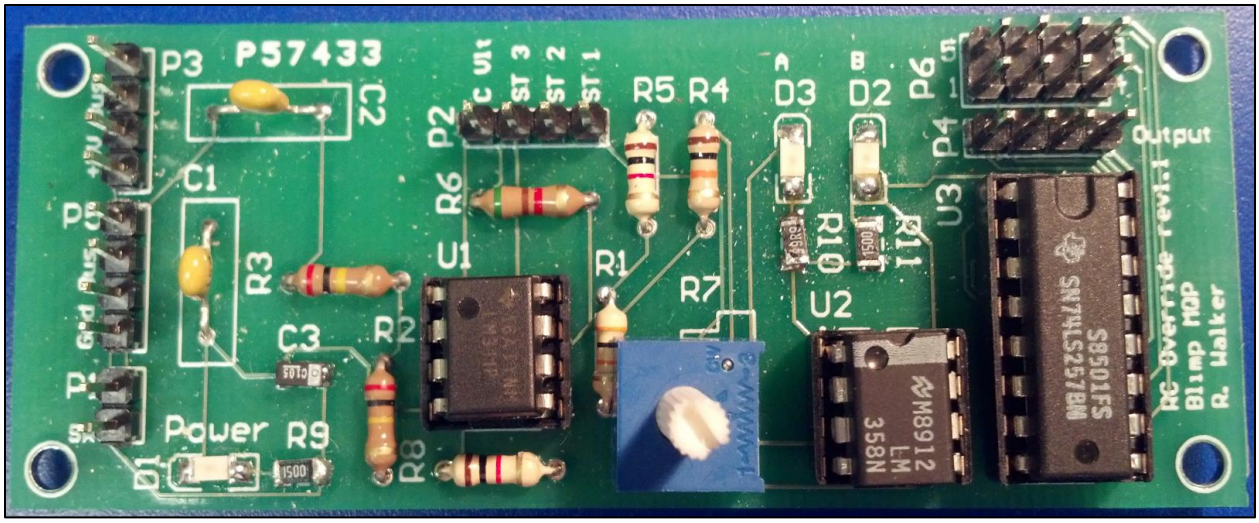


Figure 19: PCB of Multiplexer Board

5.6 Testing and Results

To test and verify our results for the control system, we first needed to identify what specific signals were used to control the drive system. For our project we decided to keep the 2011 AY team's drive system, and thus we needed to understand what signals were generated by the RC receiver so we could generate similar signals using the Android IOIO with the Samsung Galaxy S3.

5.6.1 Testing and Results

To observe PWM signals sent from the RC receiver we first had to identify each specific signal generated by the RC receiver. The detailed table for PWM signals for each specific signal can be seen in Appendix I. Using an oscilloscope, we were able to observe each signal individually including the Throttle, Aileron, Elevator, Rudder, and Gear signals. We observed the pulse width for each signal while moving the analog sticks of the RC controller. Figure 20 shows an example of a screen capture of the oscilloscope verifying PWM signal pulse width, and the frequency of these signals is approximately 45Hz.

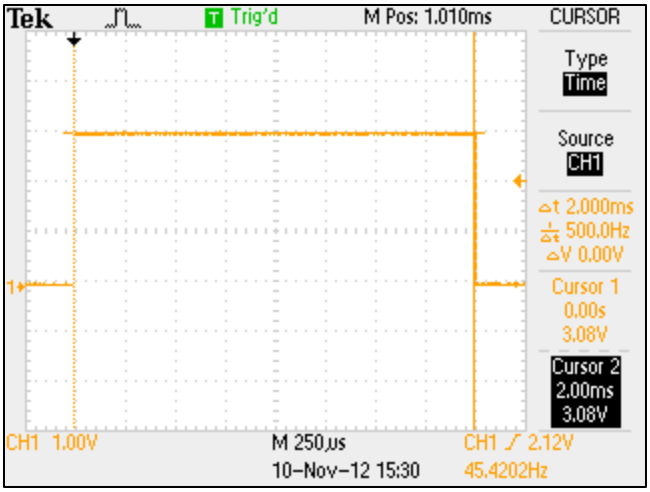


Figure 20: Sample PWM Signal on Oscilloscope

From our research we were able to observe that each PWM signal generated was at a frequency of approximately 45Hz, and ranged between 1ms and 2ms pulse width. Appendix I goes into further detail on specific information regarding the signals, and Table 10 summarizes the ranges for each drive system component and signal.

Signal	Drive Component	Range (Min)	Drive System Function (Min)	Range (Max)	Drive System Function (Max)
Throttle	Servo	1ms	0 Degrees	2ms	90 Degrees
Aileron	Tail Motor	1-1.5ms	Motor On Clock Wise (Power varies)	1.5-2ms	Motor On Counter-Clock Wise (Power varies)
Elevator	Left Motor	1-1.3ms	Motor On (0% Power)	2.0ms	Motor On (100% Power)
Rudder	Right Motor	1-1.3ms	Motor On (0% Power)	2.0ms	Motor On (100% Power)

Table 10: PWM Signal Summary

5.6.2 Tests and Results for PWM Signals Generated by Android IOIO

Now that we documented and identified all the possible ranges and functionality for the PWM signals generated by the RC receiver, we then had to make the IOIO generate similar PWM signals. To generate various PWM signals from the IOIO, we wrote an app for the S3 called "BlimpPWMTest." A screenshot for this test app can be seen in Figure 21. This test app was programmed so that different PWM signals were sent from the I/O pins on the IOIO board depending on which buttons were pressed. Table 11 shows a detailed pulse width correlation for the signals sent by the IOIO depending on the settings in the test app.

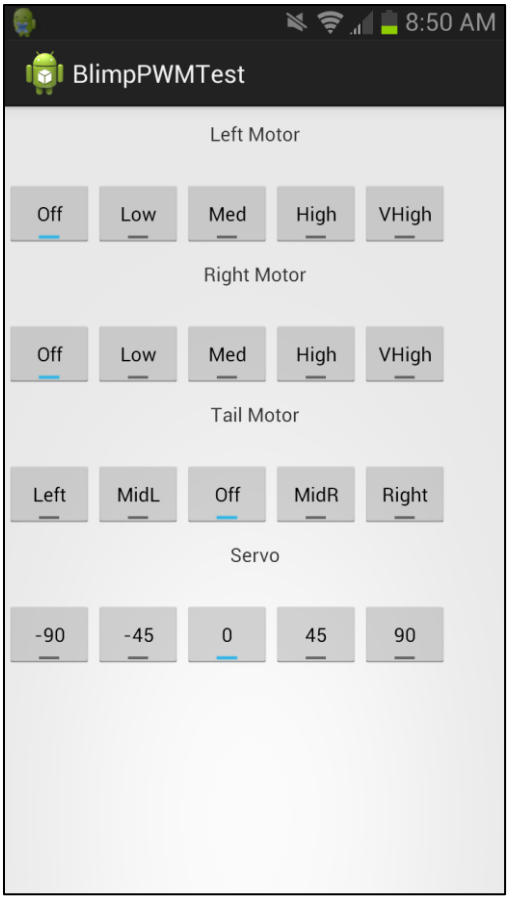


Figure 21: Screenshot for BlimpPWMTest App

		Signal (Pulse Width)				
		1ms	**1.25ms**	**1.5ms**	**1.75ms**	**2.0ms**
App Control	**Left Motor**	Off	Low	Med	High	VHigh
	Right Motor	Off	Low	Med	High	VHigh
	Tail Motor	Left	MidL	Off	MidR	Right
	Servo	-90	-45	0	45	90

Table 11: PWM Signal Correlation to BlimpPWMTest App Buttons

To verify that our signals generated by the IOIO were the expected signals, we observed the signals sent by the IOIO on an oscilloscope. Figure 22 shows that the current configuration for what the PWM signals should be were as follows:

- Left Motor – High
- Right Motor – VHigh
- Tail Motor – Off
- Servo – -90

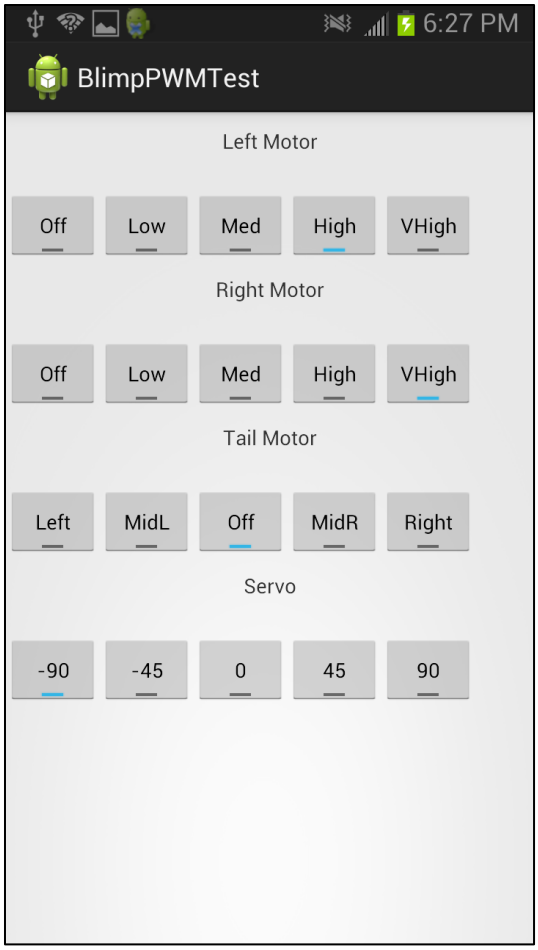


Figure 22: Screenshot for BlimpPWMTest App with Specific Configuration

Using Table 11 we should see the IOIO produce signals that correlate to the input configurations for the test app. The signals for the drive system should be as follows:

- Left Motor (Purple Signal) – 1.75ms
- Right Motor (Green Signal) – 2ms
- Tail Motor (Blue Signal) – 1.5ms
- Servo (Yellow Signal) – 1ms

Figure 23 shows the observed PWM signals on the oscilloscope. Figure 23 also contains two vertical yellow bars that represent the placement of 0ms and 2ms ranges. Looking closely at each of the signals we can verify that we see the expected pulse widths for this specific configuration for our test app. It should be noted that not all of the PWM signals are perfectly in sync at the 0ms marker because there is a natural hardware delay that prevents all the signals from being synced and triggered at the same time, but the difference between their set generated times are small, approximately 100 µsec.

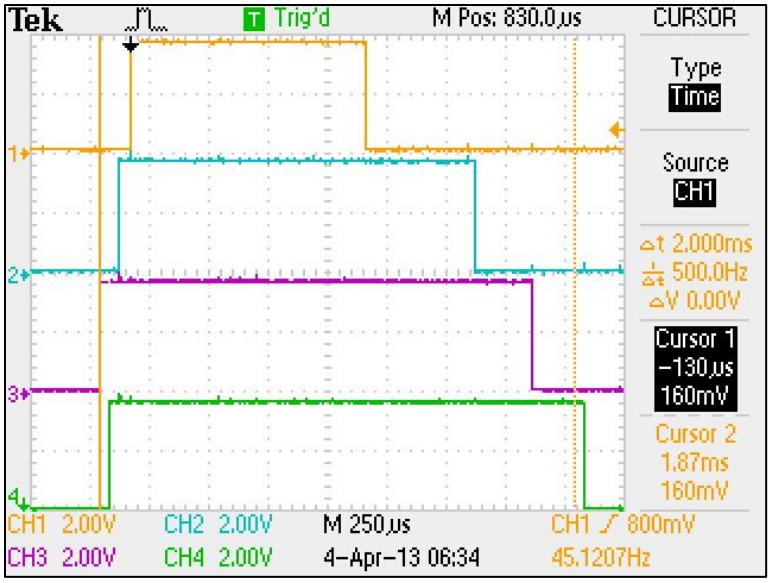


Figure 23: Signals from IOIO for BlimpPWMTest App Configuration

From various configuration tests using our BlimpPWMTest app we were able to verify that we can properly produce the signals we desire to send to the control system. After verifying that the PWM signals are as expected, we then connected the motor ESCs and servo directly to the IOIO and verified that we could manually control the drive system components using our BlimpPWMTest app.

5.6.3 Tests and Results for Manual Override System

We needed to test the multiplexer system to verify that it worked as expected. To test the multiplexer functionality we connected the signals from the RC receiver and the Android IOIO to the 8-to-4 multiplexer. Then we connected the Gear signal from the RC receiver to the input connection for the Low-Pass Filter portion of the multiplexer system. We then observed the output signals of the multiplexer on four separate channels on an oscilloscope. We verified that when we flipped the Gear switch on the RC controller, which corresponds to a 1ms or a 2ms pulse width, we could see the expected PWM signals. When the Gear switch was triggered on, we observed signals generated by the RC receiver. When the Gear switch was triggered off, we observed signals generated by the Android IOIO.

It should be noted that there were two mistakes in the original multiplexer PCB design. One mistake was that the pin-out for the potentiometer was incorrect. The other mistake was that a trace connecting pin 4 of the op-amp (U2 in the PCB design) to ground was missing. For our

project we manually corrected these mistakes by soldering wires to connect the correct pins. These mistakes have been corrected in an updated PCB design, but this PCB has not been printed.

5.6.4 Fulfillment of Requirements

This section will review the requirements for the control system, how each was tested, and whether the test passed or failed.

ID	Requirement	Need	Test	Result
CS-01	The control system components in total must weigh no more than 907 g (2 lbs.).	Maximum weight allotment for control system.	Weighed the Samsung Galaxy S3, Android IOIO, RC Receiver, Multiplexer PCB, and jumper wires.	Met, 586g
CS-02	The control system must have a GPS receiver capable of determining the blimp's location accurately within 4 meters.	To find the GPS location of the blimp to use for the autonomous algorithm.	Researched the datasheet specifications for the Samsung Galaxy S3 to verify GPS receiver and its accuracy.	Met
CS-03	The control system must be capable of determining the blimp's altitude accurately within 1 meter.	To determine the blimp's altitude for the autonomous algorithm.	Researched the datasheet specifications for the Samsung Galaxy S3 to verify barometer sensor and used this sensor reading to determine a resolution for altitude height.	Met
CS-04	The control system must have an inertial measurement unit (IMU) with a minimum accuracy of +/- 250-2000 degree/sec.	To meet sensor accuracy for the 2011 AY team's component, and to utilize sensor readings for the autonomous algorithm.	Researched the datasheet specifications for the Samsung Galaxy S3 to verify IMU sensor and its accuracy.	Met
CS-05	The control system must have a processor with a minimum speed of 720MHz.	To meet processor speed for the 2011 AY team's processor, and to process sensor readings in real time for the autonomous algorithm.	Researched the datasheet specifications for the Samsung Galaxy S3 to verify processor had a minimum speed of 720MHz.	Met

CS-06	The control system must have a hardware component capable of generating a minimum of 4 PWM signals simultaneously based on standard RC aircraft PWM signal specifications.	The four simultaneous PWM signals must be generated to control the four drive components: Servo, Tail Motor, Left Motor, and Right Motor.	Connected the Android IOIO I/O pins 3, 5, 7 and 10 to an oscilloscope to verify if the IOIO was capable of simultaneously generating four PWM signals to control the drive system.	Met
CS-07	The control system must be able to select which signals are sent to the drive system, either from the RC receiver (manual control), or from the processor device (autonomous control).	The FAA mandates that that all autonomous aircraft have a two way link to ground control that overrides automatic controls for safety reasons. [1]	Constructed and built a multiplexer system, and verified by looking at the output signals of the multiplexer on a four channel oscilloscope to verify whether RC receiver signals (manual control) or the IOIO signals (autonomous control) were sent to the output of the multiplexer based on the manual switch flipped on the RC controller.	Met

Table 12: Requirements Fulfillment Table for Control System

5.7 Summary

In this chapter we discussed the design, implementation, and testing of the control system for the blimp. We explained the needs and requirements of the system, and then we discussed the architecture for the control system. We also discussed the implementation for the multiplexer and verified the manual override control for the control system. Finally, we discussed the results we found through testing the control system, and we were able to verify that the control system works as expected and the control system properly controls whether we are manually or autonomously in control of the drive system.

6 Autonomous Flight

6.1 Introduction

This chapter covers the design, implementation, and testing of the software written for autonomous flight. First, we discuss the overall design of the software. This includes system diagrams, high level algorithms, general code organization, some key design choices, the needs of the system, and the system requirements. Second, we discuss the system architecture and the research we conducted to help us design the architecture. Last, we describe the final system design and our testing and results.

6.2 System Design

To achieve our goal of autonomous flight we needed to design a system that would be able to consume sensor readings from the Android phone to determine the actions needed to travel to the next waypoint. As we stated earlier in Section 3.4.1, we intend to implement non-optimized autonomous flight.

After conducting research on autonomous control theory, discussed in more detail in Section 6.3, we created a high-level algorithm to implement autonomous flight. This algorithm involved three main phases and is shown in Figure 24. The first phase is the *sense* phase which takes in the sensor readings. In the subsequent *plan* phase, sensor readings are compared to their expected values. Based on any differences, the software executes a plan to make progress toward the next waypoint. In the final *act* phase, the plan is executed, and then this algorithm returns to the sense phase.

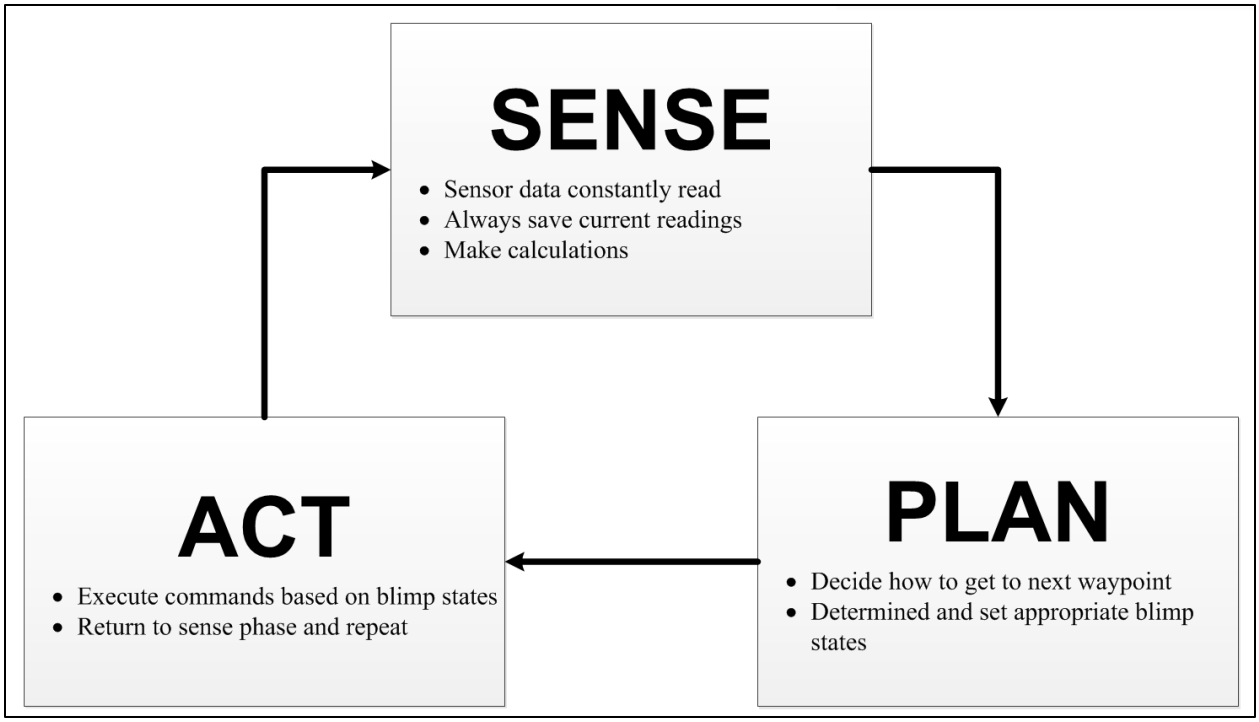


Figure 24: Sense-Plan-Act Model

6.2.1 System Needs

While designing and implementing our system for autonomous flight, we considered what the system needed to accomplish. These needs included logging, keeping track of the blimp's current and previous states, maintaining a queue of GPS coordinates, and interfacing the Android software with the drive motors and servo.

We needed a logging system that would be able to log to files on the microSD card of the Android phone, and that would be able to log from multiple classes and threads simultaneously. We wanted to be able to log everything that happened in the software so that we would be able to perform analysis of what happened during ground and flight tests.

Another one of the needs of our system was to keep track of the blimp's current state and previous state. We needed to save information to be able to make certain calculations and to compare current sensor readings to the required readings to figure out what needed to be done next. State information we needed included all sensor readings, position information, and derived data such as altitude and speed. Part of the state also included GPS waypoints.

Lastly, we needed to be able to interface our Android app that was written in Java with the drive motors and the servo on the gondola. The ideal situation was to have a method where

we could manage the motor and servo signals from within our Android app and have only one code base.

6.2.2 System Requirements

From the needs of our system, we derived our system requirements. Our requirements included the frequency at which we received sensor information, the resolution of our sensors, the amount of error we accounted for in the sensor readings, the reaction time of the drive components to the sensor readings, and the successful implementation of waypoint navigation. These requirements are shown in Table 13.

Autonomous Flight Requirements	
AF-01	The Android device must be able to control the speed of the motors and rotation of the servo
AF-02	The system must update the blimp's current position every second
AF-03	The system must update the blimp's current altitude at least every second
AF-04	The system must update the blimp's current heading at least every second
AF-05	The blimp must be able to autonomously navigate between two GPS waypoints

Table 13: Autonomous Flight Goal Requirements

For the frequency of sensor readings, we needed to collect five readings every second. We needed this frequency of readings because sometimes sensors receive a spike in readings or there is noise. With many readings we could use a moving average to account for possible noise. We wanted to be able to receive these readings quickly because the blimp needed to react to the environment, and a large number of sensor readings in a small amount of time would allow us to write code to make this possible.

From the readings gathered from the sensors, we calculated heading, speed, altitude, and horizontal position. The sensors used for each of these calculations included the gyroscope, accelerometer, GPS, magnetometer, and barometer. We required that each of the calculations be within a specified amount of error. We required the heading to be accurate within +/- 5 meters, the speed within +/- 2 meters/second, the altitude within +/- 1 meter, and the horizontal position within +/- 4 meters. We specified these margins of error because we wanted to have the option of filtering out noise from the sensors to allow precise control of the blimp. Our plan initially was to implement flight using averaged and filtered sensor readings, but we wanted to eventually allow for fine-tuned adjustments of the drive system based on the sensor readings.

As stated earlier, we required a sample size of 5 readings per second from the sensors because we wanted to be able to take averages and filter out noise. Since we knew that the

vibration of the gondola and weather factors had the potential to interfere with sensor readings, we needed to account for some percentage of error with regard to our readings. We also knew that averaging readings could significantly slow down the response time of the software between taking input from the environment and actually using that reading in the control algorithm. For example, at 5 readings per second, the initialization of an average of 20 readings would take 4 seconds to complete. After the initialization, averages could be taken every 10 readings which is every 2 seconds. This would essentially cause the control algorithm to lag 2 seconds behind what was actually happening.

Another requirement of our system was that the blimp should be able to react to its environment in under 2 seconds. This means that the time delay between the sense and act phases of flight, as introduced in Section 6.2, should be no more than 2 seconds. This was a requirement because the wind could change, and we needed the blimp to be able to react to changing conditions as quickly as possible. There are many different ways that this could be implemented, but we needed our system to be able to perform well with our Android phone and the interface with the drive components. This will be described in detail in Section 6.5.

The last requirement we had for the autonomous flight system was that we needed to successfully implement waypoint navigation in our software. Waypoint navigation is derived from autonomous control theory and states that given waypoints A, B, C, if some vessel can travel from points A to B, then it can also travel from B to C [13]. Our implementation of waypoint navigation will be discussed in Section 6.5.

6.3 System Architecture and Trade Studies

We derived our final system design from researching autonomous control theory and the best practices of writing Android apps in Java. We also performed research on different logging and log analysis systems.

6.3.1 Autonomous Control Loop

As stated in Section 6.2, based on the research we performed on autonomous control theory, we were able to adopt the sense-plan-act algorithm to implement our autonomous control. In this section we will discuss each phase of the algorithm in detail.

During the sense phase, sensor readings are taken in from the environment and processed. Processing includes filtering out noise, managing running averages, and derivations such as speed and altitude. The sensor readings we took directly while only filtering noise were the GPS,

accelerometer, and gyroscope readings. We managed running averages of the barometer and heading readings, and derived speed from GPS readings and altitude from barometer readings. An in-depth discussion of how we performed our derivations is explained in Section 6.6.1.

The next phase is the plan phase. Based on the next waypoint the blimp needs to travel to, the software keeps track of what the heading, altitude, and speed need to be in order to get to the waypoint. During the plan phase, current readings and derivations are compared to what they need to be to get to the next waypoint. Depending on the outcome of the comparisons, a command is generated that is sent to a state manager. The command encapsulates information that indicates what the state of the blimp needs to be. Flight states will be described in Section 6.3.2.

The final phase is the act phase. The act phase executes the command that was generated in the previous phase. In our system, the command is translated into a set of PWM signals that are sent out to the drive system. This phase also updates the current flight state of the blimp to what is currently being executed. After this phase, the loop returns to the sense phase and the entire loop repeats.

The loop exits when either the autonomous flight is manually aborted, or if the sequence of waypoints has been traversed and the flight has been completed.

6.3.2 Flight States

From our sense-plan-act design, we defined several different orthogonal states that the blimp could be in at any given time. States are divided into state types and state actions. The five state types of the blimp are:

- INACTIVE – the blimp is not yet active (default initialization)
- ALTITUDE – the blimp is seeking to stabilize the appropriate altitude
- HEADING – the blimp is seeking to stabilize the appropriate heading
- SPEED – the blimp is seeking to stabilize its horizontal speed
- STOP – the blimp has stopped

Within each state type, the blimp could perform a number of state actions. Combinations of state actions are not allowed. The different state actions are defined as follows:

- HEADING
    - TURN_LEFT_FAST
    - TURN_LEFT_SLOW
    - TURN_RIGHT_FAST
    - TURN_RIGHT_SLOW

- ALTITUDE
  - INCREASE
  - DECREASE
- SPEED
  - CRUISE
  - NEUTRAL

The INACTIVE and STOP state types indicate that the autonomous flight has not yet begun or that autonomous flight has been stopped for some reason, respectively. Within the context of the sense-plan-act algorithm, the plan phase generates commands that encapsulate the state type and action that should be executed, and the act phase executes that command. Figure 25 shows a state chart for the blimp state types and actions.

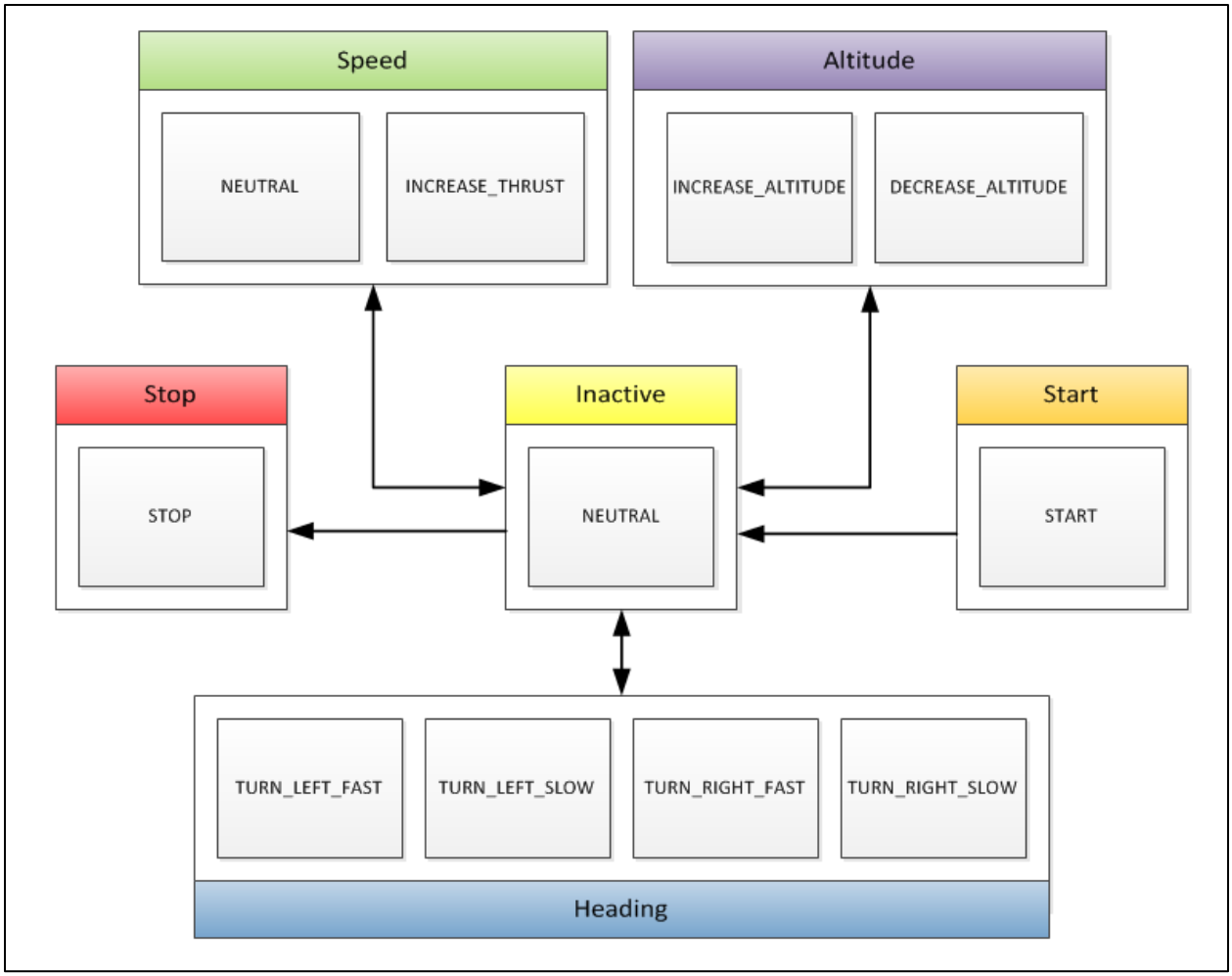


Figure 25: Blimp State Chart

When the software is initialized, the state goes from the Start block to the Inactive block. Start is not actually a state type in the software. The Start block only indicates the initialization of the software. From there, the altitude is checked until the blimp reaches the required altitude.

Then, the autonomous control loop is entered, and the heading and speed are checked as well as the altitude. Once the flight has been completed, the system reaches the Stop state. Table 14 shows the mapping of hardware PWM signals for each of the state types and actions in the software.

Software State Type & Action		Drive System Hardware PWM Signals			
States	**Functions**	**Servo**	**Tail Motor**	**Left Motor**	**Right Motor**
Start	NULL	NULL	NULL	NULL	NULL
Inactive	NEUTRAL	1.500ms	1.500ms	1.300ms	1.300ms
Stop	NEUTRAL	1.500ms	1.500ms	1.300ms	1.300ms
Speed	NEUTRAL	1.500ms	1.500ms	1.300ms	1.300ms
	INCREASE_THRUST	1.500ms	1.500ms	1.475ms	1.475ms
Altitude	INCREASE_ALTITUDE	1.250ms	1.500ms	1.475ms	1.475ms
	DECREASE_ALTITUDE	1.750ms	1.500ms	1.475ms	1.475ms
Heading	TURN_LEFT_FAST	1.500ms	2.000ms	1.300ms	1.475ms
	TURN_LEFT_SLOW	1.500ms	2.000ms	1.350ms	1.475ms
	TURN_RIGHT_FAST	1.500ms	1.000ms	1.475ms	1.3000ms
	TURN_RIGHT_SLOW	1.500ms	1.000ms	1.475ms	1.350ms

Table 14: PWM Signals for Blimp States

Again, the Start state only indicates the initialization of the software. The minimum PWM signal for any drive system component is 1.000ms, and the maximum is 2.000ms. However, for the left and right motor, 1.300ms indicates the motor is off and 2.000ms indicates maximum speed. For the tail motor 1.500ms indicates off and either 1.000ms or 2.000ms indicates maximum speed in either direction. Lastly, the servo is neutral at 1.500ms and is rotated the maximum amount in either direction at 1.000ms or 2.000ms.

While developing the flight states, we considered the experience of the 2011 AY team. The team determined that attempts to change altitude while simultaneously turning would break the wooden dowel the motors were mounted on because of the high gyroscopic forces 134[1]. We decided that altitude is more important than heading or speed, so whenever the altitude has to be changed, the autonomous control algorithm overrides any actions regarding the heading and speed. After the altitude, we decided heading is more important than the speed, so heading

commands override speed commands, but not altitude commands. Lastly, speed commands only execute if neither altitude nor heading commands are already being executed.

6.3.3 Flight States

The logging system we used contained two main components. The first component is implemented inside the Android app and uses Log4j[4] to write to files saved onto the microSD card of the phone. We wanted to be able to analyze the logs after a test to determine what happened during the test. Everything that happens inside the app is logged to a file in plain-text. This includes sensor readings, planned commands, executed commands, and position and speed information. The next step is to retrieve the logs from the microSD card in phone and analyze them. To analyze the log files, several options were considered. Since we used Log4j, an Apache project, we considered using Chainsaw, another Apache project. Chainsaw is a GUI[5] tool that can be used to parse and analyze Log4j logs. We decided that Chainsaw would not work for our application because it only was able to use Log4j files in XML[6] format, not plain-text. The next option we considered was Sawmill. Sawmill is a professional application for parsing and analyzing Log4j files. We were able to quickly decide that we could not fit this into our budget. The final option was to write our own log analysis tools. Using Python 3, we successfully wrote scripts to parse our log files and analyze statistics from them.

6.4 Final System Design

For our final system design we wrote an Android app in Java 1.6, since the Android system did not support Java 1.7 at the time of this writing. The Android app ran on the Samsung Galaxy S3 in conjunction with an Android IOIO board as described in Chapter 5. Our Android app implemented the sense-plan-act algorithm described earlier and was able to autonomously control the blimp's drive system using sensor readings from the environment.

When writing our autonomous control software, we decided that we would encapsulate different modules in packages, as is common when using software best practices. Our package structure is shown in Figure 26. The `util` package contains utility classes for operations such as logging, formatting timestamps, and keeping moving averages. The `usb` package contains the `ioio` package which deals with sending the PWM signal information to the IOIO board. Lastly,

[4] Log4j is a commonly used logging system for Java applications.
[5] GUI – Graphical User Interface
[6] XML – Extensible Markup Language

the `autonomous` package contains the `ui`, `sensors`, `model`, and `monitors` packages. The `ui` package contains all of the user interface classes for the Android app. The `sensors` package contains classes that listen for sensor readings and update the UI. The `model` package contains classes that maintain the state information of the blimp. Lastly, the `monitors` package contains classes that check the current heading, altitude, and speed readings with their required values. The `monitors` package also generates commands based on the how the readings compare with the required values. Essentially, the sense phase is contained within the `sensors` and `model` packages, the plan phase is contained in the `monitors` package, and the act phase is contained within the `ioio` package.

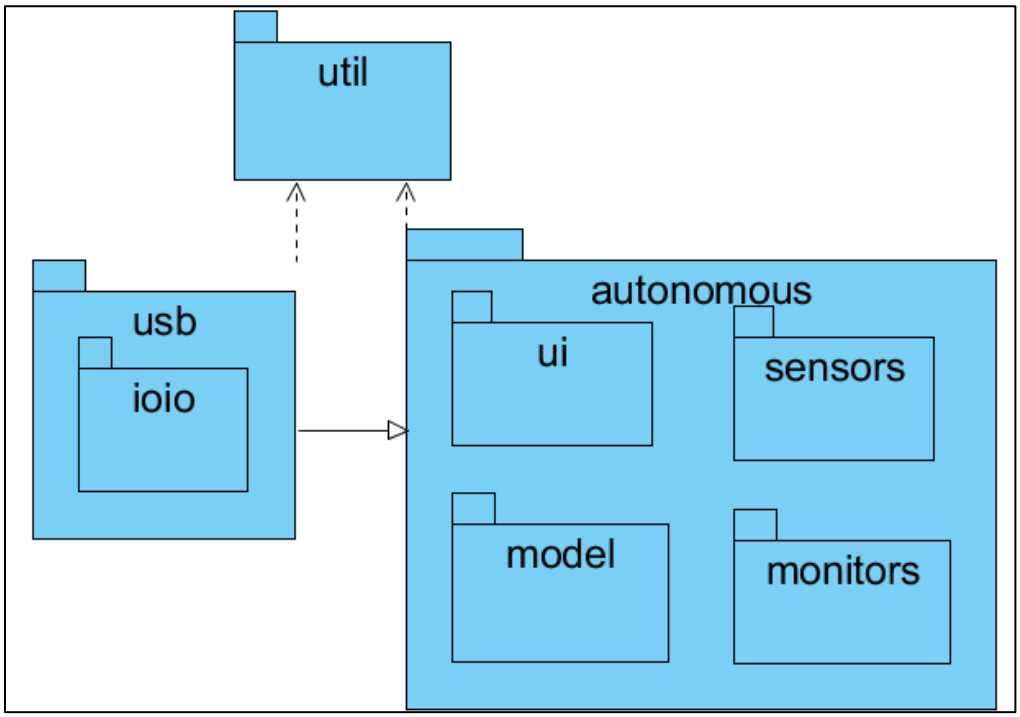


Figure 26: Package Structure of Autonomous Code

6.5 Detailed System Design

In this section we describe each package as well as the classes in each package. We will also discuss the purpose of each class and any important details or design choices that went into each class.

The `util` package contains classes for logging, averaging sensor readings, and various enumerations. All of the classes contained in this package are as follows:

- `Class BarometerAverage`

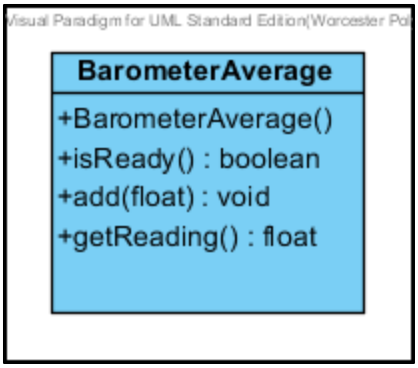


Figure 27: BarometerAverage Class UML

- This class encapsulates the collection of barometric pressure readings. It can be used to retrieve the average barometric reading which is averaged from the most recent readings. The 20 most recent readings are averaged, and at 5 readings per second, this means that there is a 4 second delay on calculating the averages.

- `Class Command`

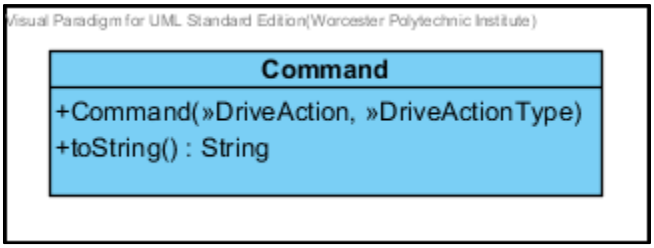


Figure 28: Command Class UML

- This class encapsulates commands that are sent from the `Monitor` classes to the `StateManager` and executed by `PWMGenerator`. The `StateManager` and `PWMGenerator` are described below.

- `Class ConfigureLogging`

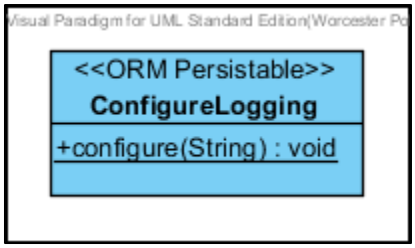


Figure 29: ConfigureLogging Class UML

- This class is used to configure logging using Log4j.

- `Class Coordinate`

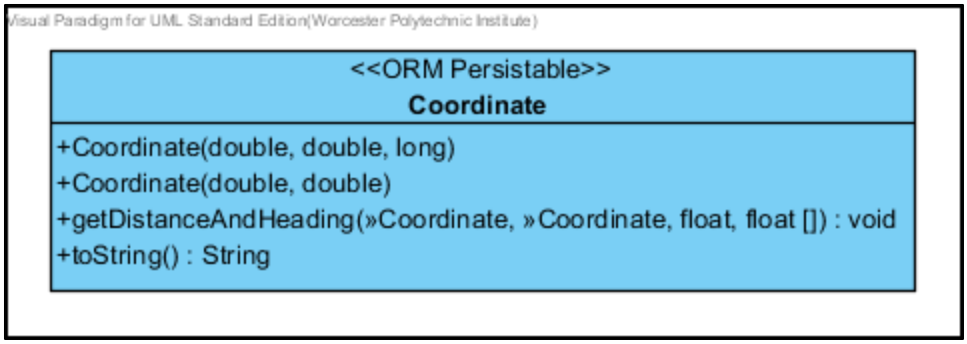


Figure 30: Coordinate Class UML

- This class encapsulates latitude/longitude GPS coordinates with the timestamp of when the coordinate was recorded.
- The `getDistanceAndHeading()` method takes in a `to` and a `from` coordinate, the total distance from the start to the next waypoint, and a float array of size 3. This method makes a call to the Android Location class which calculates the distance between the `to` and `from` coordinates. The initial and final bearings to the `to` coordinate are also calculated during the distance calculation. The distance, initial, and final bearings are put into the float array that is passed into this method.

- `Class Timestamp`

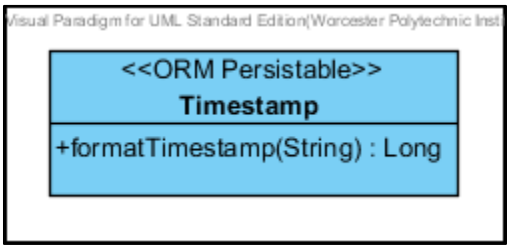


Figure 31: Timestamp Class UML

  - This class is used to format timestamps.
- `Enum DriveAction`
  - This enumeration defines drive actions. For example: `INCREASE_THRUST`, `TURN_LEFT`, etc.
- `Enum DriveActionType`
  - This enumeration defines the types of drive actions. For example: `HEADING`, `ALTITUDE`, `SPEED`.

- `Enum DriveComponent`
    - This enumeration defines the drive components. For example: `SERVO, LEFT_MOTOR, RIGHT_MOTOR, TAIL_MOTOR.`
- `Enum Speed`
    - This enumeration defines the speeds of PWM signals. For example: `OFF, LOW, MED, HIGH.`

The `usb.ioio` package contains the following class:

- `Class PWMGenerator`

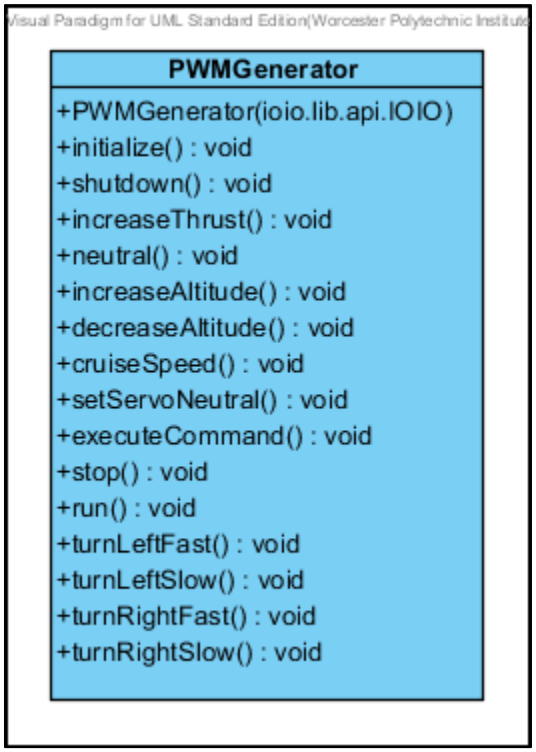


Figure 32: PWMGenerator Class UML

    - This class represents the hardware abstraction layer and runs in its own thread. Commands are taken from `StateManager` and executed to tell the IOIO board which signals to send to the motors and servo.

The `autonomous` package contains the following classes:

- `Class Navigation`

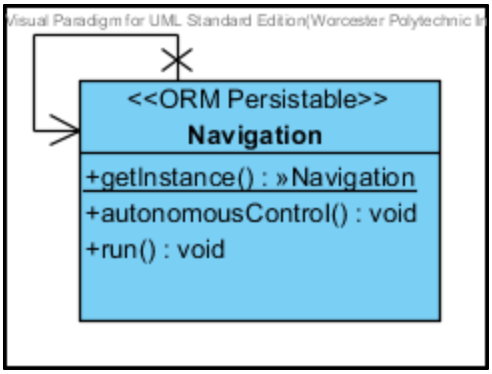


Figure 33: Navigation Class UML

- This class implements the sense-plan-act autonomous control loop and runs in its own thread. In each iteration of the loop, calls are made to `AltitudeMonitor`, `HeadingMonitor`, and `SpeedMonitor` to check the current sensor readings against the required readings. Each of these calls return a command based on what needs to be done. Then, this class adds a command to the currently executing command in `StateManager` based on the priorities of altitude, heading, and speed.

- `Class StateManager`

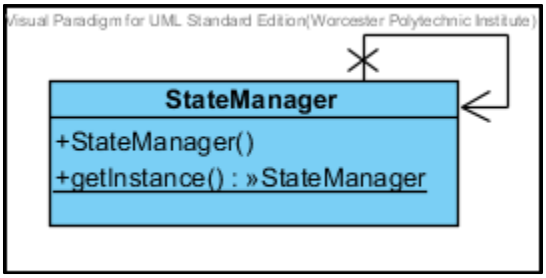


Figure 34: StateManager Class UML

- This class maintains the command that the `PWMGenerator` should execute. This class also maintains the previously executed command.

The `autonomous.model` package contains classes that maintain the state of the blimp.

- `Class AltitudeState`

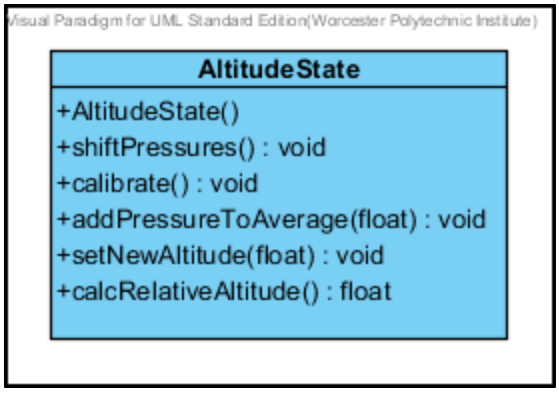


Figure 35: AltitudeState Class UML

- This class encapsulates the barometer readings and altitude data.
- The distinction between `shiftPressures()` and `addPressureToAverage()` is that `shiftPressures()` sets the value of the current pressure reading to the previous pressure reading, while `addPressureToAverage()` adds a new pressure reading to the moving average of pressure readings, which is contained in the `Barometer` class.

- `Class BlimpState`

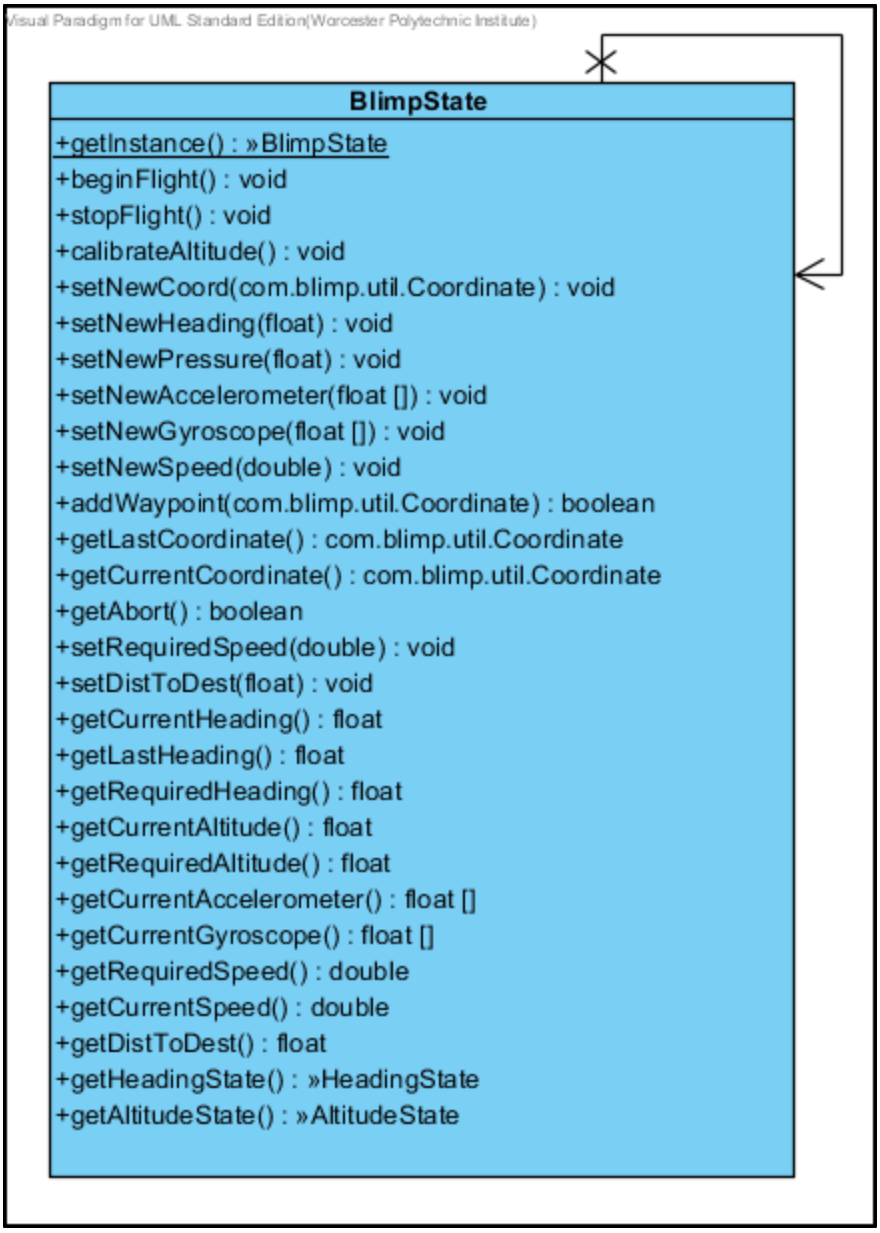


Figure 36: BlimpState Class UML

- This class represents the current state of this blimp and maintains references to all the other classes in this package.

- `Class HeadingState`

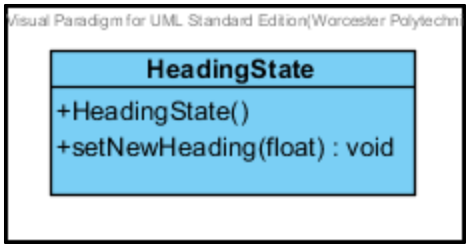


Figure 37: HeadingState Class UML

- This class encapsulates the heading information which includes current, required, and previous headings.

- `Class IMUState`

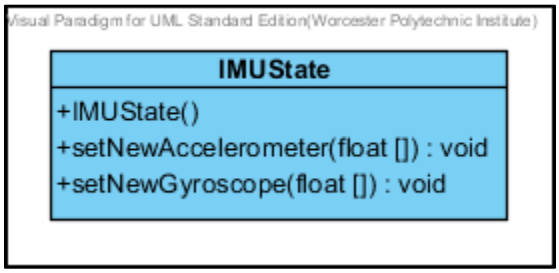


Figure 38: IMUState Class UML

- This class encapsulates the gyroscope and accelerometer data.

- `Class NavigationalState`

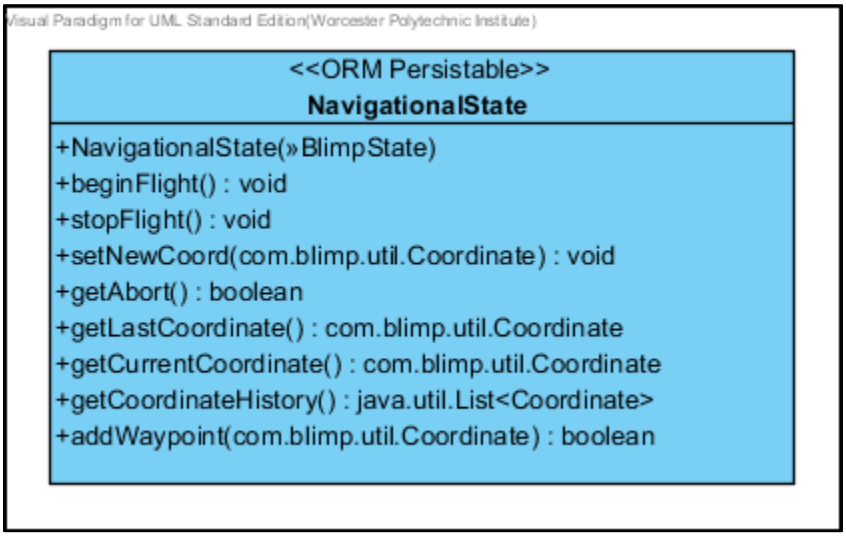


Figure 39: NavigationalState Class UML

- This class maintains the current state of the flight, the GPS coordinate of the blimp, the history of coordinates for this flight, the distance to the destination waypoint, and references to the `Waypoints` and `Progress` classes.

- `Class Progress`

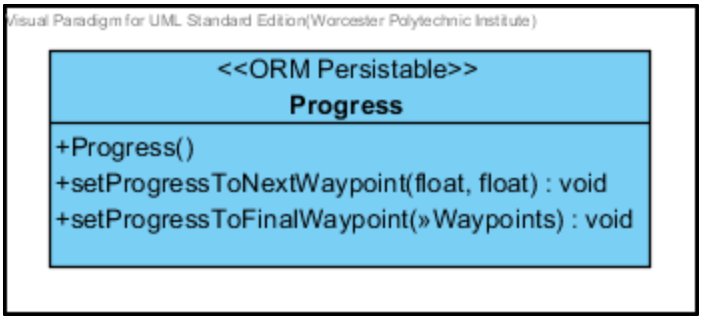


Figure 40: Progress Class UML

- This class encapsulates the blimp's progress percentage to the next waypoint, and the progress percentage to the final waypoint.

- `Class SpeedState`

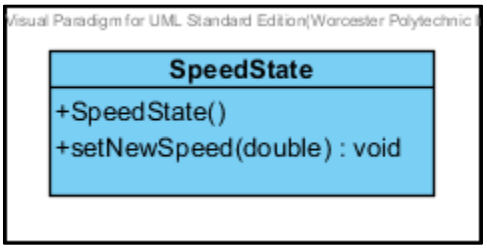


Figure 41: SpeedState Class UML

- This class encapsulates the speed information of the blimp including current, required, and previous speeds.

- `Class FlightPlan`

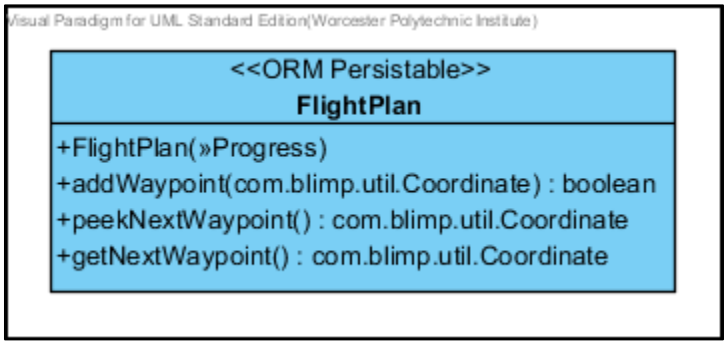


Figure 42: FlightPlan Class UML

- This class maintains a queue of coordinates for this flight, the total number of waypoints in the queue, and a reference to the `Progress` class.

The `autonomous.monitors` package contains classes that check the current readings against the required sensor readings during the plan phase of the control algorithm.

- `Class AltitudeMonitor`

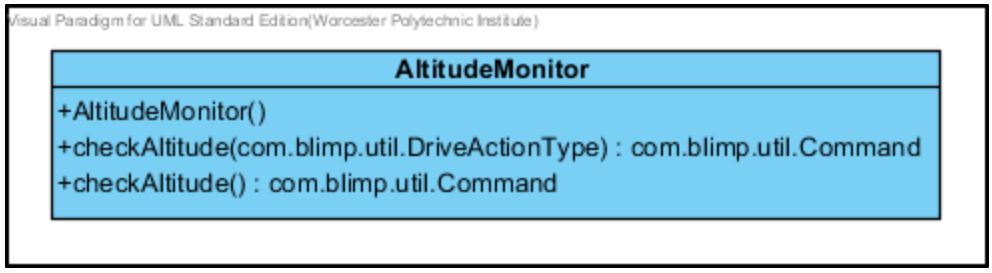


Figure 43: AltitudeMonitor Class UML

- This class checks the current altitude against the required altitude and returns a command. If the current altitude is less than the required altitude minus the error threshold, an `INCREASE_ALTITUDE` command is returned. If the current altitude is greater than the required altitude plus the error threshold, a `DECREASE_ALTITUDE` command is returned. Otherwise, the altitude is correct within the error threshold, and a `DO_NOTHING` command is returned. Figure 44 shows the relationship of the gondola and blimp to the required altitude. The required altitude has +/- 1 meter of error, and the maximum possible altitude is 30 meters because of FAA regulations.

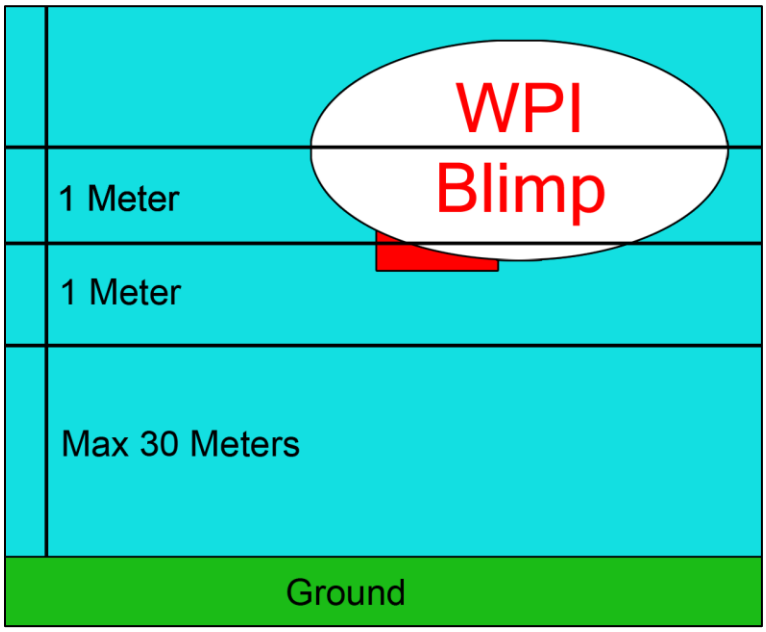


Figure 44: Blimp Altitude Adjustments

- `Class HeadingMonitor`

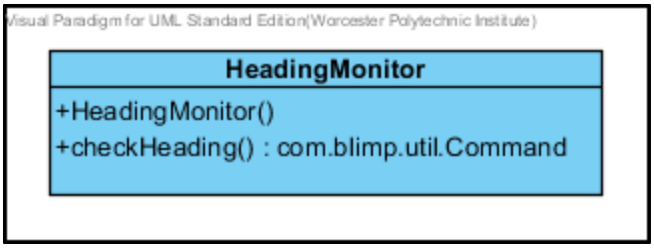


Figure 45: HeadingMonitor Class UML

- This class checks the current heading against the required heading and returns a command. To check the heading, the required and current headings are first shifted so that the required heading is 180 degrees and the current heading gets moved by the difference between the actual required heading and the shifted required heading. Then, if the shifted current heading is within the minimum error threshold of the required heading, a `DO_NOTHING` command is returned. If the shifted current heading is outside of the minimum error threshold range and inside of the maximum error threshold range, either a `TURN_LEFT_SLOW` or `TURN_RIGHT_SLOW` command is returned, depending on whether the shifted current heading is greater or less than the shifted required heading. If the shifted current heading is outside of the maximum error threshold of the required heading, either a `TURN_LEFT_FAST` or `TURN_RIGHT_FAST` command is returned, depending on whether the shifted current heading is greater than or less than the shifted required heading. Figure 46 shows the angle thresholds where each state action is executed depending on the orientation of the blimp.

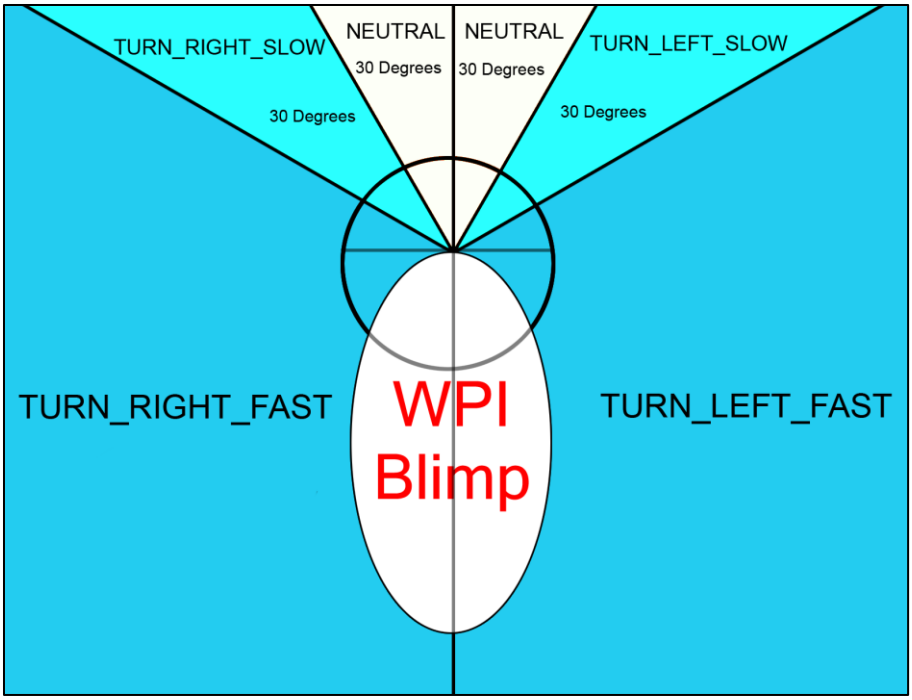


Figure 46: Blimp Heading Adjustments

- `Class SpeedMonitor`

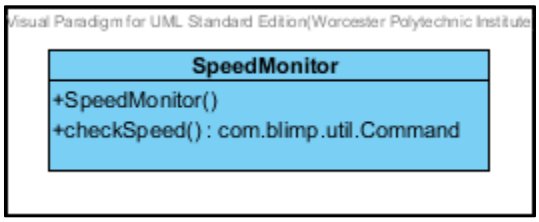


Figure 47: SpeedMonitor Class UML

  - This class checks the current speed against the required speed and returns a command. If the current speed is less than the required speed minus the error threshold, an `INCREASE_THRUST` command is returned. If the current speed is greater than the required speed plus the error threshold, a `NEUTRAL` command is returned. Otherwise, the speed is correct and a `DO_NOTHING` command is returned.

The `autonomous.sensors` package contains listeners for all of the Samsung Galaxy S3's sensors.

- `Class GPSListener`

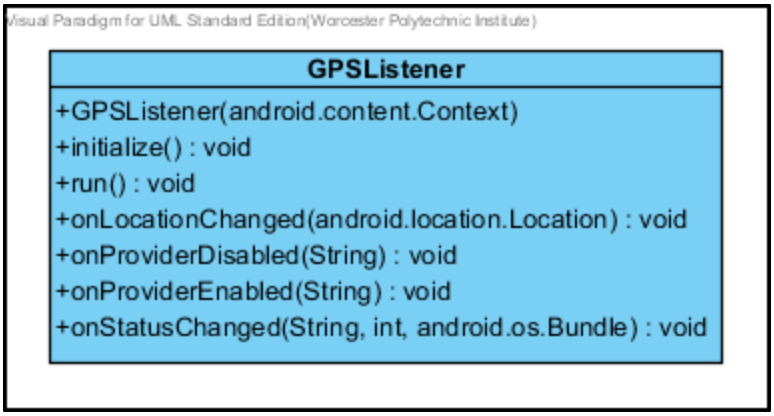


Figure 48: GPSListener Class UML

- This class listens for updates from the GPS and updates `BlimpState` whenever a new reading is received.

- `Class SensorListener`

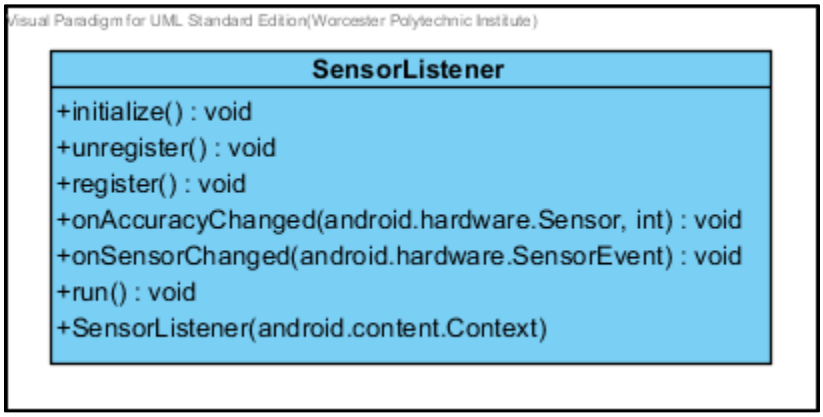


Figure 49: SensorListener Class UML

- This class listens for updates from the accelerometer, gyroscope, magnetometer, and barometer and updates `BlimpState` whenever new readings are received.

The `autonomous.ui` package contains the classes necessary for the user interface on the S3 which allows the user to enter waypoints, start flight, and view the state of the blimp.

- `Class InputActivity`

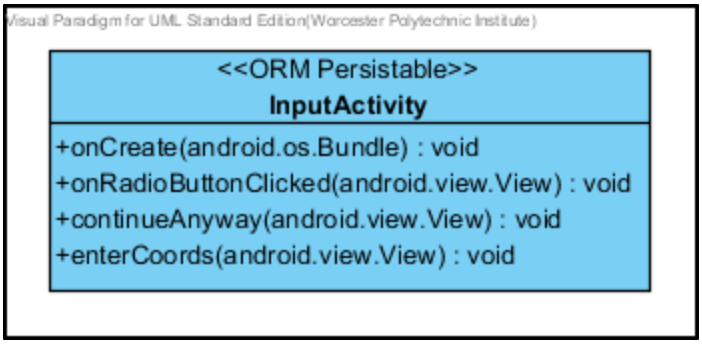


Figure 50: InputActivity Class UML

- This class is used to display the screen that allows the user to select pre-programmed GPS waypoints by tapping the corresponding radio button.

- `Class SensorTab`

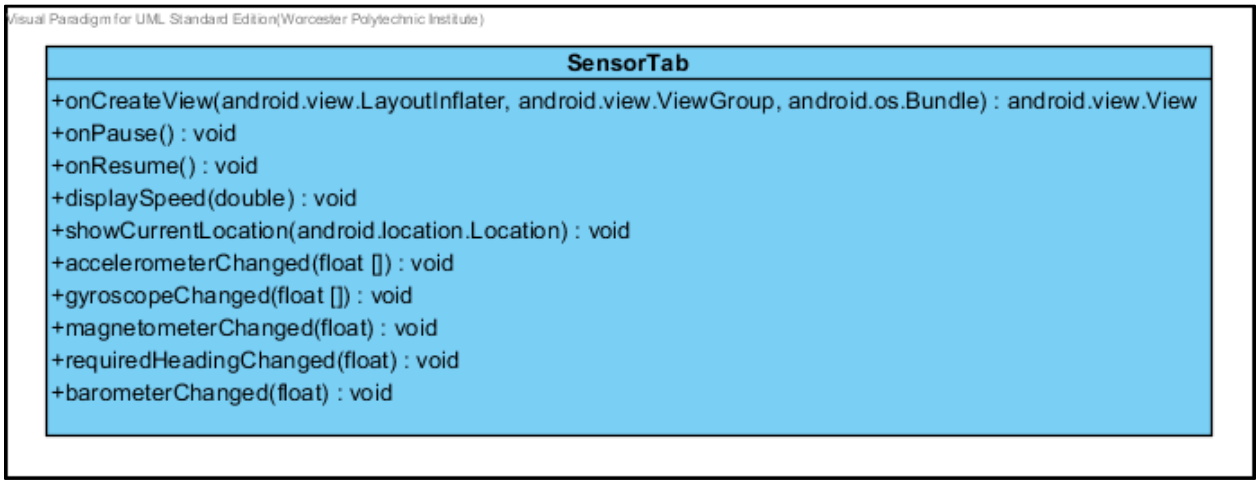


Figure 51: SensorTab Class UML

- This class is used to display the tab that contains all of the phone's sensor readings as well as the calculated readings such as speed and altitude.

- `Class StartStopFlightTab`

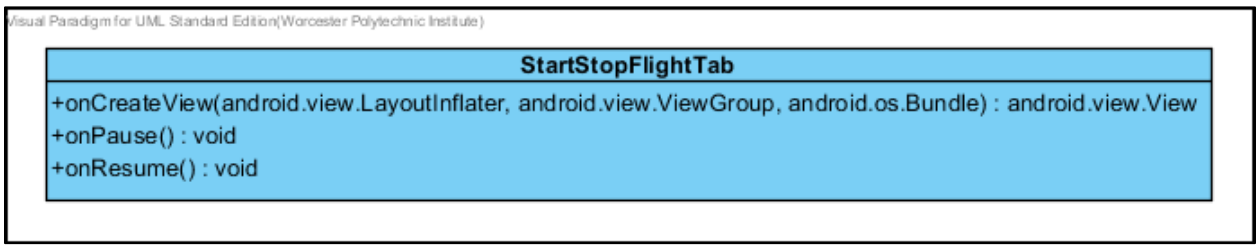


Figure 52: StartStopFlightTab Class UML

- This class is used to display the buttons for starting and stopping the autonomous control algorithm.

- `Class StateTab`

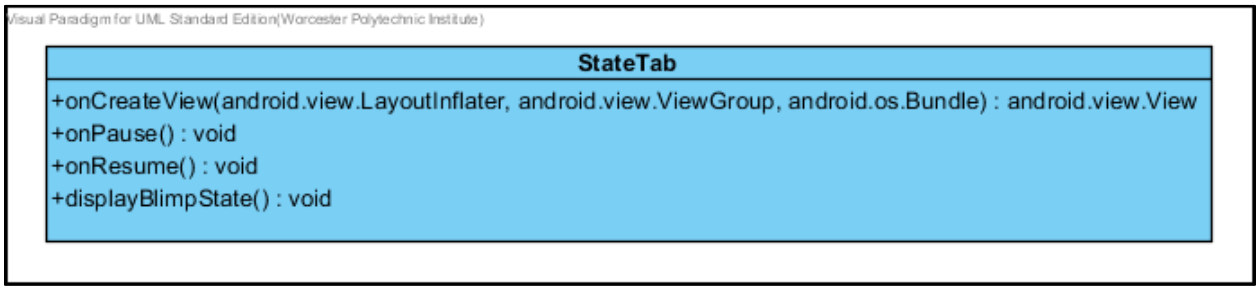


Figure 53: StateTab Class UML

  - This class is used to display the current flight state of the blimp. The flight state includes the `DriveActionType` and the `DriveAction`.

- `Class TabLayoutActivity`

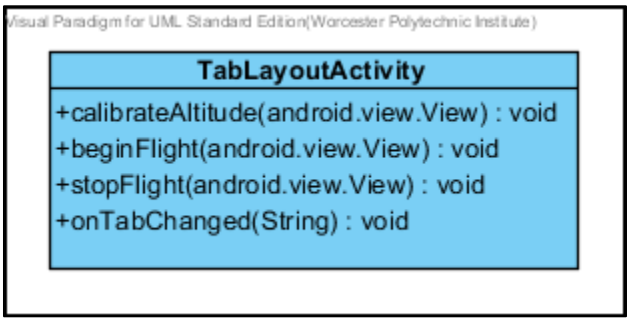


Figure 54: TabLayoutActivity Class UML

  - This class initializes and manages the other tab classes.

6.5.1 Android App

Figure 55 shows the first UI screen that is displayed when the Autonomous Blimp Android App is started. This screen contains a set of radio buttons that allow the user to select pre-programmed GPS destination coordinates. There are also input fields for latitude and longitude values which allow the user to enter a custom GPS destination coordinate. Once the desired coordinate(s) have been entered, the user may press the "Continue" button to enter the coordinate(s) as a destination. The user may also choose to press the "Skip" button to avoid entering any destination coordinates.

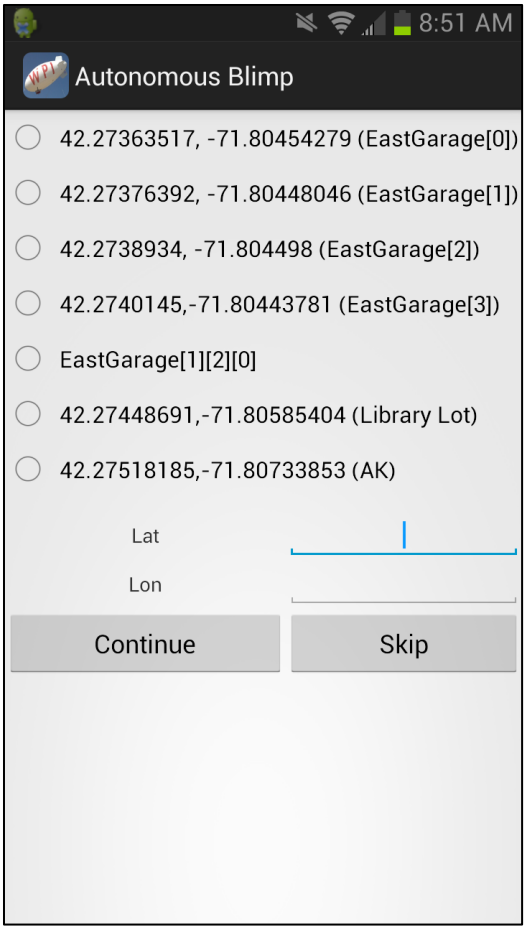


Figure 55: UI Screen Used to Enter GPS Destination Coordinates

Figure 56 shows the next UI screen that is displayed to the user after either "Continue" or "Skip" are pressed on the initial screen. This screen displays a tabbed layout that remains active throughout the remainder of the lifetime of the app. The tab displayed in Figure 56 shows all of the current sensor readings and derived information from the sensor readings. These include altitude, GPS location, current and required heading, air pressure, speed, accelerometer, and gyroscope readings. There is also a "Calibrate Altitude" button that sets the current altitude to 0 when tapped. This is necessary because the altitude reading is derived from the air pressure reading and is relative to the air pressure. Altitude calculations and tests are discussed in detail in Section 6.6.1.

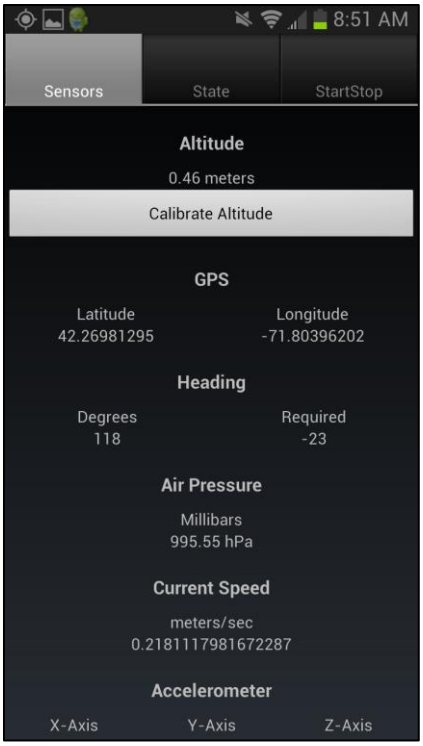


Figure 56: UI Screen Displaying the Sensors Tab

Figure 57 shows the tab that displays the current state information. The state information includes the currently executing state type and state action as described in Section 6.3.2.

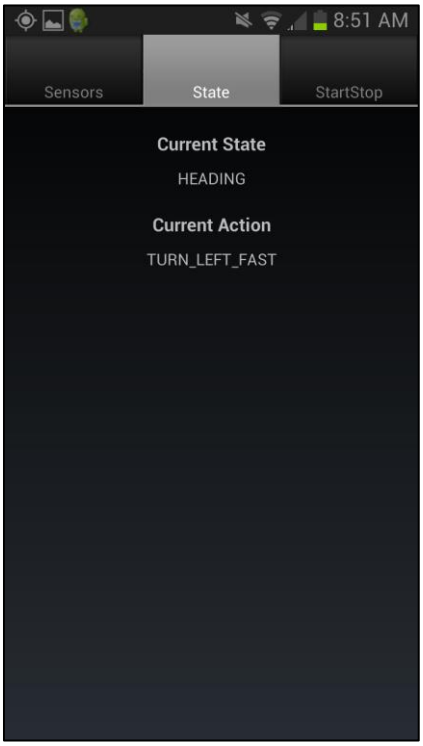


Figure 57: UI Screen Displaying the State Tab

Figure 58 shows the tab that displays buttons that can be used to begin or end the autonomous control algorithm. Pressing the "Begin Flight" button will begin the autonomous control algorithm, and pressing the "Stop Flight" will exit the autonomous control algorithm.

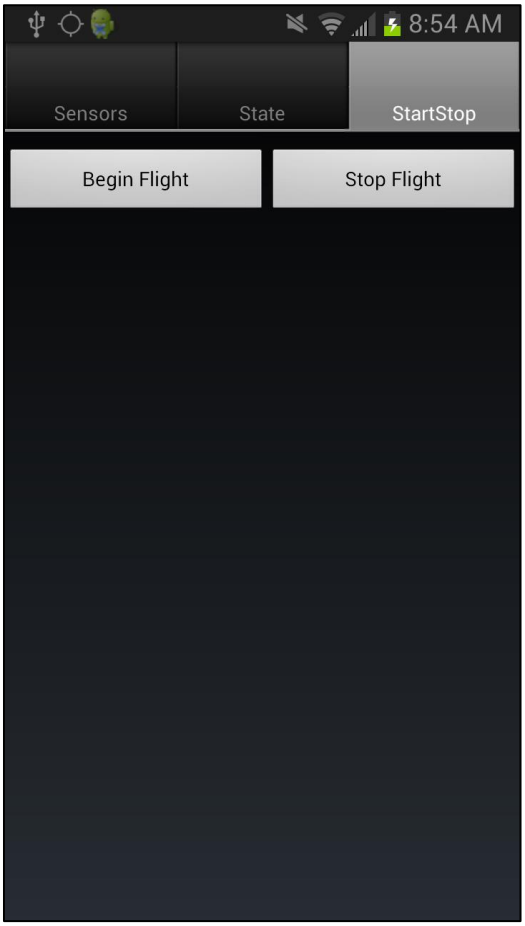


Figure 58: UI Screen Displaying the StartStop Tab

6.6 Testing and Results

In this section we detail the testing of the software and the results that were returned. The tests include sensor testing, ground testing, and flight testing.

6.6.1 Sensor Tests

To accurately and reliably determine how to instruct the blimp to navigate to its next waypoint, we needed to have new sensor data arrive at least every second. On testing the Galaxy S3's sensors, we discovered that all the sensors with the exception of the GPS updated 5 times per second. The GPS updated less frequently at about once every two seconds.

Next, we tested the accuracy of the sensors. To accomplish this, we conducted tests outdoors to determine the accuracies of the GPS, magnetometer for the heading, and the

barometer for the altitude. For the GPS test, our procedure was as follows, with a visual representation in Figure 59:

1. Choose a start and end point in the WPI library parking lot
2. At the start point, begin logging GPS and heading data on the phone
3. From the start, move five parking spaces, stop, and take readings
4. Keep moving five spaces at a time until the end point is reached
5. Recover and examine log file from the phone
6. Plot all recorded GPS coordinates onto a map

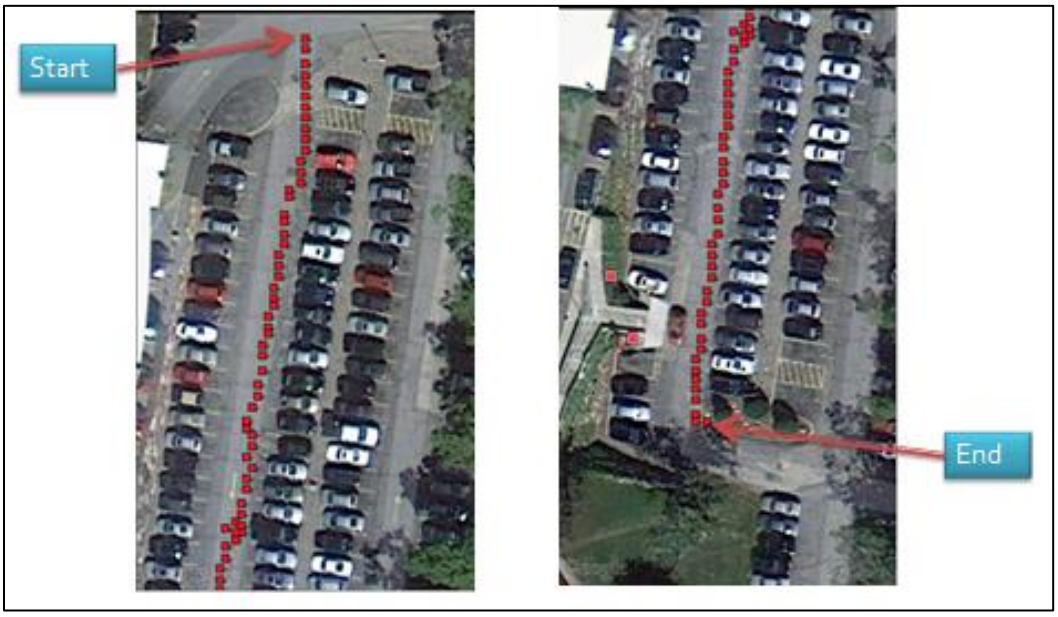


Figure 59: GPS Test Recorded Coordinates

From this test, we were able to conclude that the GPS was accurate within four meters, and the heading was accurate within five degrees. We also decided that by using the distance between two GPS coordinates and the timestamps at which each were recorded, we could calculate the speed. To calculate speed we simply divide the distance by the time. However, finding the distance between GPS coordinates is not trivial. To get this value, calls are made in the Android app to the Android software API. The Android API uses the Inverse Formula to calculate the distance between two GPS coordinates; the Inverse Formula is described in Appendix B.

Next we tested the barometer on the S3. The purpose of this test was to figure out how a given barometer reading can be used to calculate relative altitude. The procedure for this test was as follows:

1. Start at the top of the WPI library stairs at the street level
2. Log barometer reading

3. Walk down to the next landing, counting how many steps were taken
4. Log the barometer reading
5. Repeat walking down to the next landing, counting the steps and logging data until reaching the bottom at the parking lot
6. Measure the height of the first step and one other step within each section of steps
7. Compare barometer readings to the vertical distance travelled

For this test, the total number of steps traversed was 106, the total vertical distance travelled was 14.71 meters, and the total difference in air pressure from the top to the bottom step was 1.80 millibars. We averaged the calculated pressure per meter for each section of steps, which gave us the constant of 0.1369 millibars/meter. We could use this constant to calculate the relative altitude according to the following method:

1. Set altitude to 0 at starting height to calibrate
2. Divide the difference of the current and last recorded air pressure (barometer reading) by the constant 0.136939383
3. Multiple the result by -1 since a pressure increase means altitude decrease, and pressure decrease means altitude increase
4. The result is the relative altitude

It is important to note that this calculation is only valid when relatively close to the ground. This was adequate for our purposes since the blimp was not expected to fly above 30 meters. Through further testing, we determined that our calculated relative altitude was accurate within 1 meter.

Lastly, because the heading and altitude readings varied approximately +/- 30 degrees and +/- 1 meter, respectively, they had the potential to cause problems during a flight. Subsequently, we decided to average our readings. For the heading, we averaged the previous 10 readings and used a circular buffer to maintain a moving average. For the altitude, we averaged the previous 20 readings and found the difference between the most recent 10 readings and the earlier 10 readings to determine the change in altitude.

6.6.2 Ground Tests

In this section we describe our ground testing procedures and results regarding the autonomous control portion of the project. We also explain how we analyzed the results of each test and made changes to rectify any issues to improve the results of the next test.

6.6.2.1 First Ground Test

Our initial ground test was performed in front of the Atwater Kent building at WPI. The test consisted of a single destination GPS waypoint. To perform the test we carried the gondola and watched the Galaxy S3 to see which commands were being issued. We acted as the drive system and acted out each of the commands that the phone was attempting to issue. The purpose of this test was to test how the Galaxy S3 would react while mounted inside the gondola. While carrying the gondola, the S3 did not remain perfectly flat and steady, which simulated flight conditions. Figure 60 shows the start and end positions along with the path that was recorded by the GPS throughout the test.

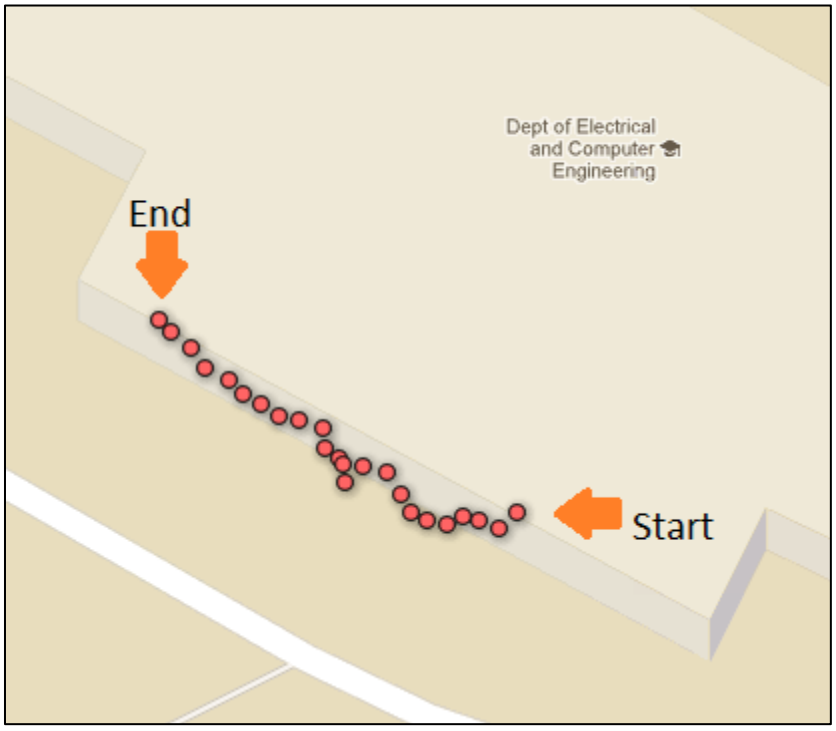


Figure 60: Recorded Path of First Ground Test

During the test, the autonomous system logged all sensor readings and commands that were executed in addition to the GPS coordinates that were recorded. After performing post-test analysis and examining the log files that were recorded on the Galaxy S3, we noticed that approximately 88% of the time the software was issuing commands to tell the drive system to

turn either left or right. We also noticed that the recorded heading readings varied within a range of approximately 300 degrees, even though the actual heading of the gondola only varied a maximum of 180 degrees during the test. This degree of error was due to inaccurate readings coming from the magnetometer. Other than the heading error, the GPS accurately recorded our actual path from start to end, and the software was always trying to correct the drive system in the correct direction based on the heading.

We performed research about the accuracies of magnetometers on smartphones and discovered that the magnetometer readings can be combined with the accelerometer and gyroscope readings to give a much steadier heading reading [14] [15]. This process is commonly called "sensor fusion". The code to implement "sensor fusion" was added to our code base, and upon initial testing we noticed that the heading readings were steadier and more responsive.

6.6.2.2 Second Ground Test

Our second ground test was performed on the top-most level of the East Hall Parking Garage at WPI. The purpose of this test was to test the manual control of the gondola by using the RC controller with the gondola mounted onto a wheeled cart as shown in Figure 61, and to test the autonomous control with the gondola on the cart. For the autonomous portion of this test, we began by starting the motors with the manual RC control, and then switched over to autonomous mode once the motors were spinning. Figure 62 shows the start and end positions as well as the path that was recorded and logged by the GPS on the S3.

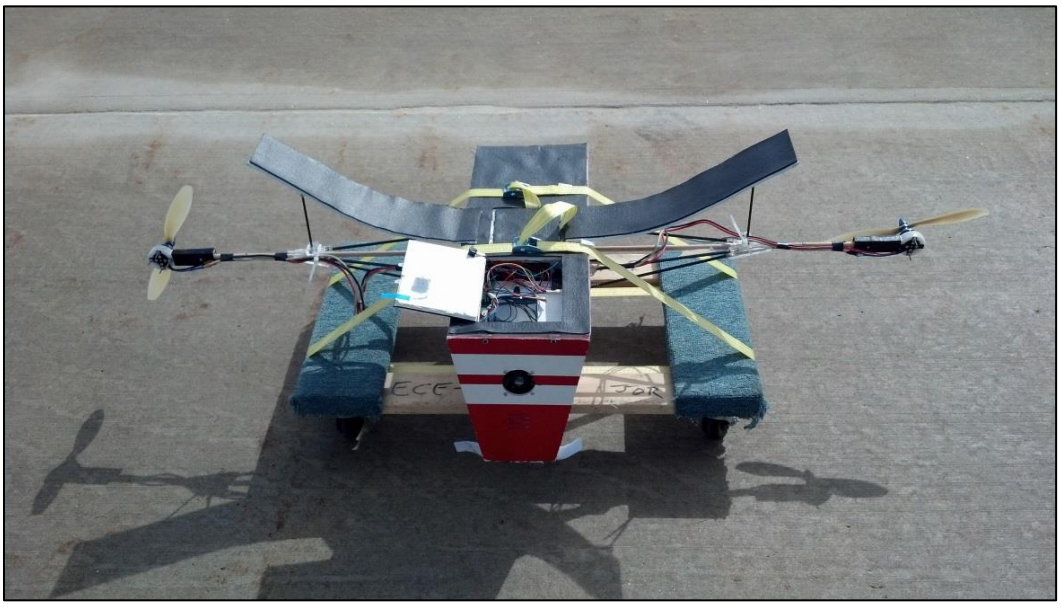


Figure 61: Gondola on Dolly for Ground Test

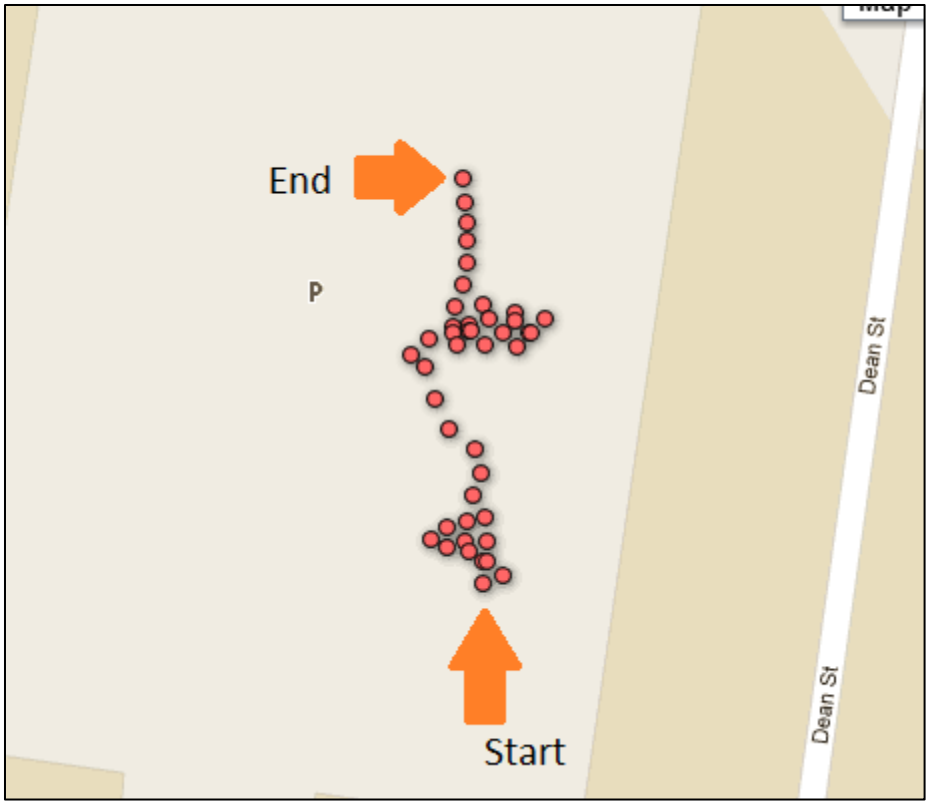


Figure 62: Recorded Path of Second Ground Test

From the plotted data, we observed that there appeared to be two areas where there were many GPS positions clustered together. Namely, at the beginning of the test and about two-thirds of the way through the test there are noticeable clusters. These can be explained by the fact that during the test, the combination of wind and uneven terrain caused the gondola on the cart to veer off course. Once the gondola went off course, the software running on the S3 attempted to correct the heading by changing the motor speeds to turn the gondola. As this was our first test with the autonomous code controlling the motors with propellers, we had no prior experience with the motor speeds and how the gondola would react to different speeds. For turning, we had specified that one motor would be completely off and the other motor would be turned on to medium speed. During the test, these motor speeds caused the gondola to turn faster than the software could react which caused the gondola to spin in a circle. One of the group members had to assist the gondola to get back on course. Once the software recognized that the heading was correct, it changed the motors to run at equal speeds. The gondola was then able to continue on its course to the destination point until going off course again due to the uneven terrain and wind. Even though between the start and end points there were some noticeable issues, once the

gondola reached the end point within 4 meters, the software shut down the motors and marked the waypoint reached and flight complete.

From this second ground test, we came to two conclusions. First, we concluded that the gondola mounted on the cart was not an optimal setup for testing, and that the dynamics would be much different when actually flying the blimp with the gondola attached to the shell in the air. Second, we determined that the differential in thrust between the left and right motors while turning had to be made smaller, so the gondola could turn more slowly and allow the software sufficient time to react.

6.6.2.3 Third Ground Test

Our third ground test was also performed on the top level of the East Hall parking garage at WPI. The purpose of this third test was to verify that the changes to the motor thrust differentials improved the path of the gondola from start to end. We also wanted to test having the gondola navigate between a queue of GPS waypoints instead of navigating to only one waypoint. Figure 63 shows the recorded path from the start to the first waypoint, Figure 64 shows the recorded path from the first to second waypoint, Figure 65 shows the recorded path from the second to end waypoint, and Figure 66 shows the paths between each waypoint for the entire test.

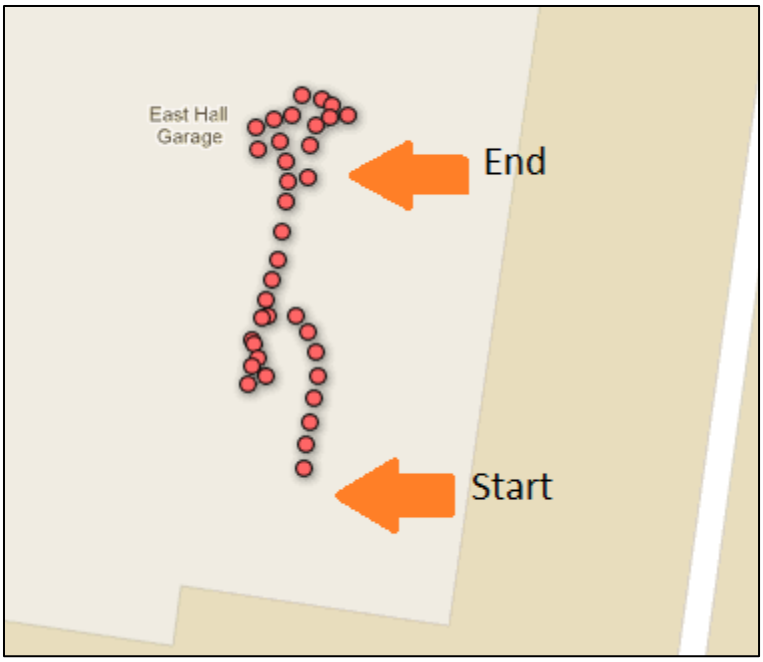


Figure 63: Path from Waypoints A to B During Third Ground Test

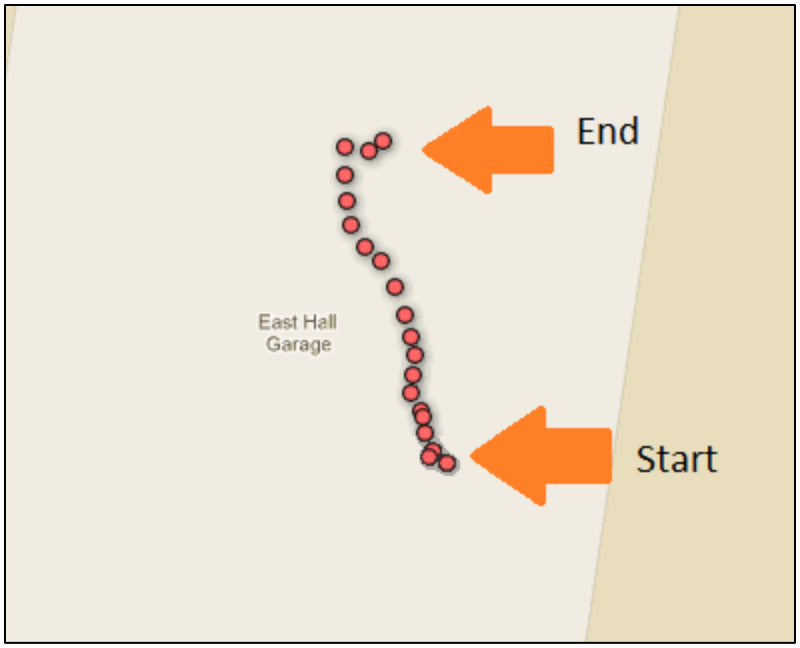


Figure 64: Path from Waypoints B to C during Third Ground Test

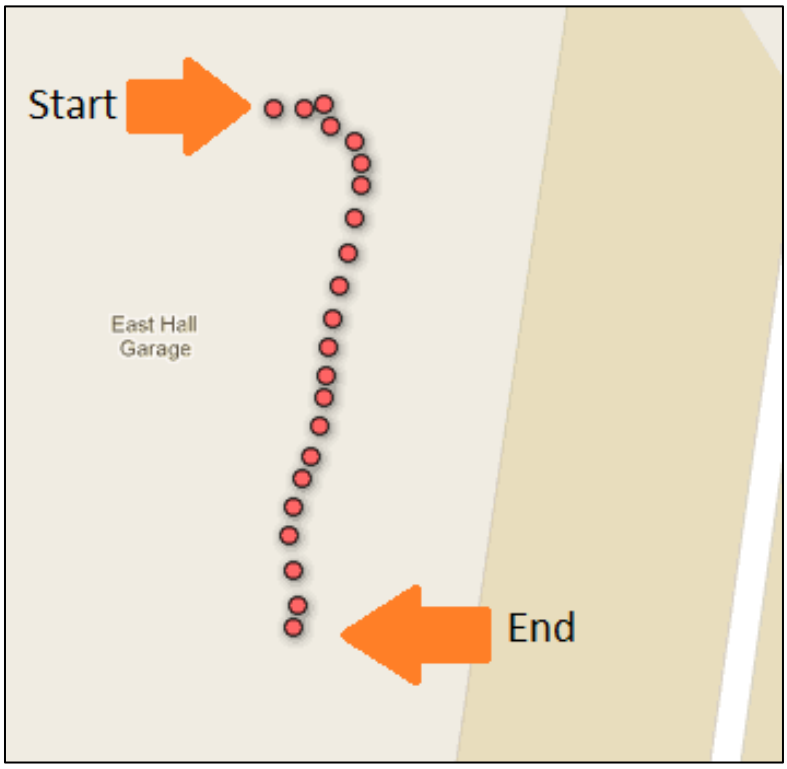


Figure 65: Path from Waypoints C to D during the Third Ground Test

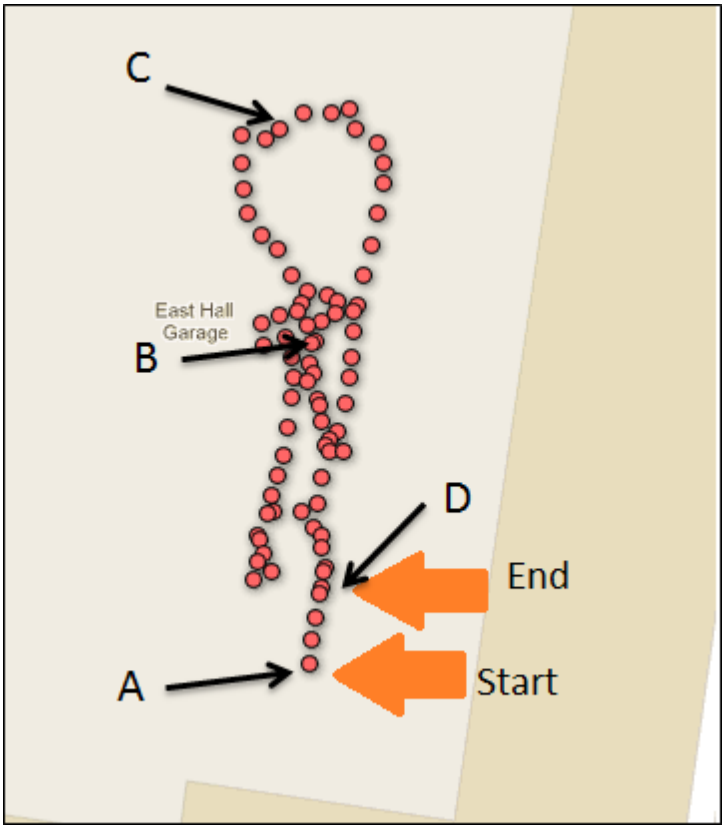


Figure 66: Complete Recorded Path from Waypoints A to D during the Third Ground Test

From the plotted data, it is most useful to examine each individual path from waypoint to waypoint as opposed to all of them plotted together. During the test, between waypoints A and B, the gondola on the cart veered off course due to the uneven terrain of the parking garage and the wind. Between waypoints B & C and C & D, the gondola was able to steadily travel without significantly veering off course. This is reflected in the plotted data, since in Figure 63 there are a couple of clusters of GPS points that were caused by the gondola veering off course. The plotted data of the following two paths however do not contain any significant clusters of GPS points. Both during the test, and by analyzing the log files recovered from the S3 and plotting the data, we noticed that there was a significant improvement over the first two ground tests that we performed. This was due to the combination of improvements that we had made to the software as discussed earlier.

The second goal of this test was to see how the autonomous software would handle navigating between a queue of GPS waypoints. We noticed that the software was able to successfully update its destination every time it reached a waypoint, that the software was able to

successfully control the drive system to get to its destination, and that at the final waypoint the software turned off all the motors.

From this third and final ground test, we concluded that to be able to improve the autonomous software further, we would need to perform flight testing. We expected that the dynamics of the blimp in flight would be much different from the gondola on a cart. Also, during our three ground tests we demonstrated and verified the functionality of the autonomous control algorithm, so the next step was to fine tune the motor speeds.

6.6.3 Flight Test

In this section we describe our flight testing procedures and results regarding the autonomous control portion of the project. We also explain how we analyzed the results of each test and made changes to rectify any issues to improve the results of the next test.

Our outdoor flight test was performed on the Alumni Field at WPI. The purpose of this test was to demonstrate and verify that the autonomous control software would work with the gondola attached to the shell. We also wanted to verify that while the gondola was attached to the shell, that the system was still able to navigate to a sequence of GPS waypoints as in the ground tests described in 6.6.2. Figure 67 shows the recorded path of the flight test from the starting point to the end. The A, B, and C arrows indicate the approximate positions of the three GPS waypoints that the blimp traversed during its flight. Figure 68 shows the flight states that were executed during the flight test. A value of 1 indicates a left turn, -1 indicates a right turn, and 0 indicates moving straight.

Unfortunately, for this flight test we were unable to secure a working tail motor for the blimp. The reason being that the tail motor that the 2011 AY team had purchased had stopped working, and we did not have sufficient time to acquire a new one by the time we performed this flight test. To deal with this problem we "guided" the blimp during the flight test so that the wind would not carry it away, since the lack of a tail motor meant that turning to compensate for the wind was difficult for the blimp, even under manual control. Our guidance of the blimp did not detract from the flight being autonomous however, since the only assistance we gave the blimp was to help it while turning.

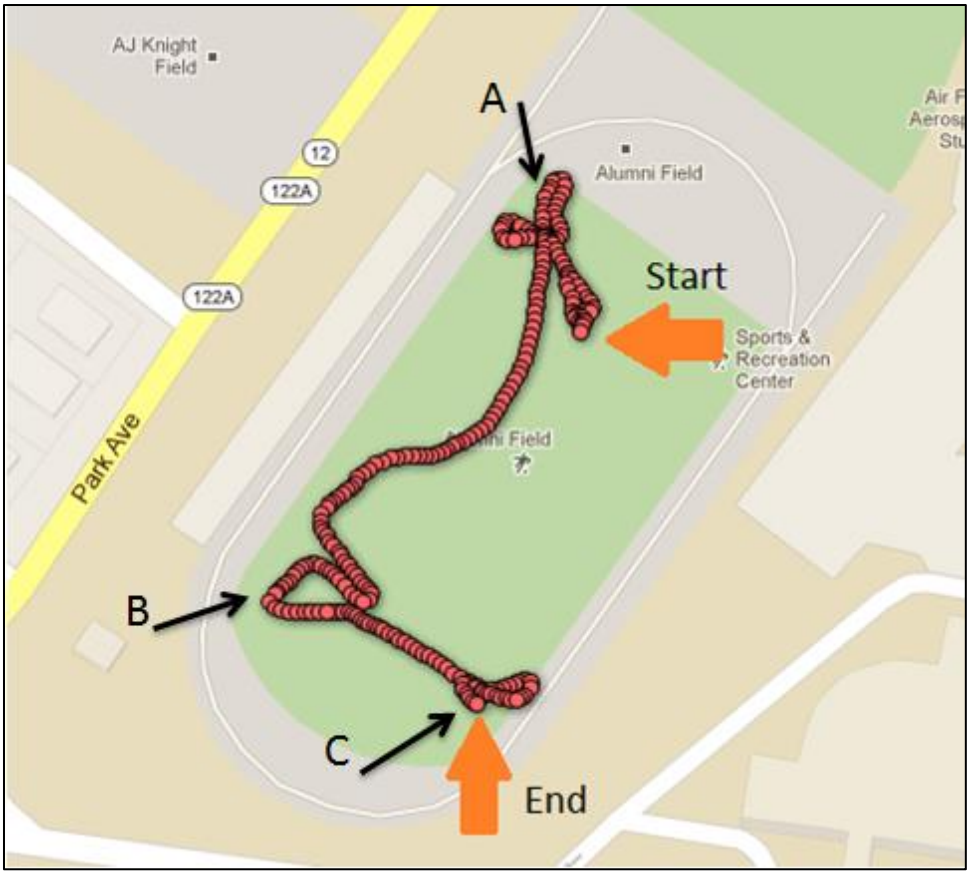


Figure 67: Recorded Path of the First Outdoor Flight Test

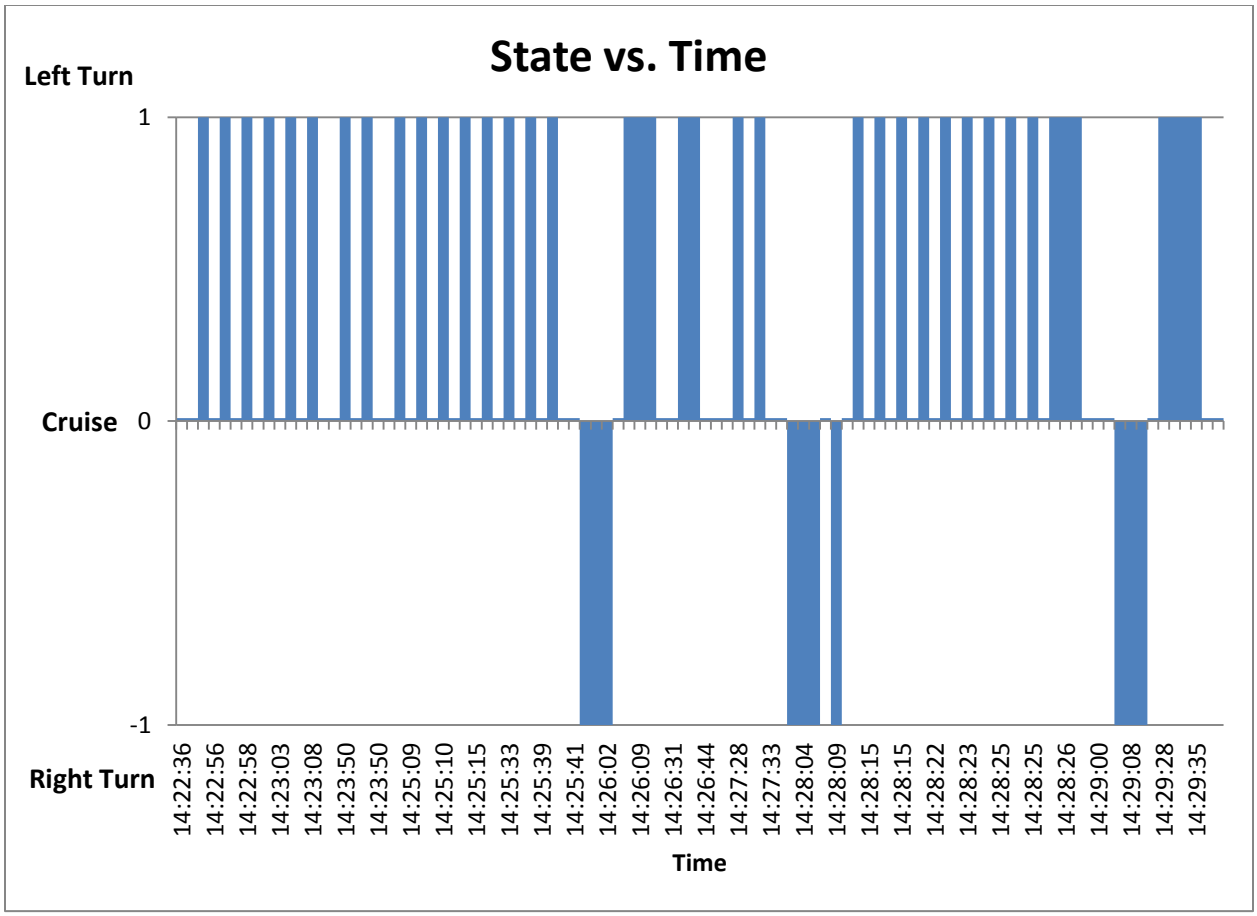


Figure 68: Drive System States During First Outdoor Flight Test

From the plotted data we observed that our goal of non-optimized autonomous flight was met. While the software was able to successfully drive the blimp from waypoint to waypoint and stop at the final waypoint, the path that it took was not optimal. From the start position to waypoint A, the plot shows that the blimp took an indirect path until it finally reached the waypoint. The path the blimp took from A to B is slightly more direct, but the blimp still veered off course slightly. The final path from B to C was more direct, but as the blimp closed its distance to C, it missed the waypoint and had to turn around to reach it, similar to point A. We also noticed that the blimp was attempting to turn either right or left for a majority of the time during flight. This can be explained because even though we were guiding the blimp during this test, the wind was still able to influence its position slightly. While the path between each waypoint was non-optimal, we successfully demonstrated that the blimp was able to navigate between a sequence of GPS waypoints with a non-optimized flight path.

From this test we concluded that to improve the autonomous control we would need to have a working tail motor and we would need to make extensive modifications to the software to allow for a more optimized flight path. Since the time constraints of our project did not allow for either of these to be accomplished, and since we already met our goal of non-optimized autonomous flight, we decided to leave these tasks to a future team.

6.6.4 Fulfillment of Requirements

This section will review the requirements for the autonomous flight goal, how each was tested, and whether the test passed or failed.

ID	Requirement	Need	Test	Result
AF-01	The Android device must be able to control the speed of the motors and rotation of the servo.	Control the drive system using Java code within the Android app	Controlled lab test with the Android device, IOIO, motors, and servo to demonstrate control with the Android app	Met
AF-02	The system must update the blimp's current position every second.	The current position is needed to calculate how to get to the destination point	Log every GPS reading from the Android's onboard GPS to plaintext files and examine the timestamps of each reading	Met
AF-03	The system must update the blimp's current altitude at least every second.	The current altitude is needed so the blimp does not fly higher than 100 feet or crash into the ground	Log every barometric pressure reading and altitude calculation to plaintext files and examine the timestamps of each reading	Met
AF-04	The system must update the blimp's current heading at least every second.	The current heading is needed so the blimp closes the distance to its destination point	Log every heading reading to plaintext files and examine the timestamps of each reading	Met
AF-05	The blimp must be able to autonomously navigate between two GPS waypoints.	Project goal	Ground and flight testing	Met

Table 15: Requirements Fulfillment Table for Autonomous Flight

6.7 Summary

In this chapter we discussed the design, implementation, and testing of the autonomous control software. We covered the needs and requirements of the system, and then we discussed the architecture and major control algorithm of the system. We also discussed the structure of the software and provided descriptions of each class that was written for the Android app. Finally,

we discussed the results we found through ground and flight testing, and we determined which requirements were and were not met by the system we implemented.

7 Power System

7.1 Introduction

This chapter discusses the design, implementation, testing, and results for the blimp's redesigned power system. First we briefly discuss the overall design of the system, followed by a discussion of the system needs and requirements. Next we discuss the available system architectures and the research we conducted to finalize the design of the system. Lastly, the final system design, the tests, and the results will be presented.

7.2 System Design

The main purpose of the power system is to measure and log the energy usage of the blimp's drive system while regulating the energy drawn from the power supply. The power system consists of two main units, a Power Distribution Board (PDB) and a Microcontroller Board (MCB). Both units can be further divided into sub-units that perform specific individual tasks. By integrating the functionalities of the sub-units, the power system is able to perform its main function.

The Power Distribution Board, shown in Figure 69, consists of three units:

- Main Power Distribution Unit
- Voltage Regulation Unit
- Data Measurement Unit

The main power distribution unit is responsible for providing the main power path for the current flow to the motor drive system, reducing the voltage drop from the batteries to the motors, and preventing the batteries from accidentally discharging into each other. The voltage regulation unit is responsible for lowering the batteries' voltage to a level needed to power the blimp's additional components. The data measurement unit provides an interface to the Microcontroller Unit and the user that relays instantaneous data about the energy usage of the drive system.

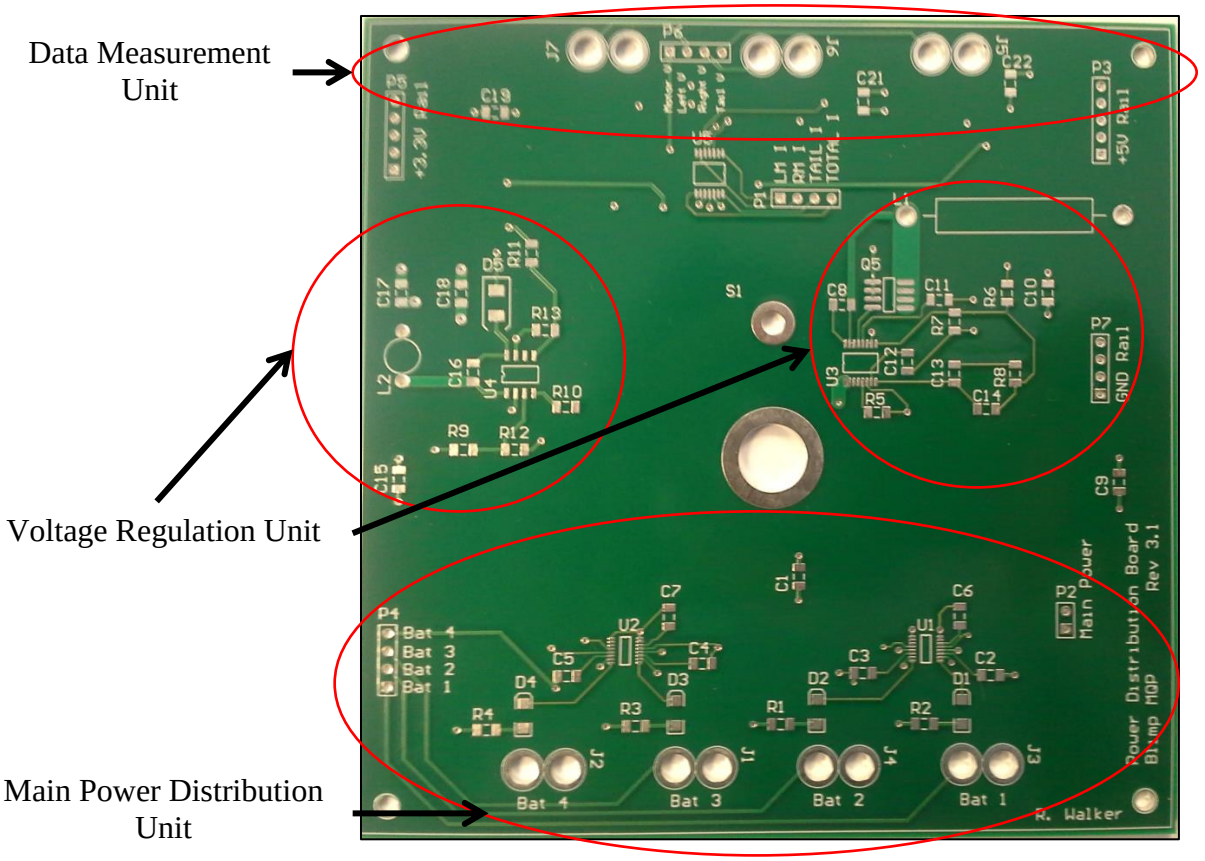


Figure 69: Power Distribution Board

The Microcontroller Board, shown in Figure 70, consists of multiple units, but for the purpose of this project specific units were used to fulfill the requirements of the power system. The specific units will be identified and explained later in this chapter.

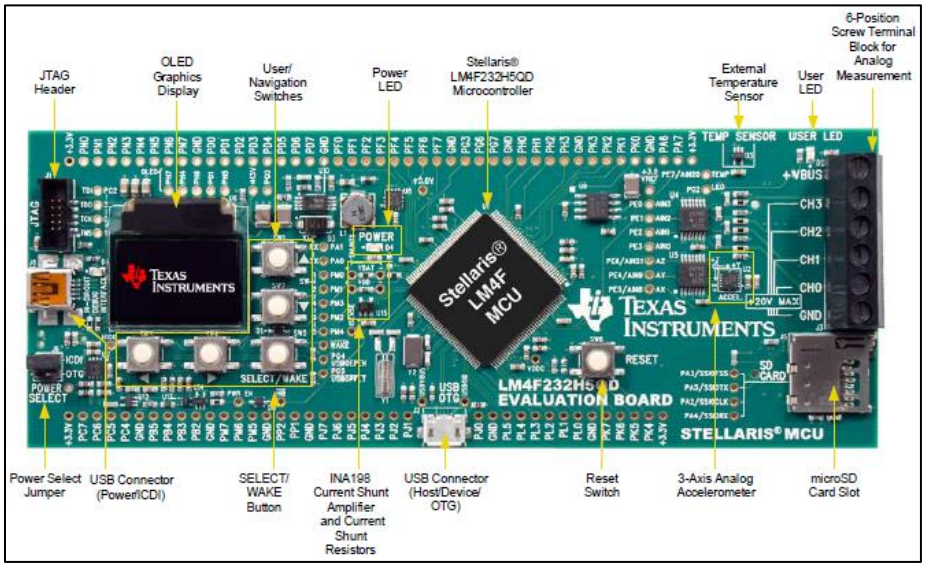


Figure 70: Microcontroller Board

7.2.1 System Needs

The energy used by a system can be expressed as the power used by the system over a specific amount of time. The instantaneous electrical power used by the blimp is the product of current used by a component and the voltage drop across it, as shown in Equation 1.

$$P(t) = I(t) * V(t)$$

Equation 1: Electrical Power Equation

The power system needed to determine the energy usage by calculating the instantaneous power used by the drive system at various times then summing the power readings over the flight time, shown in Equation 2.

$$Energy\ usage = \sum_{time\ interval=0}^{flight\ time} P\ (t)$$

Equation 2: Energy Used

The previous power system had an approximate voltage drop of 0.5V from the power supply to the motors, which limited the speed of the motors. The new power system needed to lower this voltage drop. Additionally, the power system needed to serve as a power bridge between the power supply and the other components on the blimp.

7.2.2 System Requirements

From the system needs stated above, the following requirements for the power system were derived[7]. Each requirement was broken down into quantifiable details, and is shown in Table 16.

[7] The requirements in this section may not fully reflect the requirements in Section 3.4.4due to the changing nature of the project.

Power System Requirements	
PS-01	The power system shall supply the blimp with energy for a minimum of 90 minutes when the left and right motors are operating at 0.828 lbs. of thrust.
PS-02	The difference between the power supply's voltage and the voltage measured at the left, right, and tail motors shall be no more than 0.2V.
PS-03	The power system shall be able to measure the battery voltages between 0-13V ± 0.1V.
PS-04	The power system shall measure the current individually drawn by the left, right, and tail motors between 0-16A ± 0.1A.
PS-05	The power system shall calculate the average current draw of the left, right, and tail motors accurately to 0.1A throughout the blimp's flight time, using logged values.
PS-06	The power system shall calculate the instantaneous power used by the left, right, and tail motors with a minimum resolution of 0.1 W.
PS-07	The power system shall calculate energy used by the blimp by summing the instantaneous power over the flight time with a resolution of 0.1 Watt-Hours, using logged values.
PS-08	The power system shall measure time with a minimum resolution of 1s.
PS-09	The power system shall measure the temperature inside the gondola accurately to 0.1°C.
PS-10	The power system shall record the measured and calculated voltage, current, power and energy values into a non-volatile memory.
PS-11	The power system shall be able to display the measured and calculated voltage, current, power and energy values.
PS-12	The final design for the power system shall be implemented on a printed circuit board(s) (PCB).

Table 16: Power System Requirements

7.3 System Architecture and Trade Studies

This section will discuss the design considerations, iterations, and revisions that the power system underwent throughout the project. The design will be discussed from a top-down methodology, starting from the high-level system design and ending with the implementations for the system sub-units.

7.3.1 High-Level System Design

We used a modular approach to design the power system. This approach allowed the system to be designed in sections. By designing each section separately we were able to focus on the implementation of a particular section before determining how it would interface with the system as a whole. The high-level system underwent three revisions before the design was finalized. Each revision will be discussed below.

7.3.1.1 First Revision

The initial high-level design, shown in Figure 71, consisted of three units that performed their individual roles and interfaced with each other to fulfill the requirements of the power system.

The power supply unit consisted of the batteries and the discharge control module. The unit's purpose was to provide a power path from the batteries to the drive system and additional components. The data measurement unit was composed of a voltage measurement module, a current measurement module, and a temperature measurement module. The unit was intended to convert energy usage data into a form that could interface with the data processing unit. Lastly, the data processing unit consisted of a data calculation module and a data storage module. The purpose of the unit was to access the data provided by the data measurement unit, then perform energy calculations based on the received data.

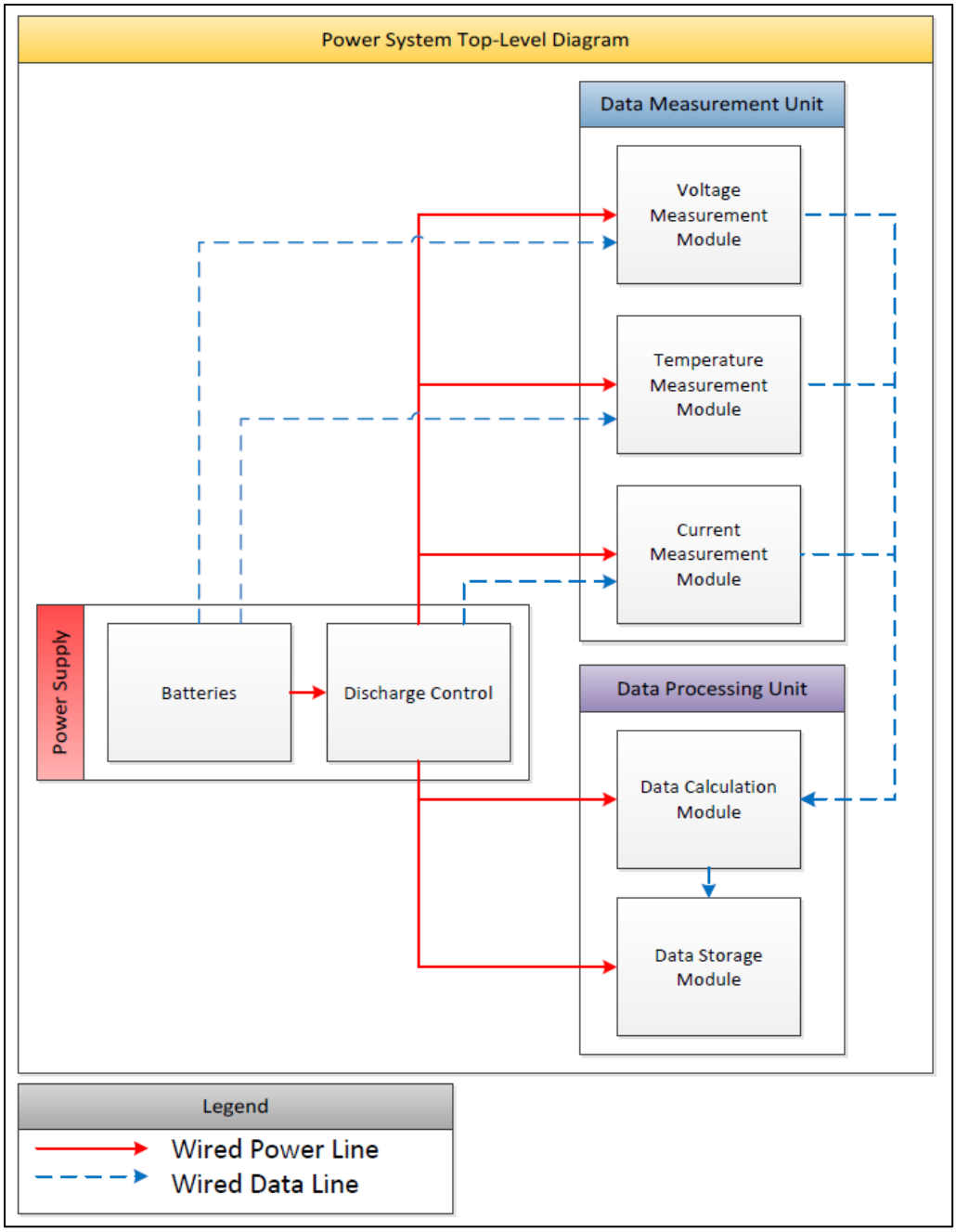


Figure 71: Power System High-Level Rev.1.0

7.3.1.2 Second Revision

The second revision of the high-level design, shown in Figure 72, consisted of the three units in the previous revision and added a data display unit. The units that were in the previous revision performed the same functions; however, the modules in each unit were modified.

A main power bus module, +5V regulation module, and +3.3V regulation module were added to the power supply unit. The voltage and current sensor modules in the data measurement

unit were further divided into smaller modules that specified the sensory data that was to be measured. The modules in the data processing unit were modified to reflect the components used in that unit.

The function of the added data display unit served as an interface for the user to monitor the energy usage of the blimp. The unit contained two modules that displayed information to the user via a Liquid Crystal Display (LCD) and Light Emitting Diodes (LED).

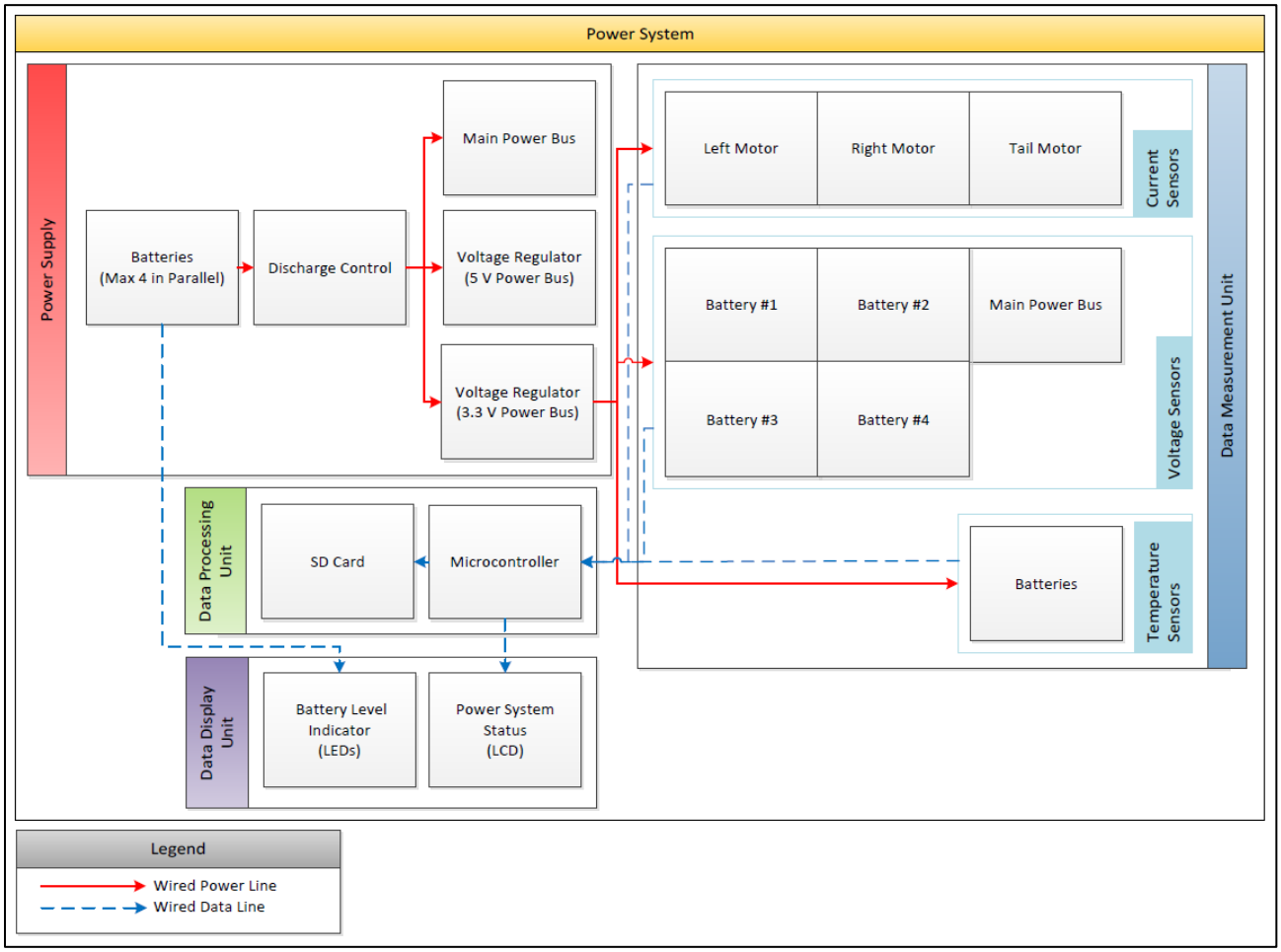


Figure 72: Power System High-Level Rev.2.0

7.3.1.3 Third Revision

There were three modifications in the third revision of the high-level design, as shown in Figure 73. The power supply unit was divided into two units, a main power distribution unit and a voltage regulation unit. The temperature sensor module from the data measurement module was removed, and a temperature measurement unit was added. Additionally, the units were grouped in the design according to whether they resided in the power distribution board or the microcontroller board.

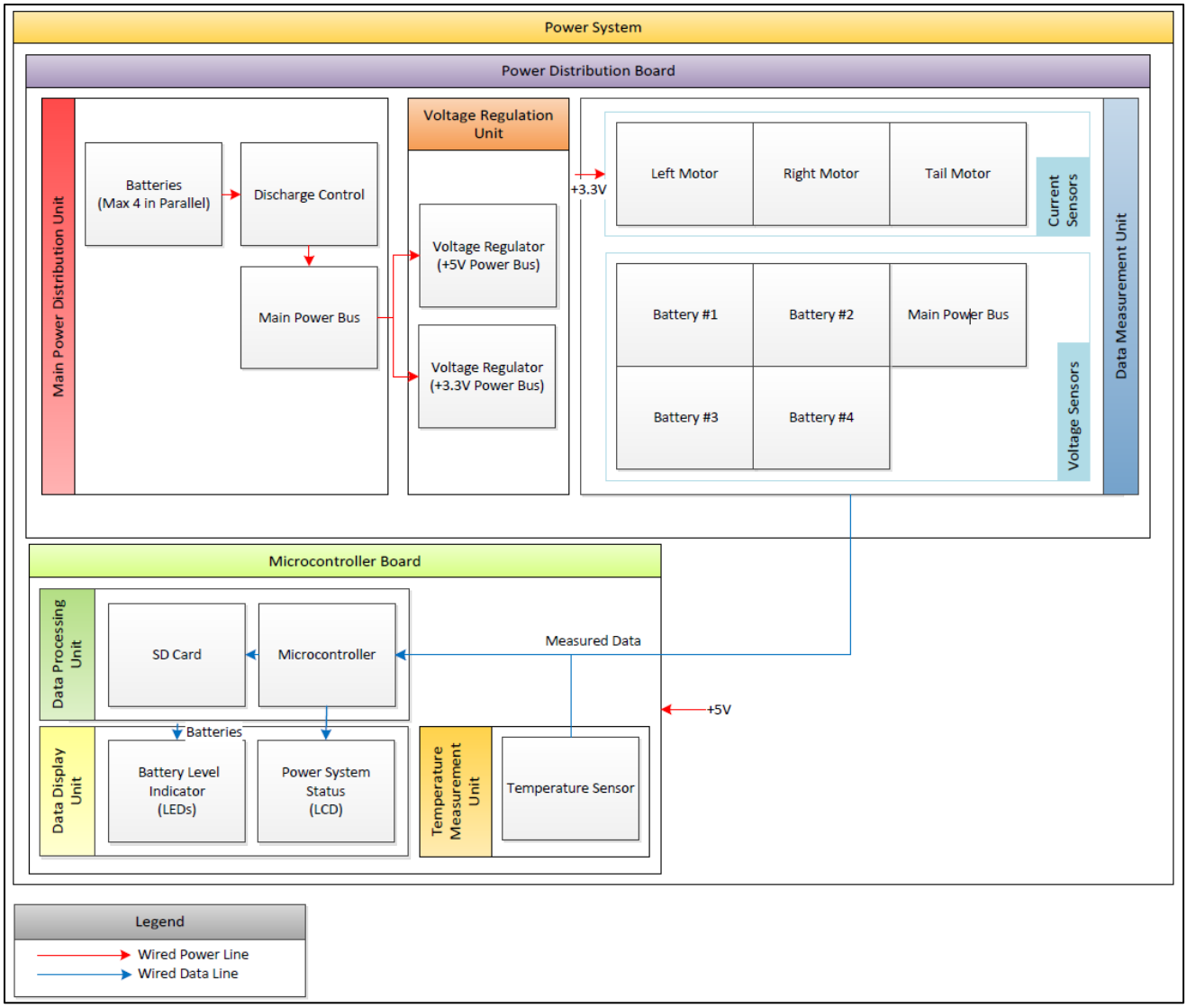


Figure 73: Power System High-Level Rev.3.0

7.3.2 Mid-Level System Design

This section discusses the five sub-units in the power system. We will describe the possible implementations of modules contained in each sub-unit. Additionally, the purpose of each sub-unit will be described in further detail.

7.3.2.1 Main Power Distribution Unit

The main power distribution unit provides a current path from the batteries to the drive system and the voltage regulation unit while minimizing the voltage drop from the batteries to the motors. As shown in Figure 74, the unit contains three modules that operate in unison to fulfill the purpose of this unit.

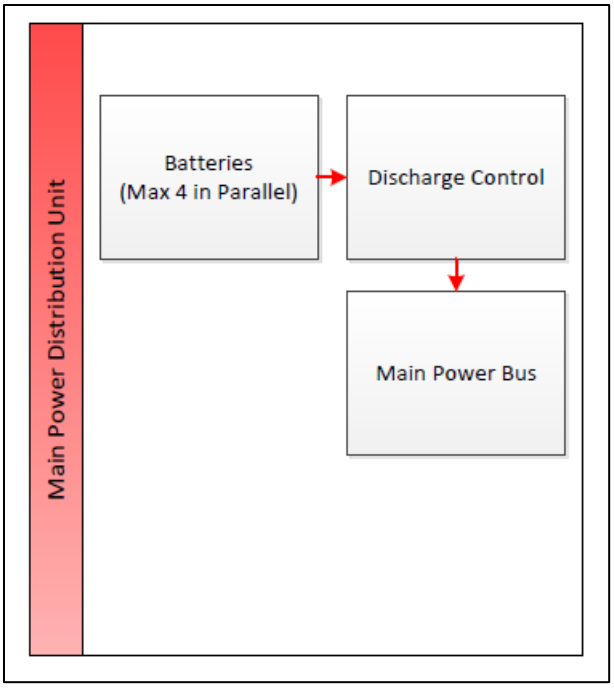


Figure 74: Main Power Distribution Unit

7.3.2.1.1 Batteries

The batteries were required to supply the blimp with energy for a minimum of 90 minutes. The 2011 AY team determined that the most important criteria for the batteries were the charge capacities and weights [1]. The team researched various batteries types and compared various batteries, shown in Table 17. The team chose the "ZIPPY Flightmax" batteries because they had the lowest cost per capacity, as indicated in Figure 75.

Battery:	Battery Chemistry	Cost [$USD]	Voltage [V]	Capacity [mAHr]	Weight [lbs]	Capacity per Weight [mAHr/lbs]	Cost per Capacity [$/AHr]
Tenergy Li-Ion 18650	Li-Ion	99.98	11.1	7800	0.938	8320.00	$12.82
NiMh 12V	NiMh	87.99	12.0	10000	3.350	2985.07	$8.80
ZIPPY Flightmax 4400mAh 4S1P 15C	LiPoly	28.48	11.1	4400	0.927	4746.49	$6.47
Rhino 4900mAh 3S1P 11.1v 20C	LiPoly	39.95	11.1	4900	0.776	6314.43	$8.15
ZIPPY Flightmax 5000mAh 3S1P 20C	LiPoly	31.72	11.1	5000	0.864	5785.70	$6.34
Turnigy nano-tech 1800mah 3S 20~40C Lipo AIRSOFT Pack	LiPoly	11.77	11.1	1800	0.331	5443.00	$6.54

Table 17: Batteries Comparison [1]

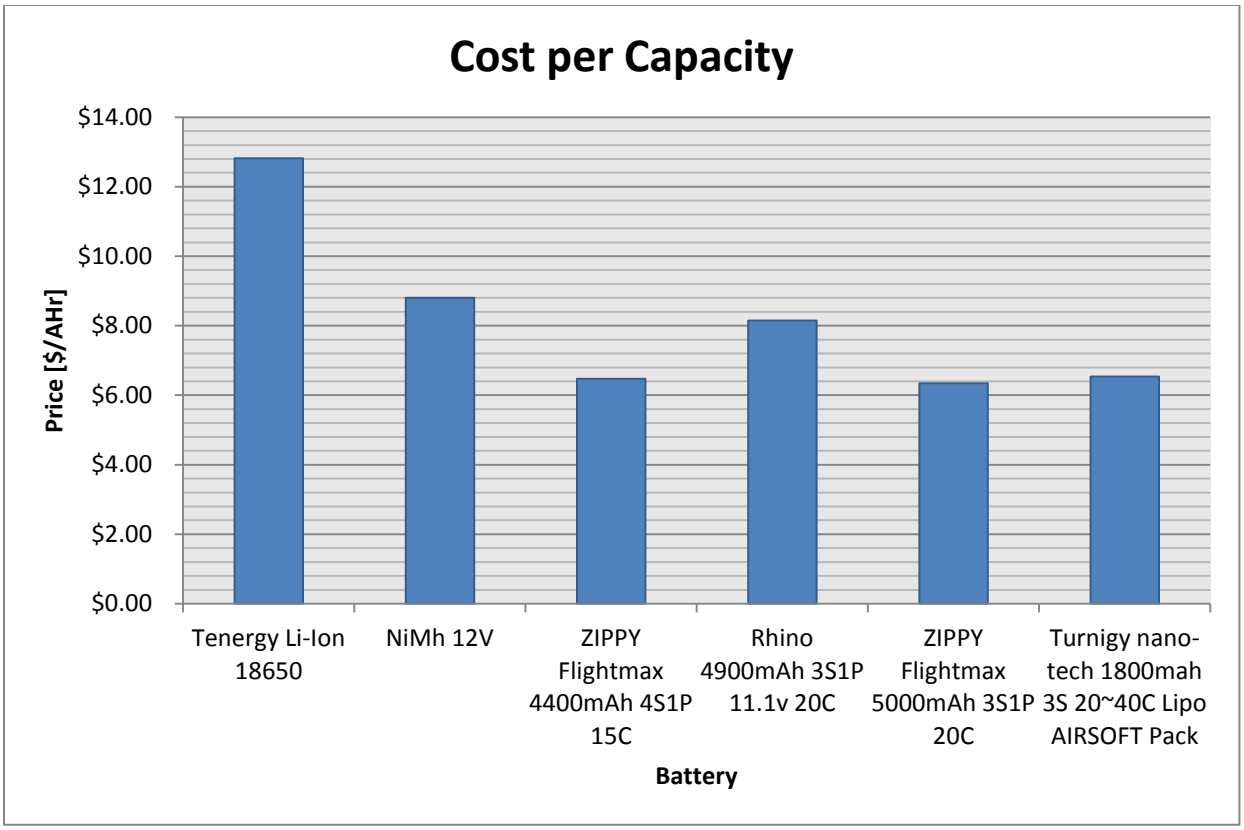


Figure 75: Cost per Battery

The 2011 AY team decided to use two "ZIPPY Flightmax" batteries in parallel. The team also created a mission module that contained two extra batteries that would serve as a secondary battery pack for the blimp if increased flight time was needed. As mentioned in Section 2.1.4 the team attempted to achieve an endurance flight time of 90 minutes. However, it was not documented whether or not they achieved this goal.

Based on the 2011 AY team's components, our team performed calculations to determine if the 2011 AY team could have achieved their endurance flight time goal. The results from our calculations predicted that batteries selected by the 2011 AY team would not achieve their goal.

To achieve the endurance flight goal, our team performed additional research to find alternate batteries that could sustain flight for 90 minutes.

7.3.2.1.2 Discharge Control

The discharge control module was required to prevent the batteries from discharging into each other due to differences in each battery's voltage levels. Our team researched possible ways to implement this module, and the results were narrowed down to two choices: Diode ORing using Schottky diodes, or Diode ORing using an Ideal Diode integrated circuitry.

Diode ORing using Schottky diodes:

The 2011 AY team implemented discharge control using this method. This implementation functions by having Schottky diodes in series with each battery before the connection to the drive system, as shown in Figure 76.

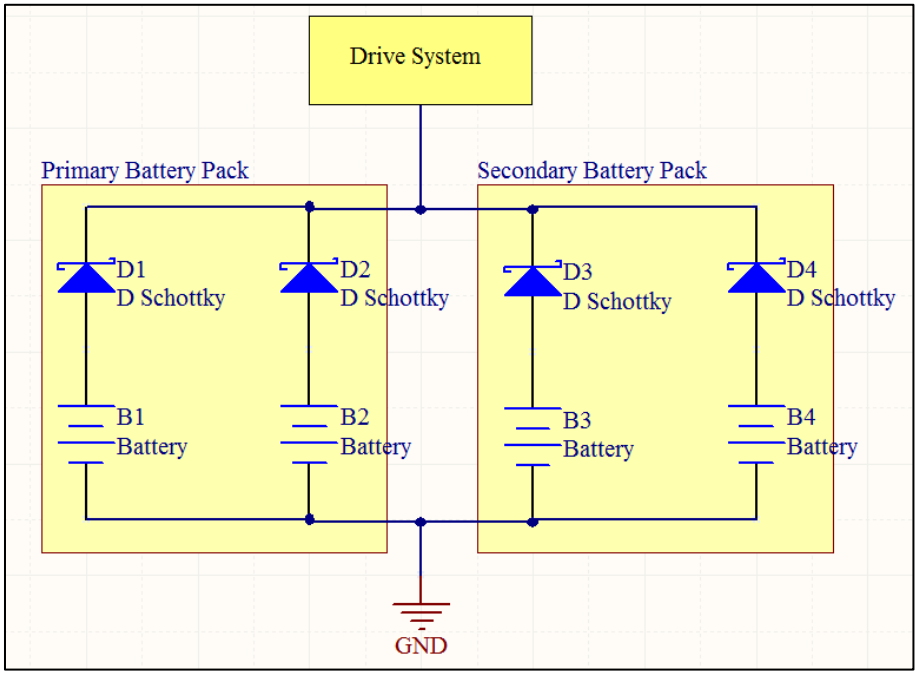


Figure 76: Diode ORing using Schottky diodes

This implementation is the more simplistic of the two because only four components would be needed to fulfill its purpose. However, the disadvantage of this method is that the diodes have a 0.15V to 0.45V voltage drop associated with them. This results in the voltage being provided to the drive system to be lower than the intended voltage of the batteries. Furthermore, the speed of the motors is proportional to voltage provided to them, thus the voltage drop limits the maximum speed of the motors.

Diode ORing using an Ideal Diode integrated circuitry:

The second implementation of this module utilizes an integrated circuit (IC) in conjunction with a power metal-oxide semiconductor field-effect transistor (MOSFET) to "idealize" diode ORing. An ideal diode, which resembles a unidirectional switch, has no voltage drop across it, as shown in Figure 77. This would allow the batteries to provide their full voltage to the drive system.

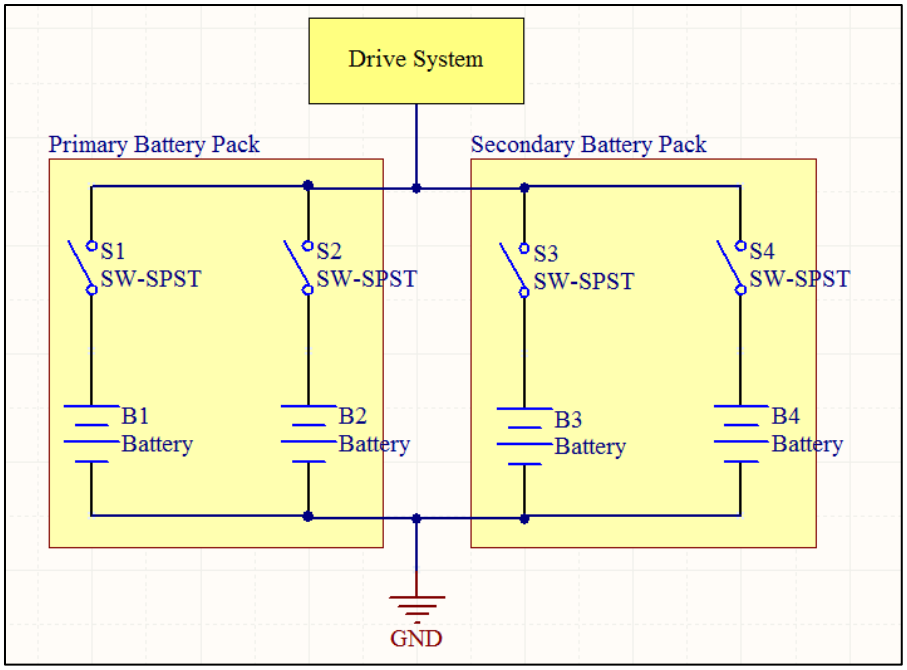


Figure 77: Ideal Diode ORing using switches

The circuit in Figure 77 can be created by replacing the switches with MOSFETs. When a MOSFET is switched on it has a low voltage drop across it attributed to its low on-resistance. This voltage drop is calculated by Equation 3 below.

$$V_{MOSFET} = I(t) * R_{on}$$

Equation 3: MOSFET Voltage Drop

The 2011 AY team documented that each motor draws 5.41A of current when operating at 0.828 lbs of thrust [1]. This potentially results in a battery supplying 16.22A[8] of current which passes through the MOSFET to the drive system.

The power system requirement PS-02 stipulates that the voltage drop between the batteries and the motors shall not be more than 0.2V. Based on the above motors' characteristics, it was calculated that a MOSFET with a R_{on} less than or equal to 12.33mΩ in conjunction with an ideal diode IC would fulfill the purpose of this module.

7.3.2.2 Voltage Regulation Unit

The voltage regulation unit serves to convert the voltage provided by the batteries to a lower voltage level usable by additional components. As shown in Figure 78, the unit contains two modules that fulfill the purpose of this unit.

[8] The total current drawn by the left, right, and tail motor.

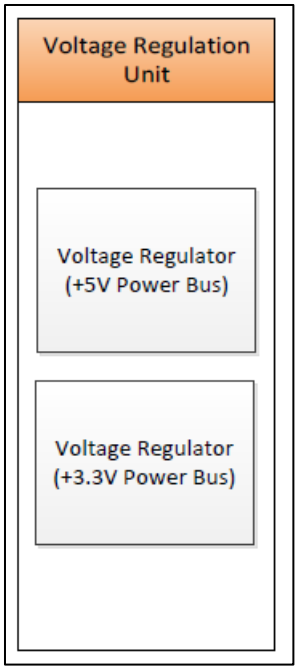


Figure 78: Voltage Regulation Unit

Our team researched possible implementations for both modules, and it was concluded there were two viable methods for the modules: linear regulation or switching regulation.

7.3.2.2.1 Linear Regulation

The two modules in this unit could have been implemented via a linear regulator IC. The input to the IC would be the main power bus's voltage, and then the IC would step down the voltage to the required level, as shown in Figure 79.

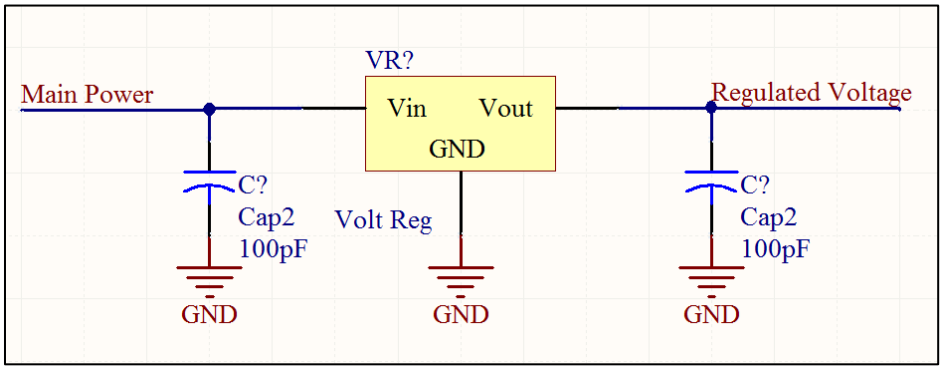


Figure 79: Linear Regulation Implementation

This implementation would require fewer components than switching regulation. However, the power loss attributed to this implementation would be higher than switching regulation. The power loss is expressed by Equation 4, where V_{drop} is the voltage difference

between the main power bus and the regulated voltage, and *I(t)* is the instantaneous current that passes through the IC.

$$P_{loss} = I(t) * V_{drop}$$

Equation 4: Power Loss of Linear Regulation

7.3.2.2.2 Switching Regulation

An alternate implementation of this module could have been implemented via a switching regulator IC. The IC would close and open a switch that connects the main power to the regulated voltage, shown in Figure 80.

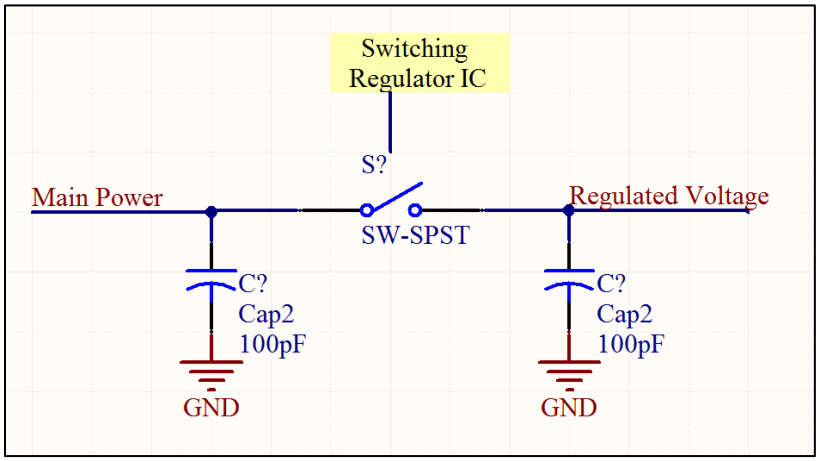


Figure 80: Switching Regulation Implementation

The regulated voltage would be determined by the duty-cycle (the percentage of the time the switch was closed) of the switch, expressed in Equation 5, where *D* is the duty-cycle and V_{in} is the input voltage, i.e. main power bus voltage.

$$V_{regulated} = D * V_{in}$$

Equation 5: Switching Regulation Calculation

This implementation requires more components than linear regulation. However, the power loss attributed to this implementation is significantly less than linear regulation.

7.3.2.3 Data Measurement Unit

The data measurement unit measures the energy used by the drive system and provides an interface for the data processing unit to acquire the energy usage data. This unit is divided into two sub-units: current sensing and voltage sensing, as shown in Figure 81. Both sub-units measure their respective sources, and the data is combined by the data measurement unit. The following two sub-sections detail the possible implementations of both sub-units.

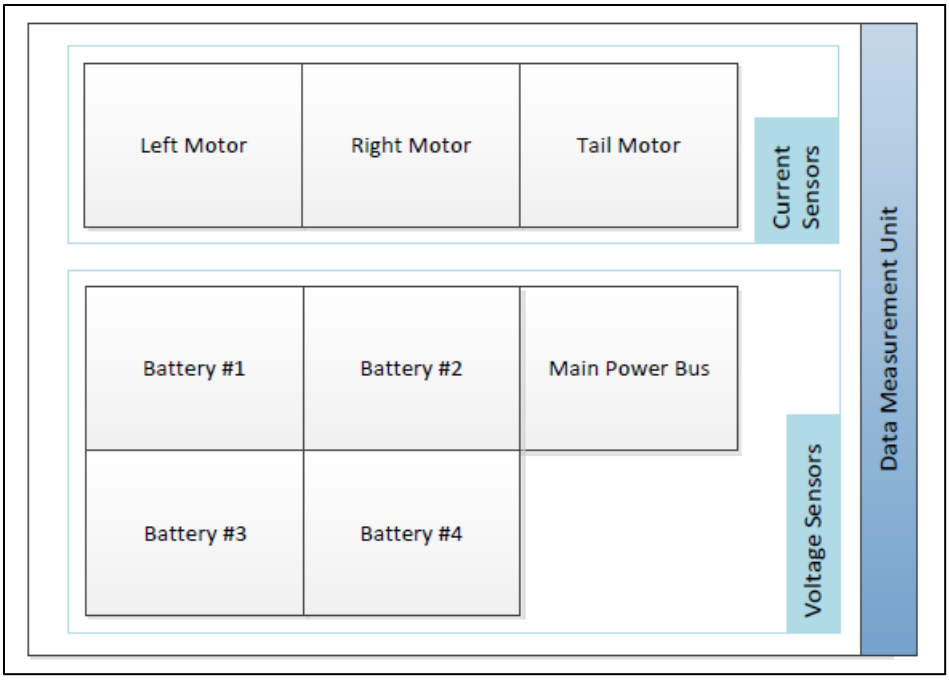


Figure 81: Data Measurement Unit

7.3.2.3.1 Current Sensing

This module needed to determine the current draw of the drive system's motors. Our team researched the various techniques of achieving this, and results were narrowed down to two viable methods: shunt resistors and Hall effect sensors.

Shunt Resistors:

Current shunt resistors are low resistance precision resistors used to measure electrical currents by the voltage drop those currents create across the resistance, as shown in Equation 6.

$$I_{measured} = \frac{V_{drop}}{R_{shunt}}$$

Equation 6: Shunt Current Calculation

Shunt resistors are usually on the order of milliohms (mΩ), which results in a low voltage drop when current passes through them. In most cases, the voltage drop has to be amplified to a level that is measureable in order to determine the current accurately. A possible implementation of shunt resistors in this module is shown in Figure 82.

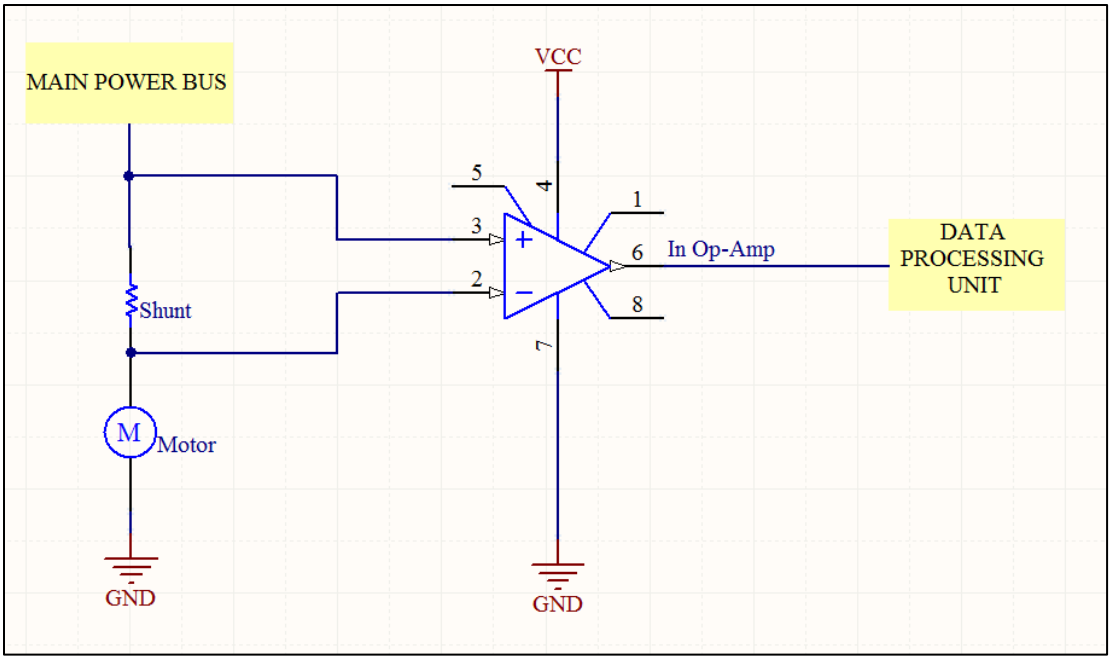


Figure 82: Current Shunt Implementation

The voltage across the shunt resistor would be measured by an instrumentational operational amplifier (op-amp), which amplifies the value, then outputs the amplified value to the data processing unit.

Hall Effect Sensor ICs:

A Hall effect sensor is a transducer that varies its output voltage in response to a magnetic field. When electricity passes through a conductor it produces a magnetic field that varies with current. The Hall effect sensor uses this phenomenon to measure the current without interrupting the circuit. Various companies integrate Hall effect sensors into ICs to provide a convenient current measuring tool. A possible implementation of this module using a Hall effect sensor IC to measure the current going to the drive system, and is shown in Figure 83.

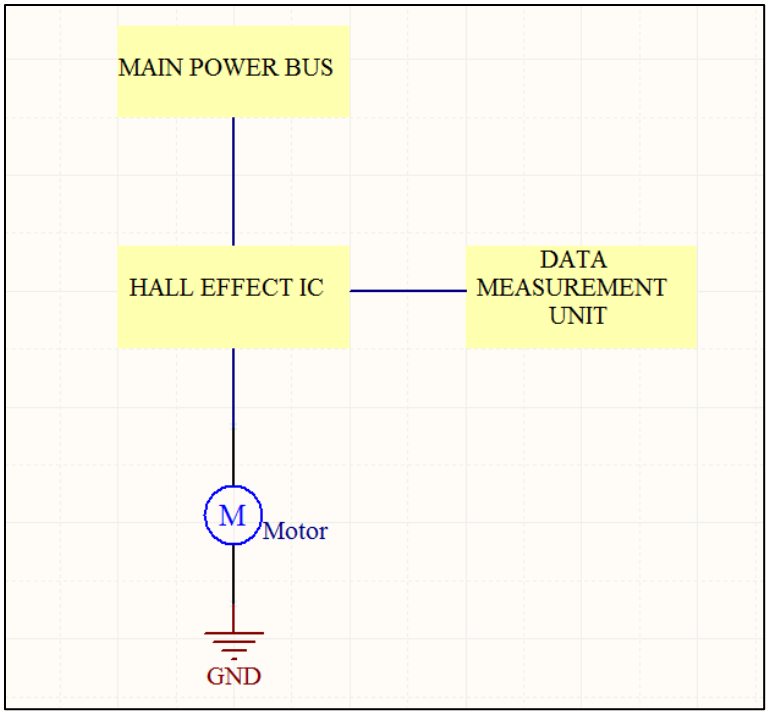


Figure 83: Hall Effect IC Implementation

7.3.2.3.2 Voltage Sensing

This module needed to determine the voltage level across each battery and each motor in the drive system. Our team researched the various implementations of this module, which will be discussed below.

The voltages across the batteries and the motors could simply be measured with respect to ground. However, in order to prevent damage to the data measurement unit, the voltage measurements needed to be converted to a suitable form. The maximum voltage level tolerable by the data measurement unit is lower than the battery's nominal level. To step down the voltage level a voltage divider network could be used, as shown in Figure 84.

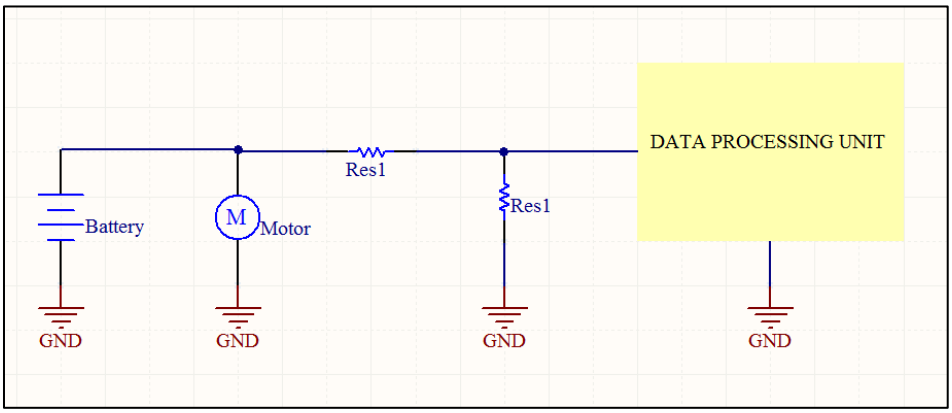


Figure 84: Voltage Conversion for Data Processing Unit

A concern that may arise with the above implementation is that the voltage signal at the data processing unit may be attenuated depending on the input impedance of the data processing unit. The schematic in Figure 84 can simplified to represent the voltage signal and the source impedance, as shown in Figure 85.

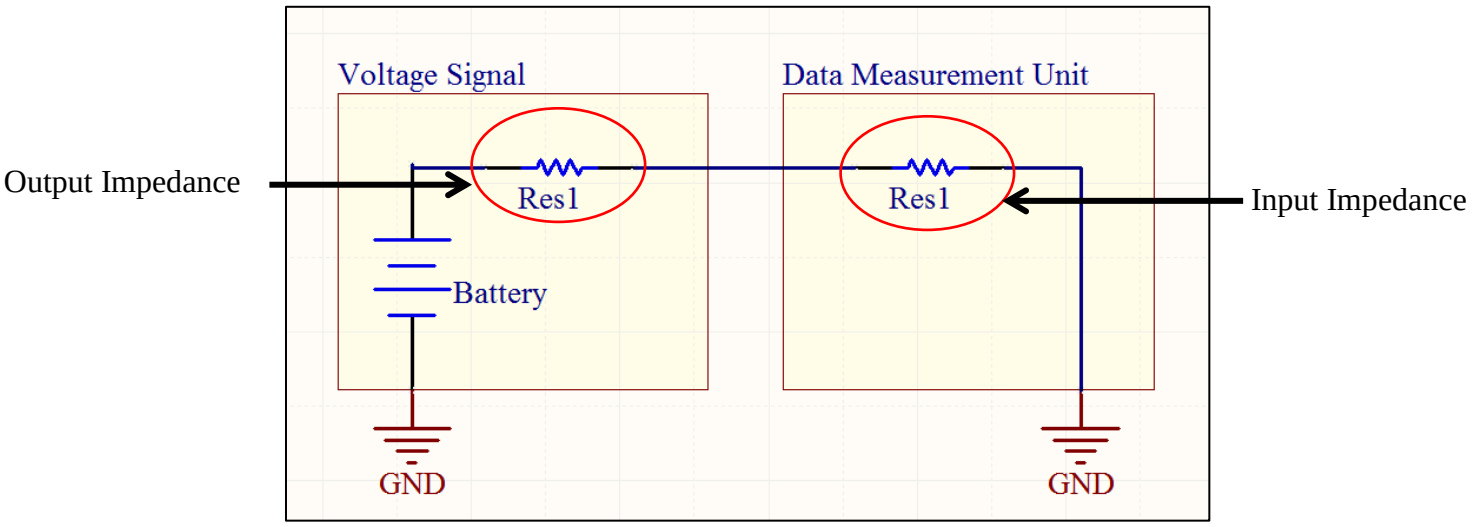


Figure 85: Simplified Voltage Conversion for Data Processing Unit

The percentage of the voltage signal that reaches the data measurement unit is expressed by Equation 7.

$$\%_{signal} = \frac{Z_{input}}{Z_{input} + Z_{ouput}}$$

Equation 7: Voltage Signal Attenuation

For maximum transfer of the voltage signal to the data processing unit, it is favorable to make Z_{input} infinitely larger than Z_{output}. To address this concern, the addition of a unity-gain op-amp may be added between the voltage signal and the data measurement unit, shown in Figure 86.

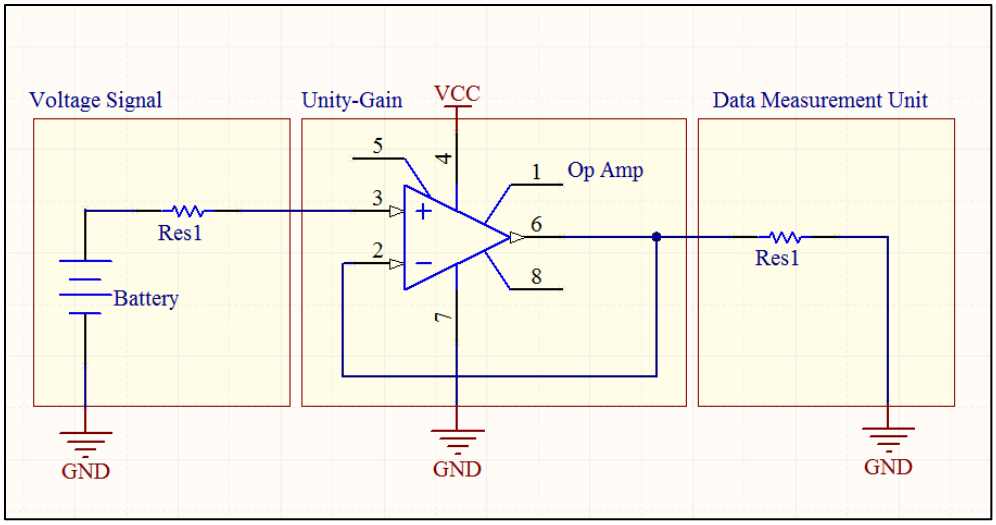


Figure 86: Added Unity-Gain stage for Voltage Signal

The ideal op-amp has infinite input impedance and zero output impedance. This would result in the output impedance in Figure 85 to appear as 0Ω, which results in 100% of the voltage signal reaching the data measurement unit, according to Equation 7. The final implementation of this module would be similar to Figure 87.

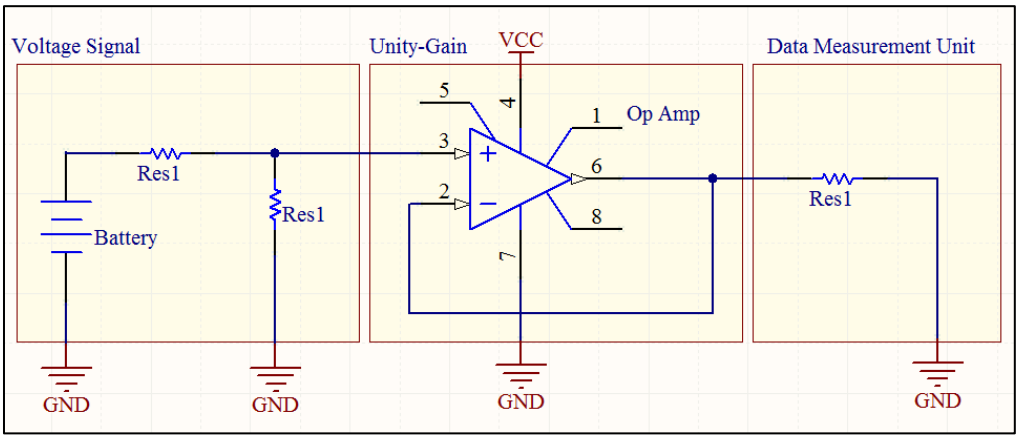


Figure 87: Full implementation of Voltage Sensing Module

7.3.2.4 Temperature Measurement Unit

The temperature measurement unit measures the temperature inside the gondola, and then relays the data to the data processing unit. This unit contains only one sub-unit: a temperature sensor module, as shown in Figure 88. The considered temperature sensors for this unit will be discussed in the sections below.

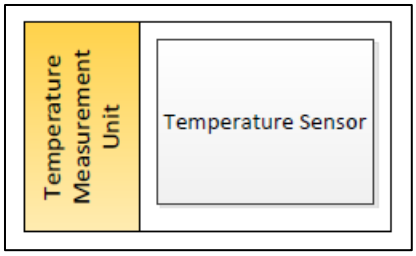


Figure 88: Temperature Measurement Unit

7.3.2.4.1 Thermistors

A thermistor is a type of resistor whose resistance varies significantly with temperature. The voltage across a thermistor can vary because of its change in resistance at different temperatures. By measuring this voltage the temperature around the resistor can be determined. A possible implementation involves using a thermistor with a fixed-value resistor to create a voltage divider network, shown in Figure 89.

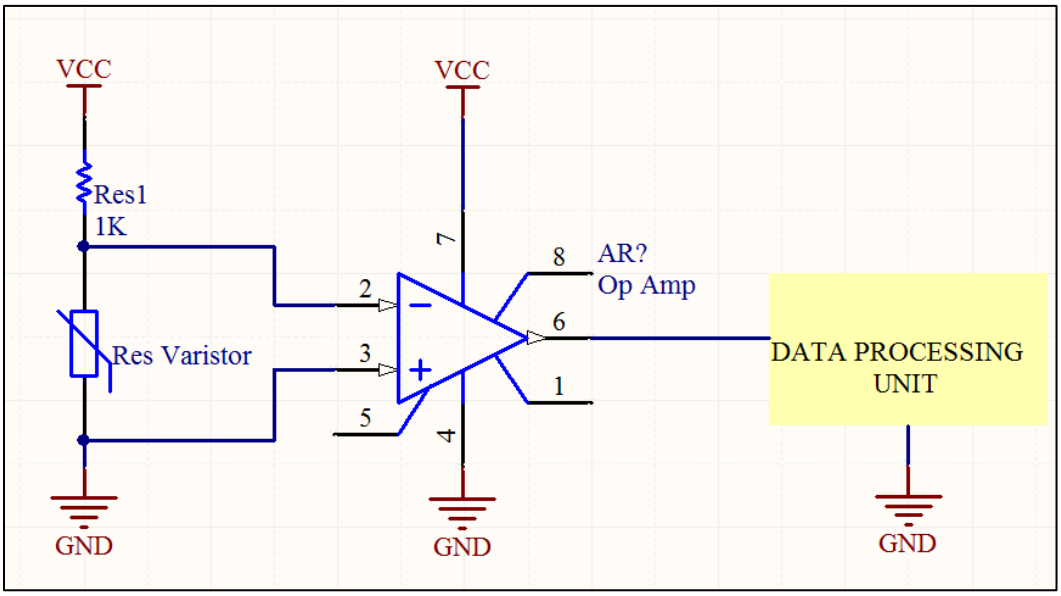


Figure 89: Thermistor Implementation

The voltage divider can be calibrated to produce a known voltage level at a specific temperature. As the temperature varies the voltage drop across the thermistor changes; the voltage is then read by the data processing unit. It is favorable to use an instrumentational op-amp to buffer the voltage signal between the thermistor and the data processing unit, because the op-amp will eliminate noise on the voltage signal.

7.3.2.4.2 Temperature Sensor IC

Temperature sensor ICs produce a voltage corresponding to the measured temperature in its vicinity. These ICs usually contain signal conditioning circuitry and require no calibration. An implementation of the IC is shown in Figure 90.

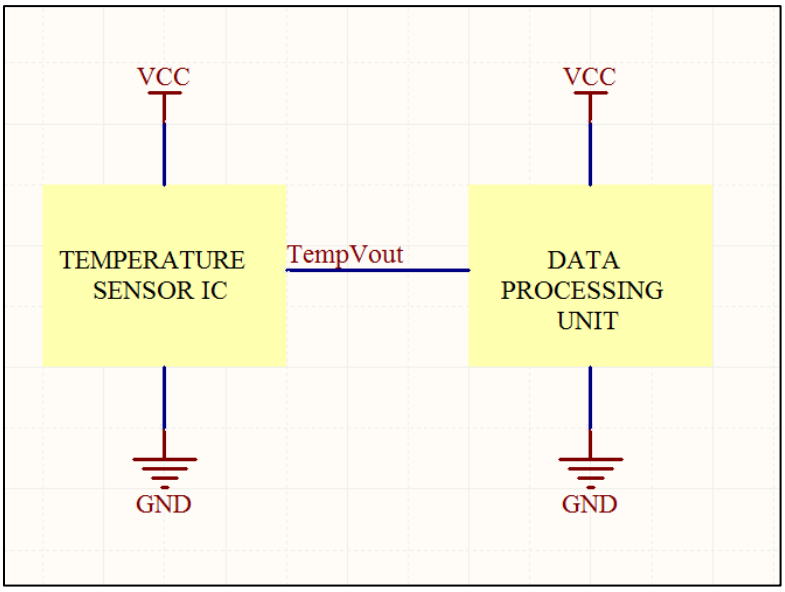


Figure 90: Temperature IC Implementation

The voltage output from the IC is read by the data processing unit, which converts the voltage reading into a temperature.

7.3.2.5 Data Processing Unit

The data processing unit reads in the data from the data measurement unit and the temperature measurement unit, and then calculates the power and energy drawn by the motor drive system. This unit contains two sub-units: A Secure Digital (SD) Card and a microcontroller, as shown in Figure 91.

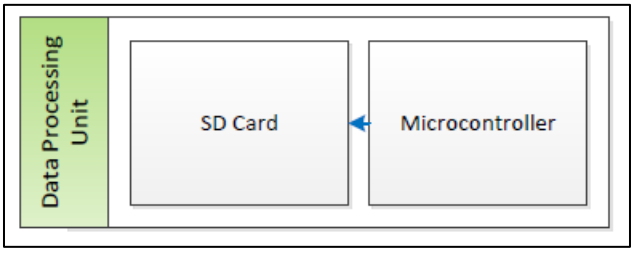


Figure 91: Data Processing Unit

The microcontroller sub-unit processes the data received from the data measurement unit and stores the energy information in volatile and/or non-volatile memory. The SD card sub-unit serves as an alternate non-volatile storage medium for the energy information.

7.3.2.5.1 Microcontroller

The selected microcontroller needed to have three capabilities to be able to calculate the energy usage. These capabilities included: general purpose input/output (GPIO) ports, analog-to-digital converters (ADCs), and timers.

General Purpose Input/Output Ports:

The microcontroller needed a minimum of nine input ports to receive the data from the data measurement unit. The breakdown for this need is shown in Table 18.

Input Ports	
Voltage Sensors	4
Current Sensors	4
Temperature Sensor	1
Total	**9**

Table 18: Input Ports Requirements

Analog-Digital Converter:

The microcontroller needed an ADC with at least nine channels and 10-bits of resolution. Each input port needed a unique channel to provide the data to the microcontroller. Additionally, the 10-bits of resolution specification was based on the resolution needed for the voltage and current sensors, 0.1V and 0.1A respectively, as stated in Section 7.2.2.

The resolution needed for an ADC is calculated by Equation 8, where the "# of codes" is minimum number of binary codes needed to represent the data.

$$ADC_{resoultion} = log_2(\#\ of\ codes)$$

Equation 8: ADC resolution

The minimum number of binary codes needed is calculated by dividing the reference voltage level of the ADC by the voltage level that maps to the required resolution of the measured data, as shown in Equation 9.

$$\#\ of\ codes = \frac{V_{ref}}{V_{resolution}}$$

Equation 9: Binary code calculation

Timers:

The microcontroller needed a minimum of two timers. One timer would provide a clock signal to the data display unit, and the second timer would be used to track the elapsed time since the system has been on.

7.3.2.6 Data Display Unit

The data display unit conveys the information calculated by the data processing unit and the information about the batteries' charge levels. This unit contains two sub-units: A battery level indicator and power system status module, as shown in Figure 92.

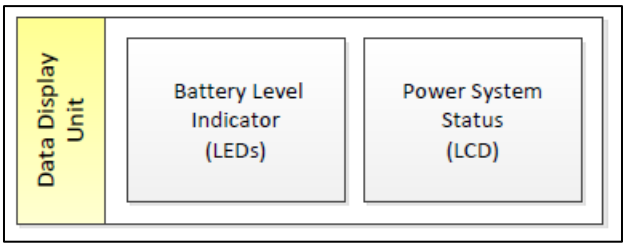


Figure 92: Data Display Unit

7.3.2.6.1 Power System Status

This sub-unit needed to display the following information to the user:

- Battery Voltages
- Current Drawn (per motor)
- Temperature inside the gondola
- Total Energy used by the motor drive system

Our team performed research to determine the most feasible method to display this information and it was narrowed down to two options: Liquid Crystal Display (LCD) or eight segment displays.

7.3.2.6.2 Battery Level Indicator

This module needed to provide a quick and intuitive way for the user to assess the amount of charge left in each battery. After researching viable methods of achieving this, our team narrowed down the choices between using a custom light emitting diode (LED) solution and purchasing a pre-made component.

7.4 Final System Design

This section will discuss the design decisions we made for each sub-unit and module discussed in Section 7.3.2, in addition to the rationale behind each design choice. The design choices are summarized in Table 19.

Section	Sub-unit/Module	Design Choice
7.3.2.1.1	Batteries	2011 AY team's battery choice
7.3.2.1.2	Discharge Control	Diode ORing using an Ideal Diode integrated circuitry
7.3.2.2	Voltage Regulation Unit	Switching Regulation
7.3.2.2	Current Sensing	Hall Effect Sensor ICs
7.3.2.3.2	Voltage Sensing	Voltage Divider with Op-Amp
7.3.2.4	Temperature Measurement Unit	Temperature Sensor IC
7.3.2.5.1	Microcontroller	Microcontroller with built-in ADC and timers
7.3.2.6.1	Power System Status	LCD Screen
7.3.2.6.2	Battery Level Indicator	Pre-made component

Table 19: Final System Design Choices

After evaluating the 2011 AY battery choice and researching suitable replacements, our team decided to continue using the batteries from the 2011 AY team because their choice had the lowest cost per capacity with respect to weight.

For the discharge control we decided to use an ideal diode IC because it had a voltage drop from the batteries to the main power bus less than 0.2V, which met the system requirements.

We decided to use switching regulator for the regulation unit because the potentially wasted power associated with the linear regulators would be 42W and 26W, for the +5V and +3.3V units respectively.

For the current sensor module we decided to hall-effect sensor IC because of the ease of implementation, built-in signal conditioning, and temperature correction.

We decided to use a voltage divider with a unity gain op-amp for the voltage sensing module. We used the op-amp because the low output impedance ensures maximum transfer of the voltage signal to GPIO pins of whichever microcontroller was selected.

For the temperature sensor unit we decided to use a temperature sensor IC because IC because of the ease of implementation, built-in signal conditioning, and temperature correction.

We selected a microcontroller with an integrated ADC and timer because of the ease of implementation, and minimization of additional peripherals in order to achieve the same functionality.

For the power system status, we chose a LCD screen because of its ability to display multiple energy information at once. Also it would provide a more intuitive display for the end user in comparison to its alternative design choice.

Lastly, we decided to use a pre-made component for the battery level indicator because of its ease of implementation, and reliability.

7.5 Detailed System Design

This section discusses the specific implementation of the sub-units and modules mentioned in the previous sections. Table 20 summarizes the components used for the implementations.

Section	Sub-unit/Module	Implementation	
		Company	Part Number
7.3.2.1.1	Batteries	Zippy	Flightmax 5000mAh 3S1P 20C
7.3.2.1.2	Discharge Control	Linear Technologies	LTC4353
7.3.2.2	Voltage Regulation Unit	Analog Devices	ADP2381; ADP2303
7.3.2.2	Current Sensing	Allegro Microsystems	ACS711
7.3.2.3.2	Voltage Sensing	Texas Instruments	TLV2374IPWR
7.3.2.4	Temperature Measurement Unit	Texas Instruments	TMP20AIDCKR
7.3.2.5.1	Microcontroller	Texas Instruments	LM4F232H5QDFIGA1
7.3.2.6.1	Power System Status	Crystalfontz	CFAL9664B-F-B1
7.3.2.6.2	Battery Level Indicator	Hobby King	2-6S LED balance Voltage indicator

Table 20: Final System Design Implementations

The details for the implementation of each sub-unit are shown in their respective schematics, which can be found in Appendix M, Appendix N, Appendix O.

7.6 Testing and Results

We performed tests to determine if power system met its requirements and performed as expected. Specific sub-units and modules were tested to confirm functionality, then the combined functionalities of the sub-units and modules were integrated to verify that the system performed as expected. Table 21 summarizes the tests and results for each requirement.

ID	Requirement	Need	Test	Result
PS-01	The power system shall supply the blimp with energy for a minimum of 90 minutes when the left and right motors are operating at 50% of maximum power.	The power system needed to allow the blimp to perform endurance flights	Allowed the motor to run continuously at 50% maximum power and monitored the time	Not Met
PS-02	The difference between the power supply's voltage and the voltage measured at the left, right, and tail motors shall be no more than 0.2V.	To allow for maximum voltage across the motors, hence maximize the speed of the motors	Measured the voltage level at the batteries and at the motor bus, then calculated the difference	Met
PS-03	The power system shall be able to measure the battery voltages between 0-13V ± 0.1V.	Needed for energy calculations	Measured the voltage level at the batteries and confirmed value with external equipment	Met
PS-04	The power system shall measure the current individually drawn by the left, right, and tail motors between 0-16A ± 0.1A.	Needed for energy calculations	Measured the current drawn by the motors and confirmed value with external equipment	Met
PS-05	The power system shall calculate the average current draw of the left, right, and tail motors accurately to 0.1A throughout the blimp's flight time using logged values.	Needed for energy calculations	Not performed	Not Met
PS-06	The power system shall calculate the instantaneous power used by the left, right, and tail motors with a minimum resolution of 0.1 W.	Needed for energy calculations	Not performed	Not Met
PS-07	The power system shall calculate energy used by the blimp by summing the instantaneous power over the flight time with a resolution of 0.1 Watt-Hours, using logged values.	Main purpose of the power system	Not performed	Not Met
PS-08	The power system shall measure time with a minimum resolution of 1s.	Needed for energy calculations	Measured the elapsed time confirmed value with external equipment	Met

PS-09	The power system shall measure the temperature inside the gondola accurately to 0.1°C.	Needed to assess the conditions around the batteries	Measured the temperature and confirmed value with external equipment	Met
PS-10	The power system shall record the measured and calculated voltage, current, power and energy values into a non-volatile memory.	Needed to log the energy data	Not performed	Not Met
PS-11	The power system shall be able to display the measured and calculated voltage, current, power and energy values.	Needed to interface with the user	Viewed the LCD to confirm the information was being displayed	Met
PS-12	The final design for the power system shall be implemented on a printed circuit board(s) (PCB).	Needed to provide an intuitive and reliable implementation of the power system	Ordered a manufactured PCB	Met

Table 21: Requirements Fulfillment Table for Power System

Requirements PS-05, PS-06, PS-07, and PS-10 all related directly to the logging capability of the power system, and were not met. We intended to implement this logging functionality in the code for the MCU. We used an example project written by Texas Instruments for this specific MCU and modified the code to fit our needs. We decided to use this example project because it provided a template adequate for our needs. The original Texas Instruments project, called `qs-logger`, contained a data logging functionality to save data read by the evaluation board sensors. The project was designed to save the data to a choice of a USB, internal flash memory, SD card, or a host PC. Unfortunately, we were unable to successfully use this functionality in either our modified project or the original Texas Instruments project. Each attempt resulted in a 'USB Error' message on the LCD screen. We could not find a solution to this problem through our research. Due to time constraints we were unable to implement the logging functionality.

We implemented the power system design on a PCB as stated in PS-12. However, there was one mistake in this PCB design. This mistake was that resistor R10 in the 3.3V regulation sub-section did not have a connection to ground. We manually fixed this mistake by soldering a wire from one pad of this resistor directly to a ground connection. This mistake has been fixed in an updated version of the PCB design, but this PCB has not been printed.

7.7 Summary

In this chapter, we discussed the proposed design, possible implementations, and final implementation of the power system. We discussed the possible system design, the needs and requirements for the system, and the final system design. We conducted trade studies comparing different implementations of the system sub-units and modules, and discussed the final implementation choices. Lastly, we discussed the testing and results of the sub-units, modules, and the overall power system.

8 Mission Module

8.1 Introduction

The camera mission module must provide users at the ground station with a live video feed from the blimp and record photos and videos while the blimp is in flight. The camera is housed in a fixed camera mount inside the mission module compartment, making it easy to exchange as needed with other mission modules. Below we discuss our method for designing and implementing the camera mission module and our results.

8.2 System Design

The camera mission module is designed for aerial photography. With a camera mounted in the gondola, the team is able to record photos and videos from the air while the blimp is in flight. The camera must be securely attached to the gondola and mounted at a fixed angle for consistency throughout the photos and videos. Users on the ground can communicate wirelessly with the camera to choose when to capture photos and record videos. The camera also provides a live video feed to the base station. Figure 93 shows the system design for the camera mission module.

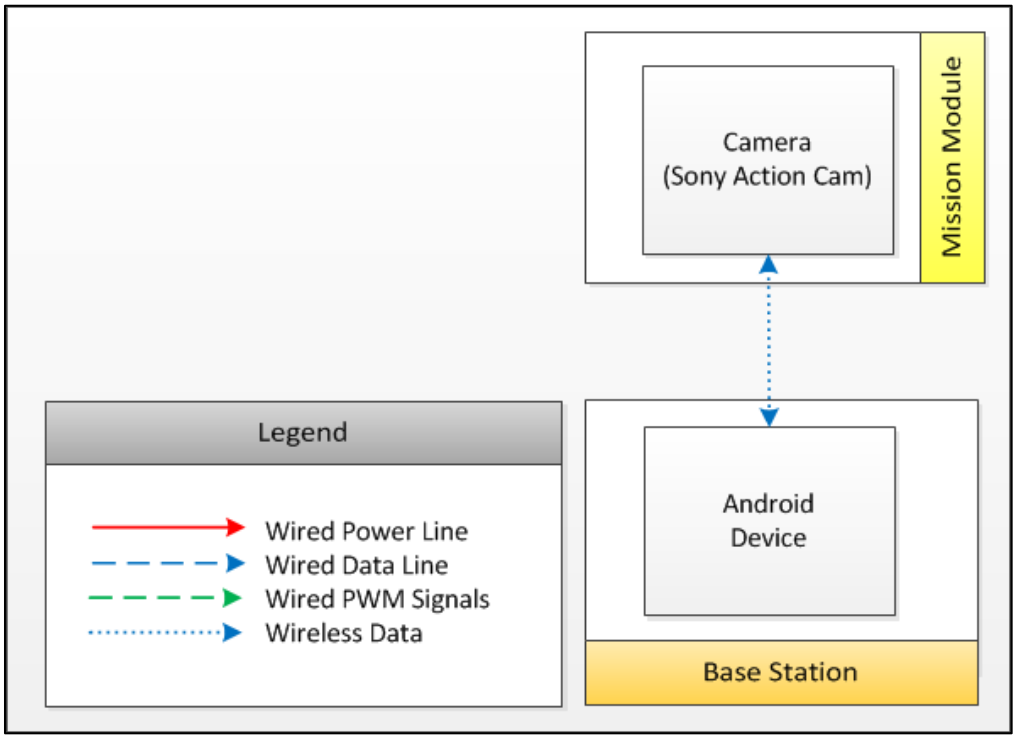


Figure 93: Mission Module System Diagram

8.2.1 System Needs

The camera needs wireless capability to communicate with the base station, sufficient resolution to take quality photos and videos, low-weight to comply with the blimp's limited lifting capacity, and external memory storage to store files on-board the camera during flight.

The camera mount securely holds the camera in place and protects it from possible damage during takeoff, flight, and landing. The camera mount must fit inside – and be attached to – the mission module compartment, yet be removed from the gondola if needed. Like everything in the gondola, the camera mount must be lightweight and durable.

The base station is the point of communication with the camera mission module. The ground station device must communicate with the camera, receive a live video feed, and send commands to the camera to tell it when to capture photos or record videos. To meet these needs, the base station device needs wireless capability and software that would be compatible with the chosen camera.

8.2.2 System Requirements

Table 22 lists the specific requirements for the camera including weight, size, image quality, price, zoom capability, and wireless capability and range. The camera must weigh no more than 794 grams (1.75 lbs.). The entire mission module is allocated a maximum weight of 907 grams (2 lbs.), and we also needed to account for the possible weight of other components and the camera mount. The camera's image resolution must be at least 1280x720 pixels to provide high quality photos. The camera must have a zoom capability of 3x or greater in either optical or digital zoom. Finally, the camera must have wireless capability compatible with the base station. Originally the team required a minimum wireless range of 125 meters (410 feet). This number was determined using the diagonal length of a football field (120 meters or 394 feet) and a maximum blimp altitude of 30.5 meters (100 feet). If the blimp were on one corner of the football field at its maximum altitude, and the ground station were at the opposite corner, they would be 123.8 meters apart. We rounded the required range up to 125 meters. Later we determined that this range requirement was unnecessarily large; the team would never be 125 meters from the blimp while it was flying. We decided to change the requirement to half of the diagonal length of the football field (60 meters or 197 feet) and a maximum blimp altitude of 12 meters (40 feet). This results in a maximum range requirement of 61 meters (200 feet).

Mission Module Requirements	
MM-01	Mission module (excluding foam compartment) must weigh less than 907 grams
MM-02	Mission module must be independent from gondola (no wired connections to other compartments in the gondola)
MM-03	Mission module must have a minimum wireless range of 61 meters
MM-04	Camera must weigh less than 794 grams
MM-05	Camera must have a resolution of 1280x720 pixels (px) or greater
MM-06	Camera must have zoom capability of 3x or greater, either optical or digital zoom

Table 22: Mission Module Requirements

The camera mount must fit within the mission module compartment, be lightweight and durable, and be able to be replicated. The floor of the mission module compartment space is a 9 cm by 15 cm rectangle, and the compartment becomes larger above the floor, as shown in Figure 94. We designed the camera mount from foam core to be lightweight and durable, like the body of the gondola. To make the camera mount easy to replicate, all pieces were designed in SolidWorks[9] and cut out with a laser cutter, and then we developed assembly instructions for the constructed camera mount.

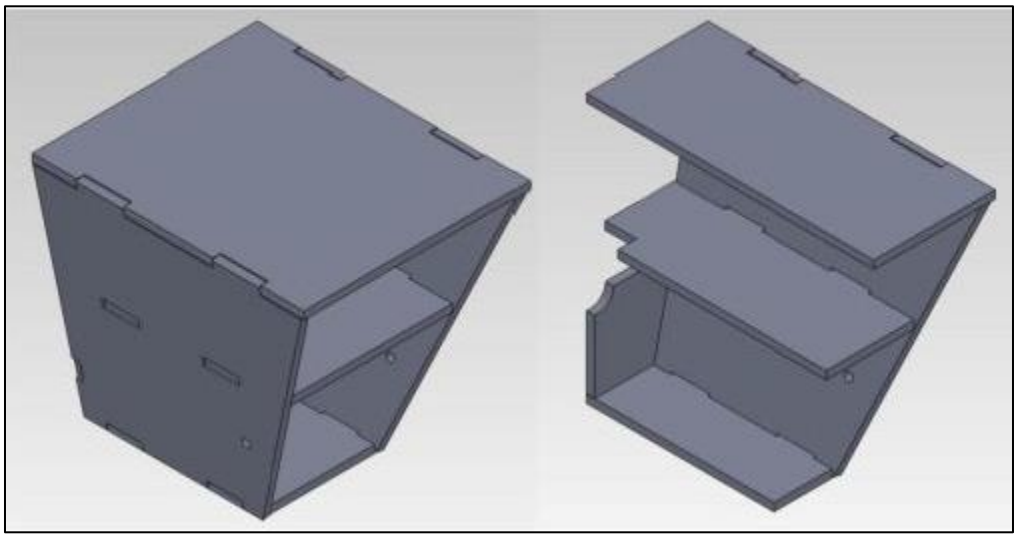


Figure 94: CAD drawing of mission module with cut-away [1]

The base station needed an electronic device such as a smartphone, laptop, or tablet with compatible software and wireless capability to communicate with the camera. If the wireless range of the camera module needed to be increased, the base station would also need an antenna.

[9] http://www.solidworks.com/

8.3 System Architecture and Trade Studies

In this section we discuss the possible options for the mission module camera and the fixed camera mount, and how we selected the ones most suited for our needs.

8.3.1 Camera Research and Selection

As a first step to designing the camera mission module, we had to choose a suitable camera. We determined that after a suitable camera was found, we could design the rest of the module around the camera. We researched a range of digital cameras and narrowed the options to four possible cameras, shown in Table 23, which records specifications for weight, size, maximum still image resolution, cost, wireless capability, zoom capability, and image stabilization. Because all four cameras could record videos in full 1080p HD, this information is not included in the table.

Camera specifications table				
Category:	GoPro HD Hero2 [14]	Samsung HMX-U20 [17]	PICSIO GC-FM1A [18]	Sony Action Cam w/Wi-Fi [19]
Weight	771 grams	111 grams	90.7 grams	65 grams
Dimensions (L x W x H)	24.3x9.9x9.9 cm	10.4x5.3x1.5 cm	9.7x1.5x 5.3 cm	8.0x2.46x4.7 cm
Maximum Still Image Resolution (pixel height)	3498 px	3648 px	3264 px	1600 px
Cost	$334.95	$131.45	$124.19	$269.99
Wireless capability	None[10]	None	None	IEEE 802.11 b/g/n (2.4GHz band)
Zoom capability	None	3x optical zoom	4x digital zoom	3x digital zoom[11]
Image stabilization	No	Yes	Yes	Yes

Table 23: Specifications for camera choices

We then created a comparison table to select the camera most suitable to our needs. We chose the categories to include in the table and assigned each category a weight according to its importance. Each category is divided into appropriate possible ratings, as seen in Table 24; note

[10] It should be noted that the GoPro HD Hero2 can have wireless capability with the purchase of an expansion pack that costs approximately $99.99 from GoPro [20].
[11] The zoom capability of the Sony Action Cam w/Wi-Fi is not listed on the website, but was instead determined through a phone call with Sony representatives.

that resolution is rated according to the maximum possible pixel height of still images. In our rating system, a higher value represents more desirable characteristics.

Rating system for value analysis						
Category	Wireless	Zoom	Price	Resolution	Weight	Image stabilization
Weight of category	35%	20%	15%	10%	10%	10%
Rating scale	Out of 3	Out of 5	Out of 5	Out of 5	Out of 5	Out of 2
Rating - 1	No wireless	No zoom	≥$400	240px or less	≥544 grams	No
Rating - 2	802.11a/b/g	Digital zoom ≤ 4x	$300-$399	241px-360px	409-544 grams	Yes
Rating - 3	802.11a/b/g/n	Digital zoom > 4x	$200-$299	361px-480px	273-408 grams	N/A
Rating - 4	N/A	Optical zoom ≤ 4x	$100-$199	481px-720px	137-272 grams	N/A
Rating - 5	N/A	Optical zoom > 4x	≤$100	>720px	≤136 grams	N/A

Table 24: Rating system for value analysis

After recording the specifications of each camera and determining the categories to compare, we created a detailed comparison table in order to select the camera most suitable to our needs, as shown in Table 25. According to our rating scale, we would select the camera with the highest total score; this was the Sony Action Cam with Wi-Fi, shown in Figure 95.

Figure 95: Sony Action Cam with Wi-Fi [21]

Comparison chart for cameras													
Category	Wireless		Zoom		Price		Resolution		Weight		Image stabilization		Total Score
Rating	Out of 3	35%	Out of 5	20%	Out of 5	15%	Out of 5	10%	Out of 5	10%	Out of 2	10%	Out of 4
Camera													
GoPro HD Hero2	1	0.35	1	0.2	2	0.3	5	0.5	1	0.1	1	0.1	1.55
Samsung HMX-U20	1	0.35	4	0.8	4	0.6	5	0.5	5	0.5	2	0.2	2.95
PICSIO GC-FM1A	1	0.35	2	0.4	4	0.6	5	0.5	5	0.5	2	0.2	2.55
Sony Action Cam w/Wi-Fi	3	1.05	2	0.4	3	0.45	5	0.5	5	0.5	2	0.2	3.1

Table 25: Comparison chart for cameras

8.3.2 Fixed Camera Mount Design

All designs for the fixed camera mount were created from foam core to be lightweight, and all designs were made with the camera inside its waterproof case and held at a fixed 45 degree down angle. The camera is shown inside its waterproof case in Figure 96.

Figure 96: Sony Action Cam w/Wi-Fi and Waterproof Case [22]

In our first design, the camera was held upside-down inside a fixed camera mount. There were parallelogram-shaped walls on either side of the camera, and the camera only extended outside of the mission module compartment by a small amount. All components were glued together for stability, with the exception of the square of foam core attached to the waterproof camera case. This piece was removable with Velcro to allow the camera to be easily removed from the fixed mount.

In our second design, we changed the orientation of the camera so that it was no longer upside-down. The parallelogram sides from the first design made it difficult to access the camera and were removed. A cut-out was made in the bottom of the mission module compartment to hold the camera mount. Since the camera was now right-side-up, it extended farther outside the bottom of the mission module compartment. The rectangular piece of foam core permanently attached to the waterproof case would be glued to the rest of the fixed camera mount. To remove the camera it would have been necessary to first remove the camera from its waterproof case.

In our third design, we tried to improve the accessibility of the camera and security of the camera mount. We designed notches in the rectangle of foam core attached to the camera's waterproof case to keep the camera firmly in place. The cut-out in the bottom of the mission module compartment was made smaller to more securely accommodate the camera mount. A curved cutout was added in the mission module compartment floor for the front of the camera to rest against. This added a bit more security for the camera; if the waterproof camera mount became detached from the foam core, this cutout would prevent the camera from falling out of the gondola. Finally, the rectangular piece of foam core attached to the waterproof case was secured in place on the top and bottom with Velcro. To remove the camera from the fixed camera mount, one need only remove this foam piece.

In our fourth and final design, we tried to improve the durability of the curved cutout that the camera rests against. When this cutout was laser cut, the cuts were close enough together to completely melt the foam in between the layers of paper. This allowed the camera to tear through the two layers of paper, defeating the purpose of the cutout. In order the make the camera mount more secure, the curved cutout was cut out by hand. This left a solid piece of foam core for the camera to rest against.

8.4 Final System Design

We chose the Sony Action Cam with Wi-Fi because it met all of our criteria for size, weight, image quality, image stabilization, and wireless capability. This camera only has digital zoom, but we decided that this was adequate.

The ground station device can be any Android smartphone or tablet that has software compatible with the Android app that is used to control the camera, PlayMemories Mobile[12]. The app is used to stream a live video feed from the camera to the Android device and remotely

[12]https://play.google.com/store/apps/details?id=com.sony.playmemories.mobile&hl=en

control the camera's photo capture and video recording functions. This ability will be discussed in more detail in Section 8.6.

The final fixed camera mount holds the camera – inside its waterproof case – at a fixed 45 degree angle in the mount attached to the mission module compartment. The waterproof case can be removed from the camera mount and the camera can then be removed from its case for charging or downloading the files from the microSD card. The final camera mount design is shown in Figure 97.

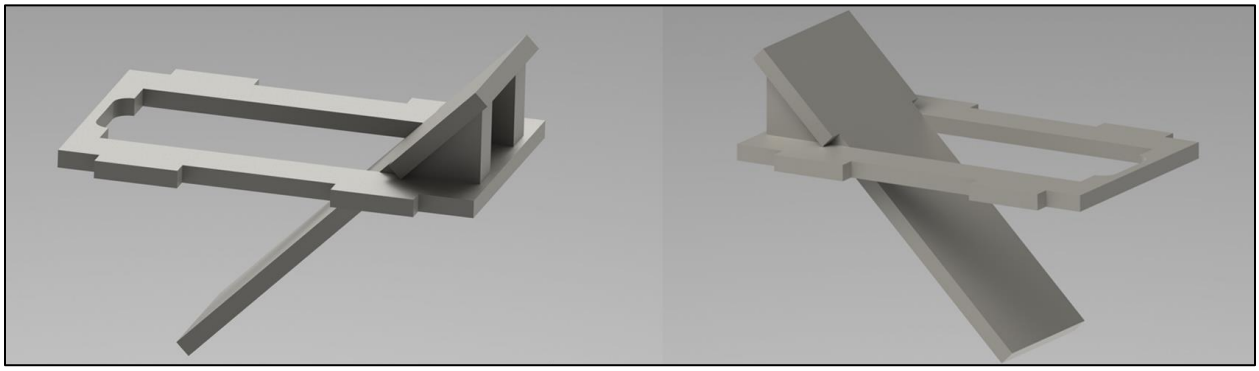


Figure 97: SolidWorks Image of Final Camera Mount Design

8.5 Detailed System Design

The final design for the camera mission module combined the Sony Action Cam with Wi-Fi, a fixed camera mount, and a base station Android device. The camera is kept inside its waterproof camera case while in use, and this case is permanently affixed to the fixed camera mount. To remove the camera, the entire piece of foam core is removed so that the waterproof case is accessible. The base station device can be any Android smartphone or tablet that is compatible with the Android app included with the camera, PlayMemories Mobile. When the camera is in the fixed camera mount, the base station device can be used to remotely control the camera to take photos and videos and a live video feed is streamed from the camera to the base station device. All photos and videos are saved on the microSD card inside the camera.

8.6 Testing and Results

In this section we detail all of the tests we performed with the camera mission module and discuss the results of each test. We also tested the mission module to determine which requirements were met and compiled these results in Section 8.6.5.

8.6.1 Documentation of Camera Settings

To test the camera, we first documented all of the available camera settings and their functions. We created three separate flow charts – one detailing all of the camera settings, one detailing the settings that are possible to change wirelessly, and another detailing the default settings. These three flowcharts are included in Appendix P, Appendix Q, and Appendix R. This process made us familiar with the camera's functions and helped to prepare us for future tests.

8.6.2 Compatibility with Sony App

Next we tested the Sony app PlayMemories Mobile, which was designed to wirelessly control the camera with an Android device. We used different Android devices with different software installations in order to determine compatibility between the devices and the camera using the app. The devices we tested and their compatibility are detailed in Table 26. We discovered that only certain devices were compatible with the Sony app. For the other camera tests we used only compatible Android devices.

Type of Device	Brand and Model	Operating System	Compatible with app?
Smartphone	Samsung Galaxy S2	Android 4.0.4 (Ice Cream Sandwich)	No
Smartphone	Samsung Galaxy S3	Android 4.0.4 (Ice Cream Sandwich)	Yes
Smartphone	Samsung Galaxy S3	Android 4.1.2 (Jelly Bean)	Yes
Smartphone	Motorola Atrix 4G	Android 2.3.6 (Gingerbread)	Yes
Smartphone	Motorola Droid RAZR MAXX	Android 4.0.4 (Ice Cream Sandwich)	No
Tablet	ASUS Transformer Prime TF201	Android 4.1.1 (Jelly Bean)	Yes

Table 26: Compatibility of Android devices with PlayMemories Mobile app

8.6.3 Wireless Range Testing

In order to determine the maximum range where the camera can be controlled wirelessly with an Android device, we performed multiple tests on the football field. We maintained line-of-sight between the Android device and the camera at all times, and both devices were fully charged before beginning the tests. For the wireless range tests we used a Samsung Galaxy S3 running Android 4.1.2 (Jelly Bean). We tested the wireless range with the camera inside and

outside the waterproof case, while taking photos, while recording videos, and while only streaming the live video feed. Each of these factors could possibly affect the wireless range. The results of these tests are detailed in Table 27. All distances were originally recorded in yards and then converted to meters.

Inside/Outside Waterproof Case	Recording Mode	Wireless Range Live Video Feed Quality Deteriorates	Wireless Range Commands can be Sent to Camera	Wireless Range Signal is Lost Completely
Outside	Not recording	27.4 meters (89.9 feet)	82.3 meters (270 feet)	91.4 meters (299.9 feet)
Outside	Recording video	18.3 meters (60 feet)	45.7 meters (149.9 feet)	82.3 meters (270 feet)
Outside	Capturing photos	22.9 meters (75.1 feet)	54.9 meters (180.1 feet)	82.3 meters (270 feet)

Table 27: Wireless Range Test Results for Mission Module

As shown in Table 27, these wireless range tests were performed with the camera outside of its waterproof case. We also attempted to test the wireless range with the camera inside the waterproof case, but could not accurately determine a consistent maximum wireless range. In some tests the wireless connection could be sustained for up to 3 meters (10 feet), but in other tests the wireless connection could not be established at all. The wireless connection with the camera inside its waterproof case proved to be unreliable.

8.6.4 Camera Mount

The final version of the camera mount was designed in SolidWorks. The pieces were either laser-cut or hand-cut as appropriate, and assembled using epoxy, hot glue, and Velcro. The final camera mount was durable and lightweight and held the camera firmly in place at a 45 degree downward angle. The camera mount was permanently attached to the mission module compartment, which can be removed from the gondola as a whole. The final camera mount is shown in Figure 98.

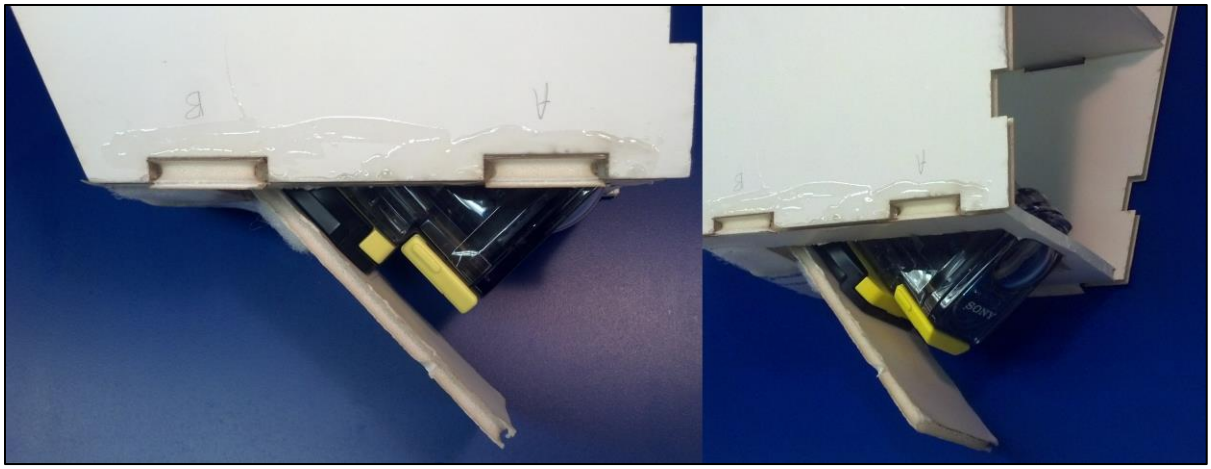


Figure 98: Photo of Final Camera Mount Design

8.6.5 Mission Module Requirements Testing

The updated requirements for the mission module, as detailed in Section 8.2.2, were tested in order to determine if the final mission module met each requirement. Table 28 shows each requirement, how it was tested, and if the mission module met the requirement. The results are explained more after the table.

ID	Requirement	Need	Test	Result
MM-01	Mission module (excluding foam compartment) must weigh less than 907 grams.	Maximum weight allotment for mission module is 907 grams.	Weighed the camera, waterproof case, and camera mount together.	Met, 218g
MM-02	Mission module must be independent from gondola (no wired connections to other compartments in the gondola).	Allows mission module to be easily switched out for other modules.	Checked mission module compartment for wires leading to other sections.	Met
MM-03	Mission module must have a minimum wireless range of 61 meters.	Maximum required range on football field.	Tested wireless range on football field.	Not Met (see below)
MM-04	Camera must weigh less than 794 grams.	Allows for weight of other components in mission module.	Weighed the camera.	Met, 86g
MM-05	Camera must have a resolution of 1280x720 pixels (px) or greater.	Needed for high resolution photos.	Checked specifications sheet for camera.	Met
MM-06	Camera must have zoom capability of 3x or greater, either optical or digital zoom.	Needed for photos and videos during flight.	Called Sony representatives to determine zoom capability.	Met

Table 28: Requirements Fulfillment Table for Camera Mission Module

The wireless range requirement was not met with the camera inside the waterproof case, but was met with the camera outside of it. The camera mount was designed with the camera inside of its waterproof case for extra protection, but this severely limited the range of the mission module. If another camera mount was designed to hold the camera outside of the waterproof case the wireless range would have met requirement MM-03, as explained in Section 8.6.3.

8.7 Summary

In this chapter, we discussed the design and implementation of the camera mission module. We discussed the system design, the needs and requirements for the module, and the final system architecture. We conducted trade studies comparing different possible cameras, chose a camera appropriate for our needs, and designed the mission module around this camera. We tested all components of the camera mission module and recorded the results. Finally, we determined which requirements the mission module met, and which requirements were not met.

9 Results and Recommendations

This chapter discusses the overall results of our project and our recommendations for future project teams.

9.1 Overall Project Results

Through ground and flight tests we confirmed that we successfully implemented non-optimized autonomous flight, portions of the power system design, and most of the camera mission module requirements. A CAD representation of our final gondola can be seen below in Figure 99.

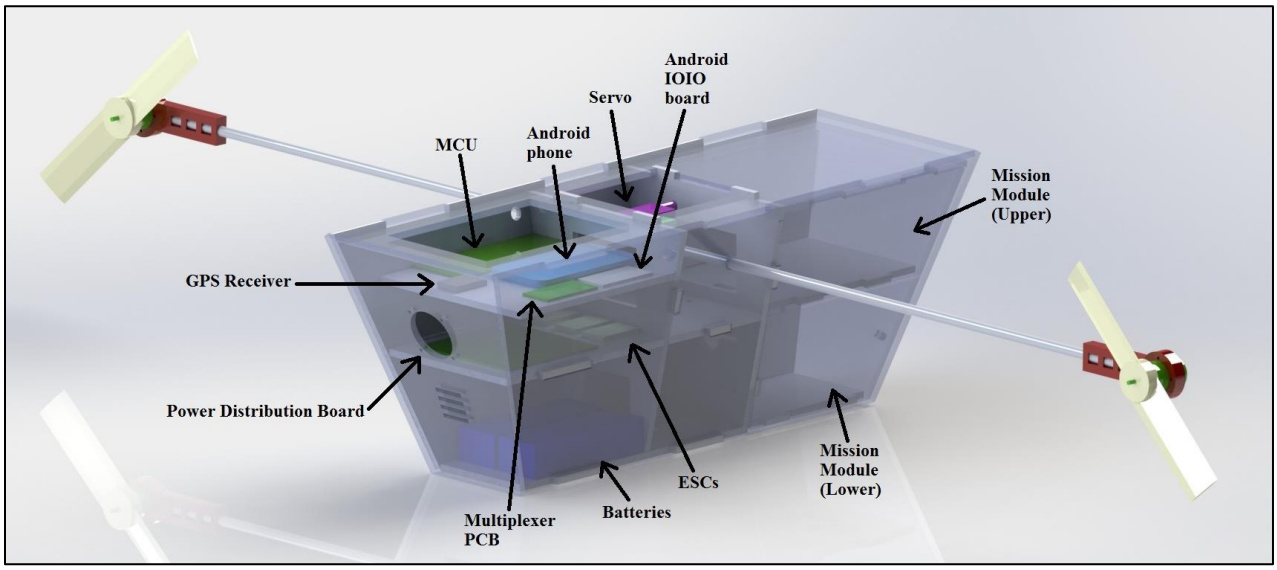


Figure 99: Final Gondola

We were unable to successfully complete certain requirements for the power system and camera mission module. The following section discusses these requirements in more detail.

9.1.1 Summary of Fulfillment of Project Requirements

In this section we discuss our detailed project requirements, how each requirement was tested, and whether each requirement was met. After discussing our project requirements, we summarize the results of our project.

ID	Requirement	Need	Test	Result
CS-01	The control system components in total must weigh no more than 907 g (2 lbs.).	Maximum weight allotment for control system.	Weighed the Samsung Galaxy S3, Android IOIO, RC Receiver, Multiplexer PCB, and jumper wires.	Met, 586g
CS-02	The control system must have a GPS receiver capable of determining the blimp's location accurately within 4 meters.	To find the GPS location of the blimp to use for the autonomous algorithm.	Researched the datasheet specifications for the Samsung Galaxy S3 to verify GPS receiver and its accuracy.	Met
CS-03	The control system must be capable of determining the blimp's altitude accurately within 1 meter.	To determine the blimp's altitude for the autonomous algorithm.	Researched the datasheet specifications for the Samsung Galaxy S3 to verify barometer sensor and used this sensor reading to determine a resolution for altitude height.	Met
CS-04	The control system must have an inertial measurement unit (IMU) with a minimum accuracy of +/- 250-2000 degree/sec.	To meet sensor accuracy for the 2011 AY team's component, and to utilize sensor readings for the autonomous algorithm.	Researched the datasheet specifications for the Samsung Galaxy S3 to verify IMU sensor and its accuracy.	Met
CS-05	The control system must have a processor with a minimum speed of 720MHz.	To meet processor speed for the 2011 AY team's processor, and to process sensor readings in real time for the autonomous algorithm.	Researched the datasheet specifications for the Samsung Galaxy S3 to verify processor had a minimum speed of 720MHz.	Met
CS-06	The control system must have a hardware component capable of generating a minimum of 4 PWM signals simultaneously based on standard RC aircraft PWM signal specifications.	The four simultaneous PWM signals must be generated to control the four drive components: Servo, Tail Motor, Left Motor, and Right Motor.	Connected the Android IOIO I/O pins 3, 5, 7 and 10 to an oscilloscope to verify if the IOIO was capable of simultaneously generating four PWM signals to control the drive system.	Met
CS-07	The control system must be able to select which signals are sent to the drive system, either from the RC receiver (manual control), or from the processor device (autonomous control).	The FAA mandates that that all autonomous aircraft have a two way link to ground control that overrides automatic controls for safety reasons. [1]	Constructed and built a multiplexer system, and verified by looking at the output signals of the multiplexer on a four channel oscilloscope to verify whether RC receiver signals (manual control) or the IOIO signals (autonomous control) were sent to the output of the multiplexer based on the manual switch flipped on the RC controller.	Met

Table 29: Results for Control System Requirements

As shown in Table 29, we successfully met all of our Control System requirements.

ID	Requirement	Need	Test	Result
AF-01	The Android device must be able to control the speed of the motors and rotation of the servo.	Control the drive system using Java code within the Android app	Controlled lab test with the Android device, IOIO, motors, and servo to demonstrate control with the Android app	Met
AF-02	The system must update the blimp's current position every second.	The current position is needed to calculate how to get to the destination point	Log every GPS reading from the Android's onboard GPS to plaintext files and examine the timestamps of each reading	Met
AF-03	The system must update the blimp's current altitude at least every second.	The current altitude is needed so the blimp does not fly higher than 100 feet or crash into the ground	Log every barometric pressure reading and altitude calculation to plaintext files and examine the timestamps of each reading	Met
AF-04	The system must update the blimp's current heading at least every second.	The current heading is needed so the blimp closes the distance to its destination point	Log every heading reading to plaintext files and examine the timestamps of each reading	Met
AF-05	The blimp must be able to autonomously navigate between two GPS waypoints.	Project goal	Ground and flight testing	Met

Table 30: Results for Autonomous Flight Requirements

As shown in Table 30, we successfully met all of our Autonomous Flight requirements.

ID	Requirement	Need	Test	Result
PS-01	The power system shall supply the blimp with energy for a minimum of 90 minutes when the left and right motors are operating at 50% of maximum power.	The power system needed to allow the blimp to perform endurance flights	Allowed the motor to run continuously at 50% maximum power and monitored the time	Not Met
PS-02	The difference between the power supply's voltage and the voltage measured at the left, right, and tail motors shall be no more than 0.2V.	To allow for maximum voltage across the motors, hence maximize the speed of the motors	Measured the voltage level at the batteries and at the motor bus, then calculated the difference	Met
PS-03	The power system shall be able to measure the battery voltages between 0-13V ± 0.1V.	Needed for energy calculations	Measured the voltage level at the batteries and confirmed value with external equipment	Met
PS-04	The power system shall measure the current individually drawn by the left, right, and tail motors between 0-16A ± 0.1A.	Needed for energy calculations	Measured the current drawn by the motors and confirmed value with external equipment	Met
PS-05	The power system shall calculate the average current draw of the left, right, and tail motors accurately to 0.1A throughout the blimp's flight time using logged values.	Needed for energy calculations	Not performed	Not Met

PS-06	The power system shall calculate the instantaneous power used by the left, right, and tail motors with a minimum resolution of 0.1 W.	Needed for energy calculations	Not performed	Not Met
PS-07	The power system shall calculate energy used by the blimp by summing the instantaneous power over the flight time with a resolution of 0.1 Watt-Hours, using logged values.	Main purpose of the power system	Not performed	Not Met
PS-08	The power system shall measure time with a minimum resolution of 1s.	Needed for energy calculations	Measured the elapsed time confirmed value with external equipment	Met
PS-09	The power system shall measure the temperature inside the gondola accurately to 0.1°C.	Needed to assess the conditions around the batteries	Measured the temperature and confirmed value with external equipment	Met
PS-10	The power system shall record the measured and calculated voltage, current, power and energy values into a non-volatile memory.	Needed to log the energy data	Not performed	Not Met (see below)
PS-11	The power system shall be able to display the measured and calculated voltage, current, power and energy values.	Needed to interface with the user	Viewed the LCD to confirm the information was being displayed	Met
PS-12	The final design for the power system shall be implemented on a printed circuit board(s) (PCB).	Needed to provide an intuitive and reliable implementation of the power system	Ordered a manufactured PCB	Met

Table 31: Results for Power System Requirements

As shown in Table 31, we were able to meet some of our project requirements for the Power System, but not all of them. Requirements PS-05, PS-06, PS-07, and PS-10 all related to the logging capability of the power system, and were not met. We intended to implement this logging functionality in the code for the MCU. We used an example project written by Texas Instruments for this specific MCU and modified the code to fit our needs. We decided to use this example project because it provided a template adequate for our needs. The original Texas Instruments project, called `qs-logger`, contained a data logging functionality to save data read by the evaluation board sensors. The project was designed to save the data to a choice of a USB, internal flash memory, SD card, or a host PC. Unfortunately, we were unable to successfully use this functionality in either our modified project or the original Texas Instruments project. Any test we performed to try to use the logging functionality resulted in a 'USB Error' message

appearing on the LCD screen. We could not find a solution to this problem through our research. Due to time constraints we were unable to implement the logging functionality.

We implemented the power system design on a PCB as stated in PS-12. However, there was one mistake in this PCB design. This mistake was that resistor R10 in the 3.3V regulation sub-section did not have a connection to ground. We manually fixed this mistake by soldering a wire from one pad of this resistor directly to a ground connection. This mistake has been fixed in an updated version of the PCB design, but this PCB has not been printed.

ID	Requirement	Need	Test	Result
MM-01	Mission module (excluding foam compartment) must weigh less than 907 grams.	Maximum weight allotment for mission module is 907 grams.	Weighed the camera, waterproof case, and camera mount together.	Met, 218g
MM-02	Mission module must be independent from gondola (no wired connections to other compartments in the gondola).	Allows mission module to be easily switched out for other modules.	Checked mission module compartment for wires leading to other sections.	Met
MM-03	Mission module must have a minimum wireless range of 61 meters.	Maximum required range on football field.	Tested wireless range on football field.	Not Met (see below)
MM-04	Camera must weigh less than 794 grams.	Allows for weight of other components in mission module.	Weighed the camera.	Met, 86g
MM-05	Camera must have a resolution of 1280x720 pixels (px) or greater.	Needed for high resolution photos.	Checked specifications sheet for camera.	Met
MM-06	Camera must have zoom capability of 3x or greater, either optical or digital zoom.	Needed for photos and videos during flight.	Called Sony representatives to determine zoom capability.	Met

Table 32: Results for Camera Mission Module Requirements

As shown in Table 32, we were able to meet all of our project requirements for the camera mission module except one. Project requirement MM-03 was not met. The camera alone can be wirelessly controlled from up to 91.4 meters away, as discussed in Section 8.6.3, but the final mission module design housed the camera inside its waterproof case, which severely limited the wireless range.

9.1.2 Problems Encountered

In this section we discuss problems that we encountered during the course of this project. It should be noted that these problems are largely mechanical in nature.

- The tail motor supplied by the 2011 AY team malfunctioned partway through the project and we were unable to secure a new one in time for our flight tests.

- We tested the functionality of the servo during ground tests, but the servo proved to be unreliable, so we did not use the servo during flight tests.
- The current design of the servo mount does not allow for easy access to the servo for replacement or maintenance.
- The gears in the servo mount caused a multitude of problems during the project. Our final gears were laser-cut from eighth-inch thick Delrin.
- When one of the motor mounts supplied by the 2011 AY team broke, we laser-cut a replacement out of a combination of quarter-inch thick Delrin and quarter-inch thick black acrylic. We used a glue called Poly Zap[13] to construct the motor mounts. While we were able to create suitable motor mounts, these motor mounts were difficult to construct and heavier than those used by the 2011 AY team.
- During our first flight test the blimp shell lost a substantial amount of helium within several hours. From close inspection we discovered a number of small holes in the shell where helium could escape. We patched these holes using the patch kit supplied with the blimp shell.

9.2 Recommendations for Future Teams

This project has created the groundwork for autonomous flight along with a self-monitoring power system and a camera mission module. While we are quite proud of our accomplishments, there is still room for improvement and expansion of the blimp's capabilities. In this chapter we outline potential goals for future projects using our system.

9.2.1 Optimized Autonomous Flight

One of the main goals of our project was to implement non-optimized autonomous flight. Through ground and flight testing we demonstrated that we were successful with this goal. In doing so, we have set the groundwork for autonomous control software, and the next step would be to implement optimized autonomous flight. One of the foremost aspects that must be considered when implementing optimized flight is the effect of the wind on the blimp. During our flight tests, we noticed that even the smallest gust of wind was able to blow the blimp off course. Corrections for the wind could be made possible with the use of an external wind sensor or using the accelerometer on the S3. Implementing an optimized autonomous flight may also

[13] www.zapglue.com/poly-zap

involve redesigning the system so that the blimp's drive system uses a lifting body approach to change its altitude rather than use the servo and motors for altitude adjustments.

9.2.2 Collision Avoidance

The blimp currently has no system for obstacle avoidance. When the blimp is under autonomous control it attempts to navigate between waypoints with no regard for the terrain or obstacles. A collision avoidance system could be added to prevent collisions during autonomous flight [1].

9.2.3 Power System Enhancements

The power system we designed was able to measure and display the voltage, current, and temperature data of the drive system during flight. Due to time constraints, we were not able to implement the data logging functionality that was one of our goals. We would recommend that a future team implement this functionality, in addition to displaying the logged information. The data logging functionality would be useful to show the energy usage of the drive system during flight. These logs could be compared to the autonomous flight logs to determine exactly what was happening within the gondola during flight. Additionally, the logs could be used to determine the efficiency of the power system by comparing the energy provided by the batteries with the energy used by the drive system.

We recommend that future teams make several changes to the power distribution board. Some additional decoupling capacitors had to be added to the power distribution board to improve the voltage regulation. The added capacitors were updated in the schematics but were not updated in the PCB design. Additionally, the layout of the power distribution board could be improved to maximize the space on the board, and allow for easier connections between the board and the additional electronic components in the gondola. Furthermore, fuses and circuit breakers should be added to protect vital components on the power distribution board. This would protect the components powered by the power distribution board if a short circuit were to occur. Lastly, the power system was designed to have an emergency switch that cut the power to the motors if needed. Due to time constraints we were not able to implement this feature, but we recommend that future teams implement this for safety.

9.2.4 Drive System Redesign

The current design of the drive system does not take advantage of the blimp's lifting body characteristics. If the drive system were redesigned to use the blimp's lifting body, altitude adjustments would require less power and occur more quickly.

9.2.5 Gondola Redesign

The current gondola, which was designed and constructed by the 2011 AY team, is durable and lightweight, but not very aerodynamic [1]. A more aerodynamic gondola would reduce the effect of wind on the gondola during flight. Another issue with the current gondola is the design of the servo mount. The custom laser-cut gears are difficult to use and wear down quickly. A redesign of the servo mount to use standard gears would be beneficial. The servo mount and the mounting system for the dowel could also be redesigned to reduce friction in the system, possibly through the inclusion of ball bearings [1]. This would make it easier to rotate the dowel and adjust altitude. Lastly, it would be convenient for a future team to either redesign the mounting system for the Android phone or mount it in a different location inside the gondola. The current design requires the gondola to be completely detached from the shell before the Android phone can be accessed.

9.2.6 Camera Mission Module

The camera mount was designed with the camera housed inside of its waterproof case. This added security for the camera and made the camera mount relatively simple to assemble, but it severely limited the wireless range of the camera module. Future teams should consider designing a camera mount that still securely houses the camera but does not use the waterproof case. This would greatly increase the range of the camera mission module.

The camera mount currently holds the camera at a fixed 45 degree downward angle. This provides consistency throughout the photos and videos, but does not allow the user to change the angle of the camera. Pan and tilt functions would be beneficial additions to the camera mount. The user at the ground station would be able to manually track specific targets to photograph or video record. These functions would also be useful in event photography.

9.2.7 Other Mission Modules

Mission modules give the blimp multiple capabilities and versatility. The 2011 AY team implemented a drop module and a range extension module, and we implemented a camera mission module. There are more roles that the blimp could fulfill through the addition of more

mission modules, including a radio or Wi-Fi repeater, a public address system platform, and advertisement [1]. The drop module created by the 2011 AY team could also be modified to drop flyers or other materials as desired.

9.2.8 Ground Testing Platform

For our ground testing we strapped the gondola to the top of a four-wheeled cart and operated the gondola on the top level of the East Hall Parking Garage at WPI. This setup had more friction from what we would normally expect during flight tests, and is not an optimal testing environment. We recommend that a future team develop a low friction ground effect platform for testing. This setup could be used for future implementations of the blimp or certain robotics projects.

9.3 Summary

In this chapter, we summarized the final results of our project. We discussed the gondola layout, our project requirements, and which requirements were met and which were not. We also discussed problems that we encountered throughout the course of this project, and provided recommendations for future teams.

Overall we were able to meet the majority of our project requirements. For example, we met all of the requirements for the control system and autonomous flight. However, we did not meet several requirements for the power system, and one requirement for the mission module.

The hardware for the power system was successfully completed. It is able to provide power to the drive system and additional components and provides data about the batteries' voltages and the current drawn by each motor. The software for the power system is able to read and display the data mentioned previously. Unfortunately, the power system does not calculate power or energy usage, nor does it log any data values.

We are confident that we have established a solid foundation for future teams to build upon. We have documented all of the procedures necessary to operate the blimp, including how to inflate the shell, how to connect the components within the gondola, how to operate the mission module, and how to initialize the blimp's systems. Finally, we provided recommendations for how future teams can improve our blimp system.

References

All figures not cited in this report were pictures taken by the Blimp MQP 2012-2013 team, except Figure 4, which was taken by the 2011 AY MQP team.

[1] Daniel Lanier, Daniel Sarafconn, Marcus Menghini. (2011, April 26). *Development of an Autonomous Blimp* [Online]. Available: http://www.wpi.edu/Pubs/E-project/Available/E-project-042512-141728/

[2] None. (2009, December 14). *Lifting Bodies* [Online]. Available: http://www.nasa.gov/centers/dryden/news/FactSheets/FS-011-DFRC.html

[3] None. (2012, September 13). *Comparison of Android devices* [Online]. Available: http://en.wikipedia.org/wiki/Comparison_of_Android_devices

[4] A. Mijatovic. (2012). *Areo Drum Ltd. RC Blimps Indoor/Outdoor* [Online]. Available: http://www.rc-zeppelin.com/RC%20Zeppelins%20indoor-outdoor.html

[5] R. Escher. (2003). *RC Blimp Parts* [Online]. Available: http://www.myairship.com/rcblimp/rc_parts.html

[6] C. Daniel. (2008). *Materials and Processing for Lithium-ion Batteries* [Online]. Available: http://www.tms.org/pubs/journals/JOM/0809/daniel-0809.html

[7] I. Buchmann. (2012). *What's the Best Battery?* [Online]. Available: http://batteryuniversity.com/learn/article/whats_the_best_battery

[8] None. *APDER in Action and Equipment* [Online]. Available: https://sites.google.com/site/aerialphotograpydisasters/what-we-do/apder-in-action

[9] None. (2012, September 4). *Platform Versions* [Online]. Available: http://developer.android.com/about/dashboards/index.html

[10] None. (2012, September 13). *USB Host* [Online]. Available: http://developer.android.com/guide/topics/connectivity/usb/host.html

[11] None. (2012). *Samsung I9300 Galaxy S III* [Online]. Available: http://www.gsmarena.com/samsung_i9300_galaxy_s_iii-4238.php

[12] None. (2012). *HTC Evo 4G LTE* [Online]. Available: http://www.gsmarena.com/htc_evo_4g_lte-4665.php

[13] H. Jin Kim, David H. Shim. (2003, April 8). *A flight control system for aerial robots: algorithms and experiments* [Online]. Available: http://robotics.eecs.berkeley.edu/~sastry/pubs/PDFs%20of%20Pubs2000-2005/Publications%20of%20Postdocs/ShimDavid/ShimFlightControl.pdf

[14] William Premerlani, Paul Bizard. (2009, May 17). *Direction Cosine Matrix IMU: Theory* [Online]. Available: http://gentlenav.googlecode.com/files/DCMDraft2.pdf

[15] Nisarg Kothari. (2011, August 19). *An Extended Kalman Filter for Cell Phone Orientation Tracking* [Online]. Available: http://www.andrew.cmu.edu/user/ndk/KDC_Report.pdf

[16] None. (2013). *GoPro HD HERO2: Motorsports Edition.* [Online]. Available: http://www.amazon.com/GoPro-HD-HERO2-Motorsports-Edition/dp/B005WY3TMA

[17] None. (2013). *Flash Memory Ultra-Compact Camcorder.* [Online]. Available:http://www.samsung.com/us/photography/camcorders/HMX-U20BN/XAA-specs

[18] None. (2013). *JVC PICSIO GC-FM1A HD Camcorder (Black Ice).* [Online]. Available: http://www.amazon.com/JVC-PICSIO-GC-FM1A-Camcorder-Black/dp/B002JPJII4

[19] None. (2013). *Action Cam with Wi-Fi.* [Online]. Available: http://store.sony.com/p/Action-Cam,-Compact,-Rugged,-HD-video,-action-camcorder,-Go-Pro,-Skateboarding,-Cycling,-Hiking,-Wi-Fi,-AS15/en/p/HDRAS15/B

[20] None. (2013). *Wi-Fi BacPac + Wi-Fi Remote Combo Kit.* [Online]. Available: http://gopro.com/hd-hero-accessories/wi-fi-bacpac-remote-combo

[21] None. (2012, August 29). *Sony Action Cam HDR-AS10 and Action Cam HDR-AS15.* [Online]. Available: http://www.photographyblog.com/news/sony_action_cam_hdr-as10_and_action_cam_hdr-as15/

[22] None. (2013). *Action Cam Waterproof Case.* [Online]. Available: http://store.sony.com/webapp/wcs/stores/servlet/ProductDisplay?catalogId=10551&storeId=10151&langId=-1&productId=8198552921666486342

Appendix A: Glossary

API
Application Programming Interface - a protocol used by software components to communicate with each other.

Barometer
Sensor that detects the local air pressure.

Barometric altimeter
Sensor that detects the altitude based on the air pressure.

Base station
Any devices related to blimp operations that stay on the ground while the blimp is in flight. For our project the base station consisted of the remote controller and the Android device capable of controlling the camera mission module.

Blimp
The shell and the gondola together as referred to as the blimp in this report.

Computer Aided Design (CAD)
Computer software used for designing mechanical components.

Control System
Hardware and Software used to control the drive system, which includes the Samsung Galaxy S3, Android IOIO, RC Receiver, and Multiplexer.

Diode ORing
Circuitry configuration that uses diodes to determine which batteries provide power.

Drive System
Consists of the left motor, right motor, tail motor, and servo that move the blimp.

Enumeration
An ordered list of a set of items.

Foam core
A sturdy and lightweight material that was used to control the gondola.

Gondola
Structure containing the control system, drive system, and power system for flight, as well as a mission module compartment. Attaches to the bottom of the blimp shell.

GPS
Global Positioning System - uses satellites to provide location information.

GPS coordinate
A specific location based on latitude and longitude values.

GUI
Graphical User Interface - allows users to interact with a computer system using images rather than text commands.

Gumstix Overo Water
Processing board used by the 2011 AY team as part of their control system.

Gumstix RoboVero
Expansion board for the Gumstix Overo Water used by the 2011 AY team.

Header pins
Electrical connector that consists of one or more rows of pins.

Inertial Measurement Unit (IMU)
An electronic device containing an accelerometer, gyroscope, and sometimes a magnetometer.

Integrated Circuit (IC)
A collection of active and passive devices mounted on a single slice of silicon and packaged as a single component.

Line-of-sight
An unobstructed path between two points.

Log4j
A commonly used logging system for Java applications.

Microcontroller Board (MCB)
The printed circuit board that contains the microcontroller.

Microcontroller Unit (MCU)
Small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals.

Mission Module
Interchangeable compartment in the gondola designed to fulfill a variety of purposes based on their hardware.

MOSFET
Metal–Oxide–Semiconductor Field-Effect Transistor – a transistor used for amplifying or switching electronic signals.

Power Distribution Board (PDB)

The printed circuit board that contains the main power distribution, voltage regulation, and data measurement units.

Printed Circuit Board (PCB)

A mechanical structure used to support and electrically connect electronic components.

PWM

Pulse-Width Modulation.

Radio button

Allows users to select from only a predefined set of elements in a user interface.

Remote Control (RC)

An electronic component used for wireless control of the blimp's drive system. Our remote control is used for manual control of the blimp as well as switching between manual and autonomous mode.

Shell

Inflatable balloon portion of the blimp. Our shell is a 16 feet long polyurethane shell purchased from Southern Balloon Works by the 2011 AY team.

SolidWorks

3D mechanical Computer Aided Design (CAD) program.

UML

Unified Modeling Language - a standardized set of graphic notation techniques used to model software design.

Waypoint

One coordinate in a series of coordinates to be traversed.

XML

Extensible Markup Language - defines a set of rules for encoding documents that is both human and machine readable.

Appendix B: Inverse Formula to find Distance between GPS Coordinates

http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf

Notation

a = radius at Earth's equator (6378137 meters)
f = flattening of the ellipsoid (1/298.257223563)
b = (1 – f) a = radius at Earth's poles
φ1, φ2 = latitude of the points
U1 = arctan[(1 – f) tan φ1] = reduced latitude (latitude on the auxiliary sphere)
U2 = arctan[(1 – f) tan φ2] = reduced latitude (latitude on the auxiliary sphere)
L = L2 – L1 = difference in longitude of two points
λ1, λ2 = longitude of the points on the auxiliary sphere
α1, α2 = forward azimuths at the points (initial and final bearing)
α = azimuth at the equator
s = ellipsoidal distance between the two points
σ = arc length between points on the auxiliary sphere

Problem

Given coordinates of two points (φ1, L1) and (φ2, L2)

Find azimuths α1, α2 and ellipsoidal distance s

Find U1, U2, and L, and set the initial value of λ = L. Then, iterate over the following equations until λ converges.

$$sin\,\sigma = \text{sqrt}((cosU2sin\lambda)^2 + (cosU1sinU2 - sinU1cosU2cos\lambda)^2)$$

$$cos\sigma = sinU1sinU2 + cosU1cosU2cos\lambda$$

$$\sigma = \arctan(sin\sigma/cos\sigma)$$

$$sin \propto = \frac{(cosU1cosU2sin\lambda)}{sin\sigma}$$

$$\cos^2 \propto = 1 - \sin^2 \propto$$

$$\cos(2\sigma m) = cos\sigma - \frac{2sinU2sinU2}{\cos^2 \alpha}$$

$$C = \frac{f}{16}\cos^2 \propto [4 + f(4 - 3\cos^2 \propto)]$$

$$\lambda = L + (1 - C)f\ sin \propto \{\sigma + C\ sin\sigma[\cos(2\sigma m) + C\ cos\sigma(-1 + 2\cos^2(2\sigma m))]\}$$

Once λ converges to the desired accuracy (ex. 10^-12 = 0.06mm), continue:

$$u^2 = \cos^2 \propto \frac{a^2 - b^2}{b^2}$$

$$A = 1 + \frac{(u^2)}{16384}\{4096 + u^2[-768 + u^2(320 - 175u^2)]\}$$

$$B = \frac{u^2}{1024}\{256 + u^2[-128 + u^2(74 - 47u^2)]\}$$

$$\Delta\sigma = Bsin\sigma\{\cos(2\sigma m) + \frac{1}{4}B[cos\sigma(-1 + 2\cos^2(2\sigma m) - \frac{1}{6}Bcos(2\sigma m)(-3 + 4\sin^2\sigma)(-3 + 4\cos\,\hat{}\,2(2\sigma m))]\}$$

$$s = bA(\sigma - \Delta\sigma)$$

$$\propto 1 = \arctan\left(\frac{cosU2sin\lambda}{cosU2sinU2 - sinU2cosU2cos\lambda}\right)$$

$$\propto 2 = \arctan(\frac{cosU1sin\lambda}{-sinU1cosU2 + cosU1sinU2cos\lambda})$$

Appendix C: Procedure to Install Software Development Environment

1. Download and install JDK 6 located at:
   http://www.oracle.com/technetwork/java/javase/downloads/index.html
   **Note: The Android SDK may have compatibility issues with the 64-bit JDK, so it is recommended to use the 32-bit JDK.*
2. Download Eclipse for Java Developers located at: http://www.eclipse.org/downloads/
3. Download and install the Android SDK located at:
   http://developer.android.com/sdk/index.html
   **Note: You must download the SDK Tools to use within Eclipse. To do this, expand the "Use an Existing IDE" section and click the download link.*
4. Follow the directions to install the Android SDK with Eclipse located at:
   http://developer.android.com/sdk/installing/index.html
   **Note: The blimp app code is written for Android 4.0.3 (API level 15)*
5. Download the Android IOIO libraries located at:
   https://github.com/ytai/ioio/wiki/Downloads
6. Import the IOIOLib, IOIOLibAccessory, and IOIOLibBT projects into the Eclipse workspace you will use for development.
7. Download and install TortoiseSVN located at: http://tortoisesvn.net/downloads.html
   **Note: You can also use your favorite SVN interface if you have one.*
8. Using TortoiseSVN, checkout the source tree for the autonomous blimp code located at:
   https://fusion.wpi.edu/svn/blimpmqp/Blimp12-13/AutonomousBlimp
   **Note: You will need to have read/write access to the project on WPI FusionForge.*
9. Import the AutonomousBlimp project into the same Eclipse workspace that contains the IOIO libraries.
10. From within Eclipse, right-click on the AutonomousBlimp project in the Package Explorer and click Properties.
    a. Select Android in the menu on the left side of the window.
    b. On the bottom part of the Android properties window, click Add and add each of the IOIO library projects you added to your workspace in step 6.
    c. The "Is Library" checkbox should NOT be checked.
    d. Click Apply, then OK.
11. You should now be able to build the project without any errors (there will be some warnings, however).
12. Happy coding!

Appendix D: Microcontroller Unit Operation

How to operate the Microcontroller Unit:

1. Install all the software included on the LM4F232 Evaluation Board kit, which includes all StellarisWare documentation, sample projects, and Code Composer Studio.
2. Connect the board to your computer via the USB cable.
3. Launch Code Composer Studio.
4. Import the project you wish to work on (for this project, either the original `qs-logger` project from Texas Instruments or the modified project, `energyLogger`).
5. Modify the code as desired.
6. Run the debugger.
7. Fix any errors found before launching the project (it is ok to leave warnings).
8. Once the debugger has launched, press play. This downloads the code to the board.
9. Using the debugger it is possible to add breakpoints and step through the code.
10. To see how the project works on the board, use the arrow and select buttons on the board to navigate the menus.

Appendix E: Initial Fill Procedures

Required Personnel (Requires 3 personnel):

1. Helium tank valve control
2. Hold the fill hose to the nozzle on the shell
3. Monitor the blimp

Fill Checklist:

1. Sweep floor
2. Check for pointy objects on the ceiling
3. Double check for debris on the floor
4. Lay down drop cloths – no shoes on drop cloths
5. Lay down blimp shell
6. Attach thin line to built-in points and to 7.5lbs of weight
7. Two people will confirm that the mass relief valve is closed
8. Clear tank nozzle
9. Attach fill hose to tank
10. Attach fill hose to blimp

From this point on any call of "STOP" will result in closing the helium tank valve

11. Person two holds the hose onto the blimp valve
12. Person one readies to open the helium tank valve
13. Person three calls start
14. Person one opens tank valve
15. Person three confirms clear and signals to increase fill speed
16. Blimp fill speed increases, with flow speed regulated upwards by person three and downward by any team member
17. At person three's call of "slow", the valve is set back to slow fill speed
18. Once the wrinkles in the shell are approximately 90% gone, person three will call "stop"
19. After 15 minutes, person three will call "slow"
20. Once the weights can be easily lifted, person three will call "stop"
21. Person two will remove the fill hose and cap the fill plug
22. Person one will re-cap the tank

Storage:

1. The blimp will be tied down using the built in fill points to more than 8 pounds of weight
2. The blimp will be protected with a tarp
3. There will be a paper with safety, project member, and advisor information visible in the vicinity of the blimp

Appendix F: Refill Procedures

Required Personnel (Requires 3 personnel):

1. Helium tank valve control
2. Hold the fill hose to the nozzle on the shell and monitor blimp

Refill Checklist:

1. Check for pointy objects on the ceiling
2. Attach thin line to built-in points and to more than 8lbs of weight, with a strain gauge in-between
3. Two people will confirm that the mass relief valve is closed
4. Clear tank nozzle
5. Attach fill hose to tank
6. Attach fill hose to blimp

From this point on any call of "STOP" will result in closing the helium tank valve

7. Person two holds the hose onto the blimp valve
8. Person one readies to open the helium tank valve
9. Person two calls start
10. Person one opens tank valve
11. Person two confirms clear and signals to increase fill speed
12. Blimp fill speed increases, with flow speed regulated upwards by person two and downward by any team member
13. At person two's call of "slow", the valve is set back to slow fill speed
14. Once the wrinkles in the shell are approximately 90% gone, person two will call "stop"
15. After 15 minutes, person two will call "slow"
16. Once the strain gauge reads 7.9lbs, person two will call "stop"
17. Person one will remove the fill hose and cap the fill plug
18. Person one will re-cap the tank

Appendix G: RC Transmitter and Receiver Bind Process

1. Turn the transmitter on
2. Check that MODEL1 is selected
3. Turn the transmitter off
4. Insert binding plug into the receiver
5. Turn the receiver on
6. You should see a flashing red light
7. Hold the ‘TRAINER’ switch up on the transmitter (top left corner)
8. Turn the transmitter on
9. Once the light on the receiver goes solid, let go of the ‘TRAINER’ switch
10. Turn off the transmitter
11. Turn off the receiver
12. Remove the binding plug from the receiver

Now they should be synchronized

Appendix H: Drive System Configuration Procedures

Required Personnel (Requires 2 personnel):

1. Flip switch for drive system, and for Tail motor ESC
2. Configure high/low settings for analog sticks on RC controller

Start Up Configure Checklist:

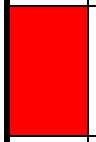
1. Person two turns on RC controller
2. Person one connects the batteries to the Power Distribution Board
3. Person two holds Right analog stick to "Max Up/Center" position
4. Person one flips the switch on the Power Distribution Board (turns on power to motors)
5. Sound will play for powering on for Left and Right motor ESCs
6. After initialization power up sound, a small pause will occur followed by two beeps, followed by a pause
7. Person two quickly moves and holds Right analog stick to "Max Down/Center" position
8. Sound will play of two beeps, followed by a single beep when finished
9. Person two holds Left analog stick to "Max Down/Max Left" position
10. Person one flips switch to turn on the Tail motor ESC
11. Sound will play for powering on Tail motor ESC
12. After initialization power up sound, three fast beeps will play, followed by a pause
13. Person two quickly moves and holds Left analog stick to "Max Down/Max Right" position and holds for 2-3 seconds
14. Person two moves (lets go of) the Left analog stick to "Max Down/Center" position
15. Left motor, Right motor, and Tail motor should now be configured for manual control.
16. Servo is capable of being control directly after Person one connects the batteries to the Power Distribution Board

Shut Down Configure Checklist:

1. Person one flips switch to turn off Tail motor ESC
2. Person one flips switch on Power Distribution Board (turns off power to motors)
3. Person one disconnects the batteries from the Power Distribution Board
4. Person two turns off RC controller

Appendix I: Detailed PWM Documentation for RC Controller

Left Analog Stick (Horizontal/Vertical)			Right Analog Stick (Vertical/Horizontal)		
Max Up/Max Left	Max Up/Center	Max Up/Max Right	Max Up/Max Left	Max Up/Center	Max Up/Max Right
Center/Max Left	Center/Center	Center/Max Right	Center/Max Left	Center/Center	Center/Max Right
Max Down/Max Left	Max Down/Center	Max Down/Max Right	Max Down/Max Left	Max Down/Center	Max Down/Max Right

Left Analog Stick								
Diagram			Control Limits		Channels Affected	Pulse Width	Signal/Drive (color)	Function
			Vertical	Horizontal				
			Max Up	Center (N/A)	Throttle	1 ms	Servo (Yellow)	Servo 0 Degrees
					Rudder	1.5ms	Tail Motor (Blue)	Tail Motor Off
					Aileron			
					Elevator			
			Max Down	Center (N/A)	Throttle	2 ms	Servo (Yellow)	Servo 90 Degrees
					Rudder	1.5ms	Tail Motor (Blue)	Tail Motor Off
					Aileron			
					Elevator			
			Center	Center (N/A)	Throttle	1.5ms	Servo (Yellow)	Servo 45 Degrees
					Rudder	1.5ms	Tail Motor (Blue)	Tail Motor Off
					Aileron			
					Elevator			
			Center (N/A)	Max Right	Throttle	1.5ms	Servo (Yellow)	Servo 45 Degrees
					Rudder	1 ms	Tail Motor (Blue)	Tail Motor Left Turn
					Aileron			
					Elevator			
			Center (N/A)	Max Left	Throttle	1.5ms	Servo (Yellow)	Servo 45 Degrees
					Rudder	2 ms	Tail Motor (Blue)	Tail Motor Right Turn
					Aileron			
					Elevator			

Right Analog Stick (Part 1)								
Diagram			Control Limits		Channels Affected	Pulse Width	Signal/Drive (color)	Function
			Vertical	Horizontal				
			Center	Center	Throttle			
					Rudder	1.5ms	Tail Motor (Blue)	Tail Motor Off
					Aileron	1.5ms	Left Motor (Red)	Left Motor On (Med)
					Elevator	1.5ms	Right Motor (Green)	Right Motor On (Med)
			Max Down	Center	Throttle			
					Rudder	1.5ms	Tail Motor (Blue)	Tail Motor Off
					Aileron	1.3ms	Left Motor (Red)	Left Motor Off
					Elevator	1.3ms	Right Motor (Green)	Right Motor Off
			Max Up	Center	Throttle			
					Rudder	1.5ms	Tail Motor (Blue)	Tail Motor Off
					Aileron	1.7ms	Left Motor (Red)	Left Motor On (High)
					Elevator	1.7ms	Right Motor (Green)	Right Motor On (High)
			Center	Max Left	Throttle			
					Rudder	2.1ms	Tail Motor (Blue)	Tail Motor Right Turn
					Aileron	1.3ms	Left Motor (Red)	Left Motor Off
					Elevator	1.7ms	Right Motor (Green)	Right Motor On (High)
			Center	Max Right	Throttle			
					Rudder	0.9ms	Tail Motor (Blue)	Tail Motor Left Turn
					Aileron	1.7ms	Left Motor (Red)	Left Motor On (High)
					Elevator	1.3ms	Right Motor (Green)	Right Motor Off

Right Analog Stick (Part 2)								
Diagram			Control Limits		Channels Affected	Pulse Width	Signal/Drive (color)	Function
			Vertical	Horizontal				
			Max Down	Max Left	Throttle			
					Rudder	2ms	Tail Motor (Blue)	Tail Motor Right Turn
					Aileron	1.1ms	Left Motor (Red)	Left Motor Off
					Elevator	1.5ms	Right Motor (Green)	Right Motor On (Med)
			Max Down	Max Right	Throttle			
					Rudder	1ms	Tail Motor (Blue)	Tail Motor Left Turn
					Aileron	1.5ms	Left Motor (Red)	Left Motor On (Med)
					Elevator	1.1ms	Right Motor (Green)	Right Motor Off
			Max Up	Max Left	Throttle			
					Rudder	2ms	Tail Motor (Blue)	Tail Motor Right Turn
					Aileron	1.5ms	Left Motor (Red)	Left Motor On (Med)
					Elevator	1.9ms	Right Motor (Green)	Right Motor On (High)
			Max Up	Max Right	Throttle			
					Rudder	1ms	Tail Motor (Blue)	Tail Motor Left Turn
					Aileron	1.9ms	Left Motor (Red)	Left Motor On (High)
					Elevator	1.5ms	Right Motor (Green)	Right Motor On (Med)

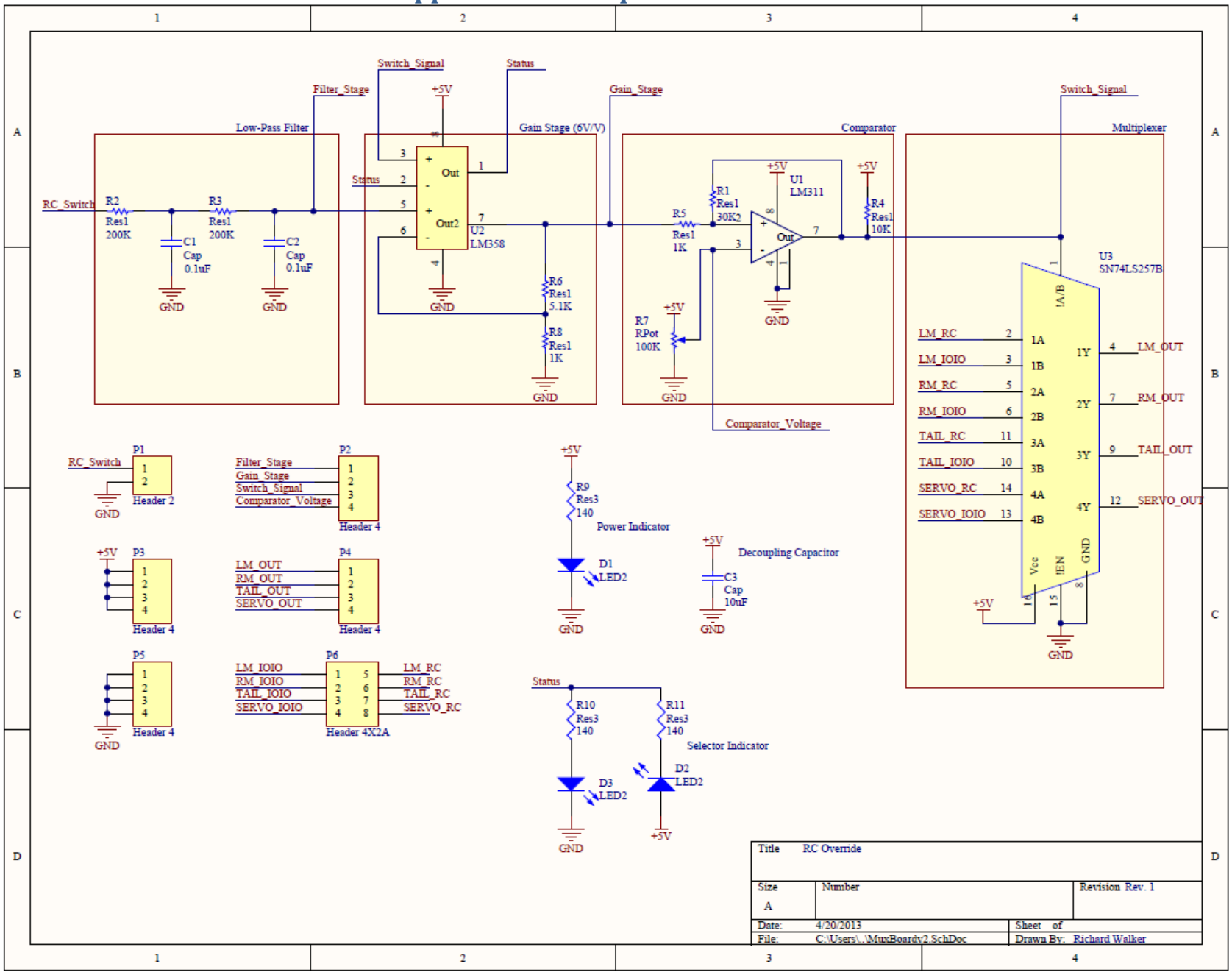
Appendix J: Pre/Post Ground Test Checklist

Pre-Test	Ground-Tests	Post Tests
1. Mount gondola on cart 2. Connect the required number of batteries to the power bus 3. Power on 4. Test RC controls for manual override 5. Test and verify RC controls for motors and servos work as expected 6. Open the Android app for autonomous flight on the Galaxy S3 7. Select GPS destination coordinate 8. Tap 'continue' to begin logging sensor data and begin autonomous navigation 9. Observe if any warning signs appear, indicating a fault in the Android system 10. If no warnings appear, place phone in gondola	1. Speed a. With servo at neutral, turn on both drive motors at equal speeds b. Change motors equally to different speeds c. We should observe the gondola moving forward in a straight line, changing speeds with the motors 2. Turning a. Set servo to neutral b. Turn on left motor to high, turn off right motor c. Turn on right motor to high, turn off left motor d. Gondola should slowly turn left or right, depending on which motors are on or off 3. Altitude a. By default, required altitude is 10 meters b. Calibrate altitude where the gondola can be raised at least 10 meters from that point c. Manually move gondola above and below the required 10 meter threshold d. Servo should change the angle of the motors as the altitude changes e. Within 1 meter of the required altitude, servo should be at neutral 4. Heading a. Set a GPS waypoint b. Move the gondola back and forth to change the heading c. Left and right motors should change speeds and try to turn right or left, depending on how the gondola is oriented toward the GPS coordinate 5. GPS a. Set a GPS waypoint b. Set the gondola so it is oriented directly at the waypoint c. The motors should run and the gondola should move toward the waypoint d. When the gondola gets within 4 meters of the waypoint, the motors should turn off	1. Power off 2. Secure and store the gondola 3. Disconnect the batteries 4. Validate Autonomous flight software has been terminated a. Remove Galaxy S3 from gondola b. Recover log files from the Galaxy S3 c. Erase logs from the Galaxy S3

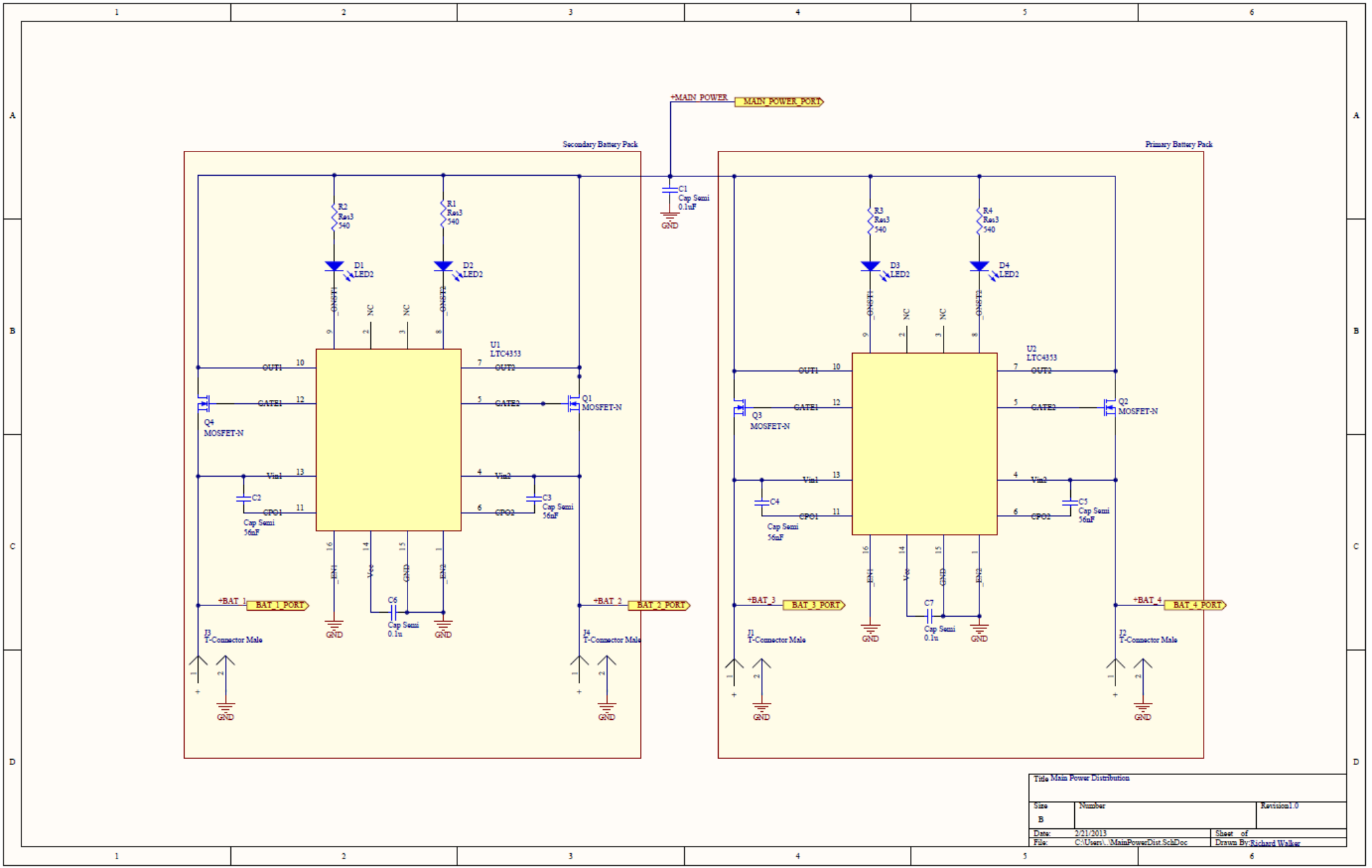
Appendix K: Pre/Post Flight Test Checklist

Pre-Flight	During-Flight	Post Flight
1. Power on Sony Action Cam 2. Sony Action Cam battery should be full 3. Insert microSD card into camera 4. Set config for wireless camera control 5. Set config for set interval times to take photos 6. Set config for video recording resolution and degree of recording 7. Ground station smartphone battery should be full 8. Establish wireless connection between ground station smartphone and Sony Action Cam 9. Place Sony Action Cam inside waterproof case on camera mount 10. Insert Camera Mission Module into the gondola 11. Connect the required number of batteries to the power bus 12. Attach the tethered rope from the blimp to a team member for safety 13. Power on 14. Check LCD and LEDs on PCB to verify battery levels are adequate for flight 15. Test RC controls for manual override 16. Test and verify RC controls for motors and servos work as expected 17. Open the Android app for autonomous flight on the Galaxy S3 18. Select GPS destination coordinate 19. Tap 'continue' to begin logging sensor data and begin autonomous navigation 20. Observe if any warning signs appear, indicating a fault in the Android system 21. If no warnings appear, place phone in gondola and attach gondola to blimp shell	1. A team member should be taking a video recording of the flight using a tripod 2. A team member wirelessly controls camera mission module and monitors the state of the video feed 3. A team member takes notes about flight 4. A team member remains tethered to the blimp 5. A team member has RC and is ready to engage manual override if needed 6. All team members are constantly on lookout for blimp failure warning signs such as frozen video feed, rapid decrease in altitude, etc.	1. Check and record the capacity levels of each battery 2. Power off 3. Secure and store the blimp 4. Disconnect the batteries a. Recover information from the power system 5. Validate Autonomous flight software has been terminated a. Remove Galaxy S3 from gondola b. Recover log files from the Galaxy S3 c. Erase logs from the Galaxy S3 6. Remove Camera Mission Module from blimp a. Close smartphone application controlling the camera b. Remove Sony Action Cam from Mission Module c. Power off camera d. Remove microSD card from Sony Action Cam e. Recover photos and videos from microSD card f. Erase photos and videos from microSD card g. Re-insert microSD card into Sony Action Cam

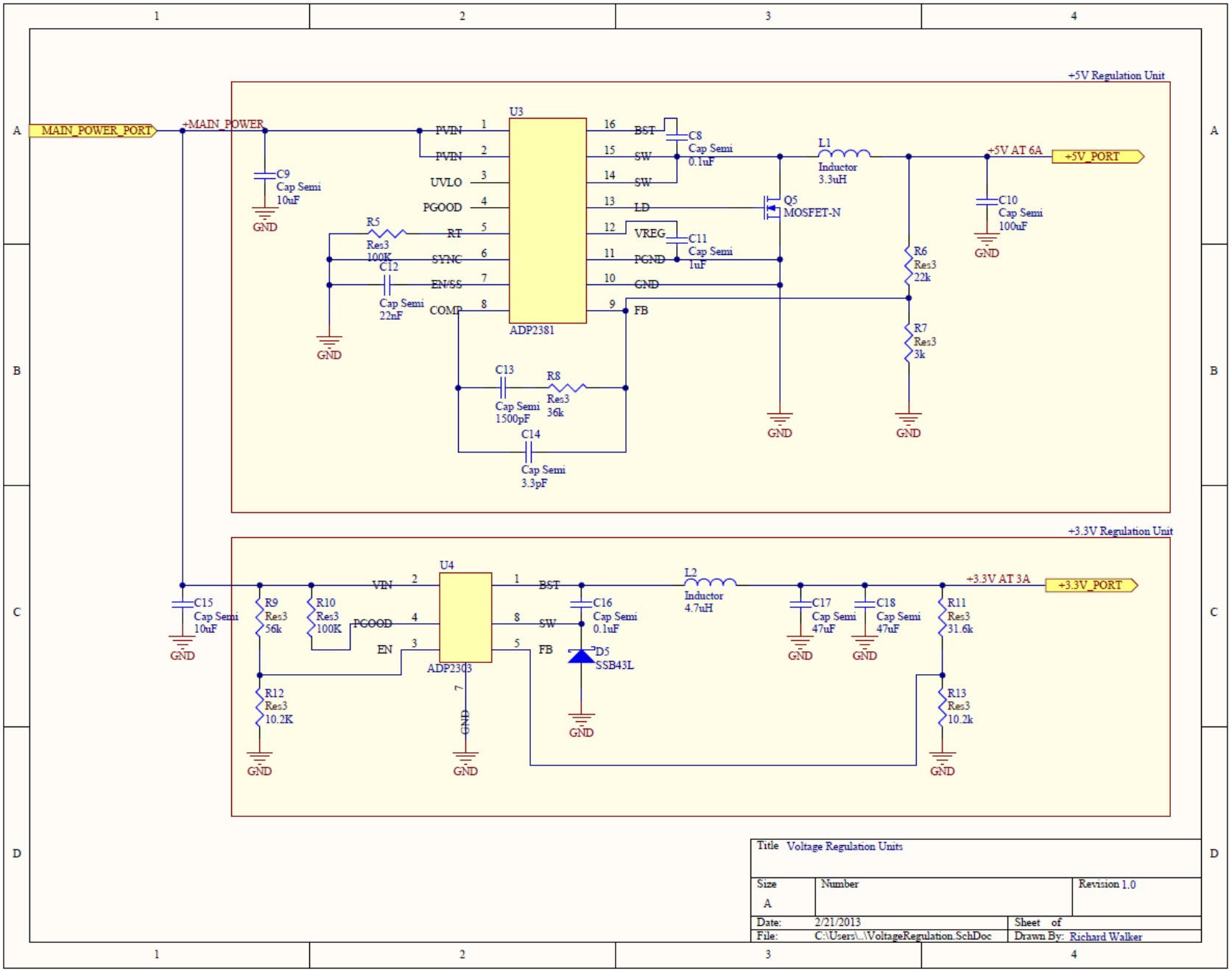
Appendix L: Multiplexer Schematic



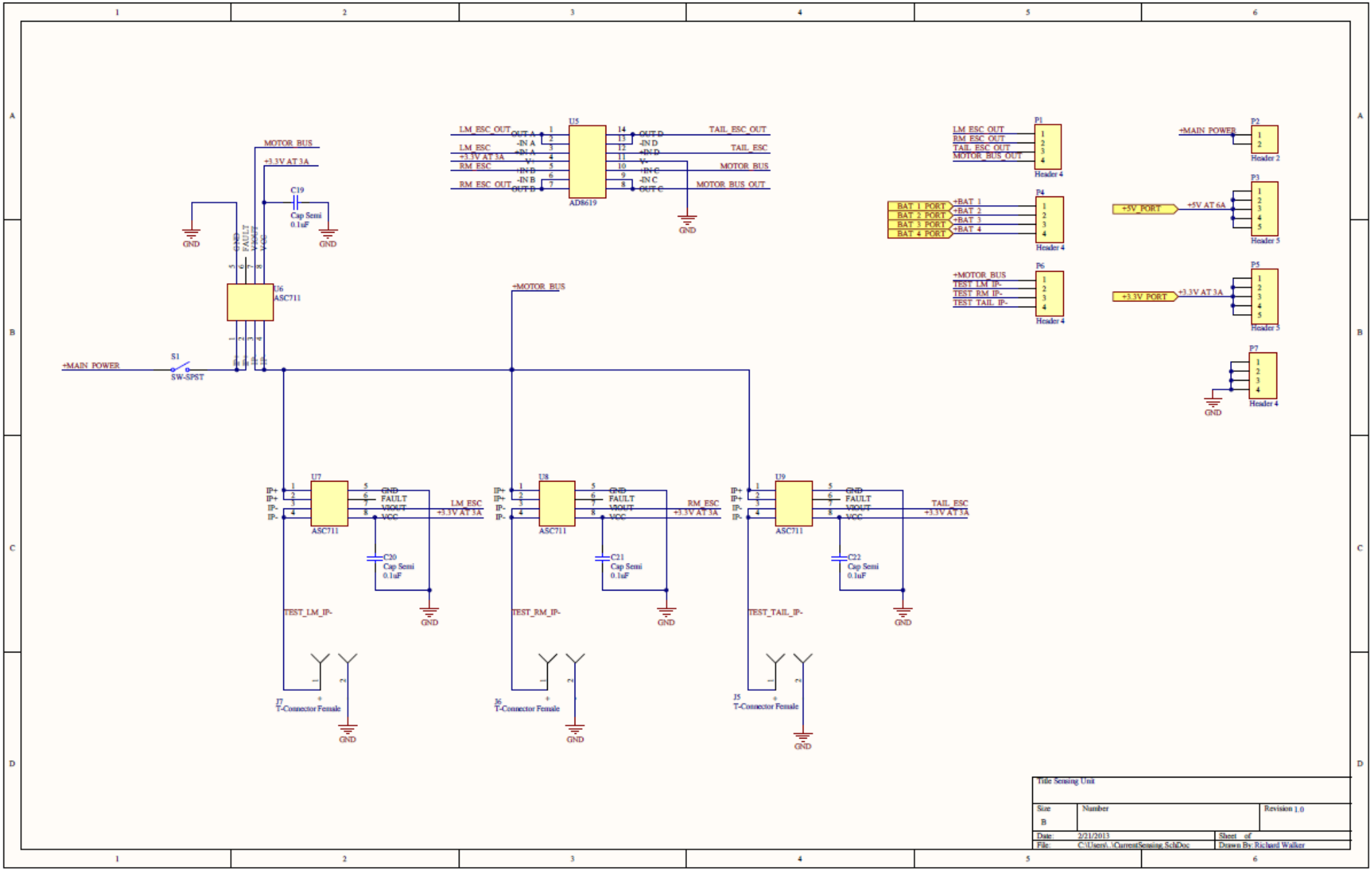
Appendix M: Main Distribution Unit Schematic



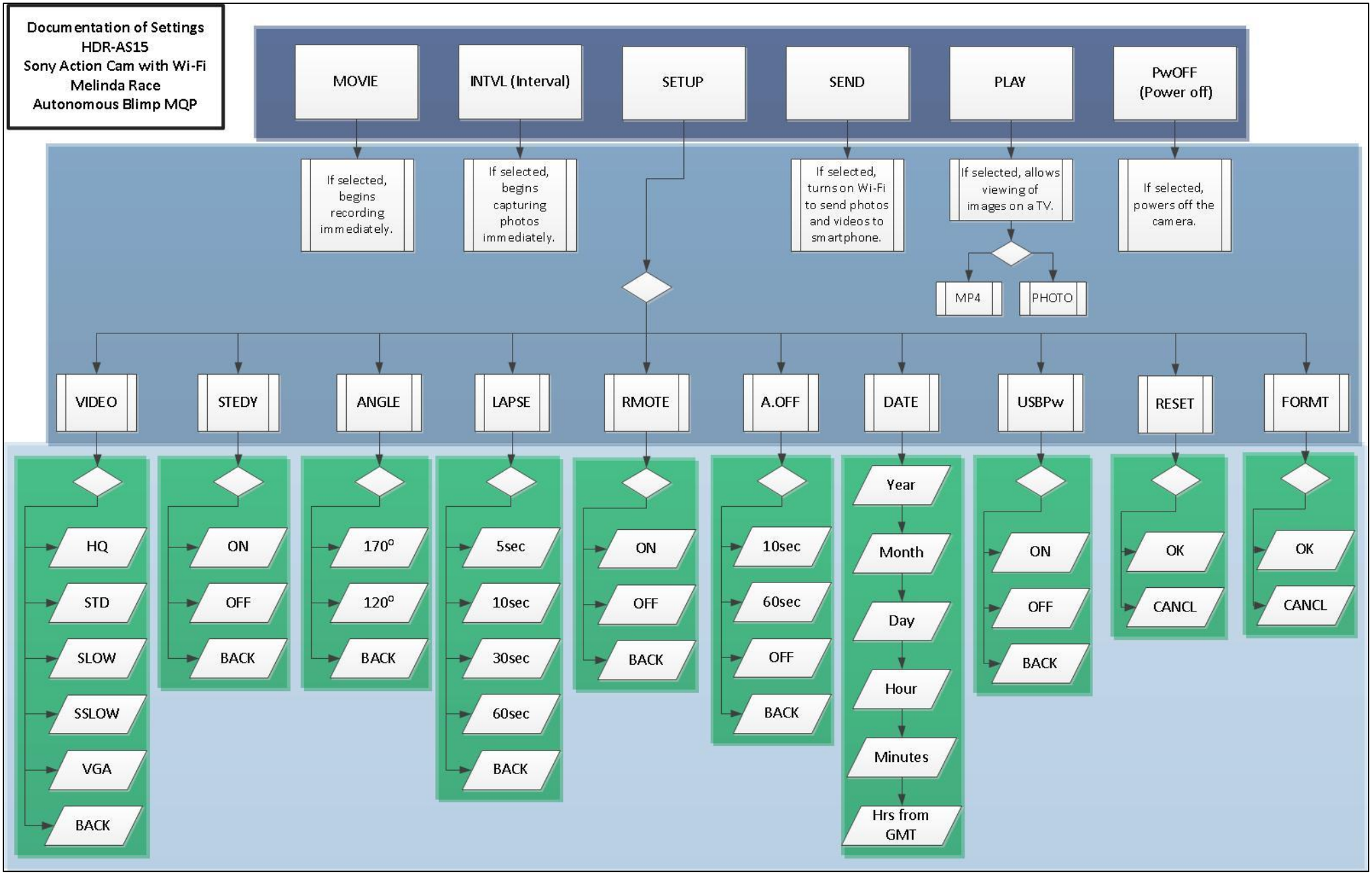
Appendix N: Voltage Regulation Unit Schematic



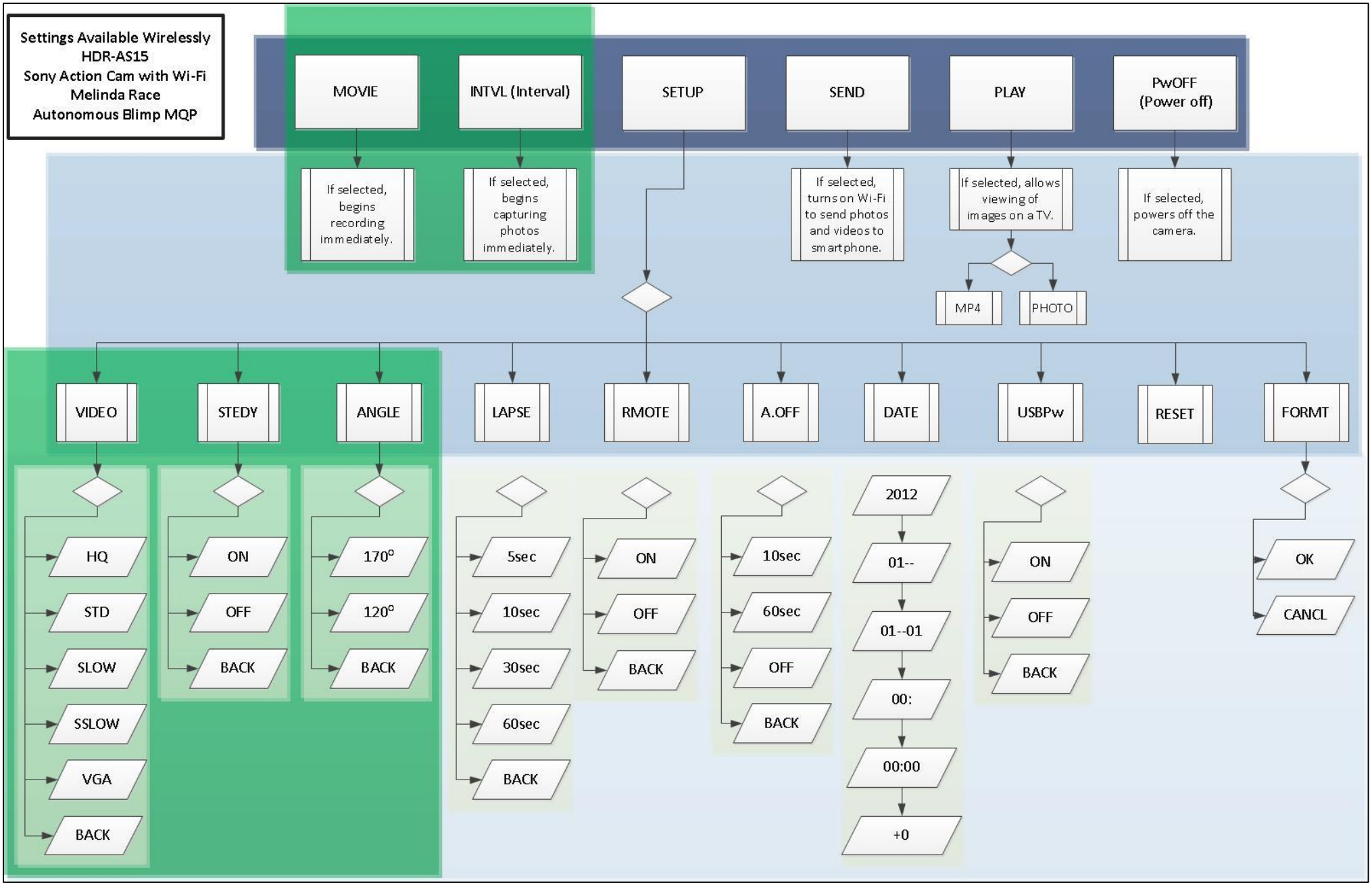
Appendix O: Measurement Unit Schematic



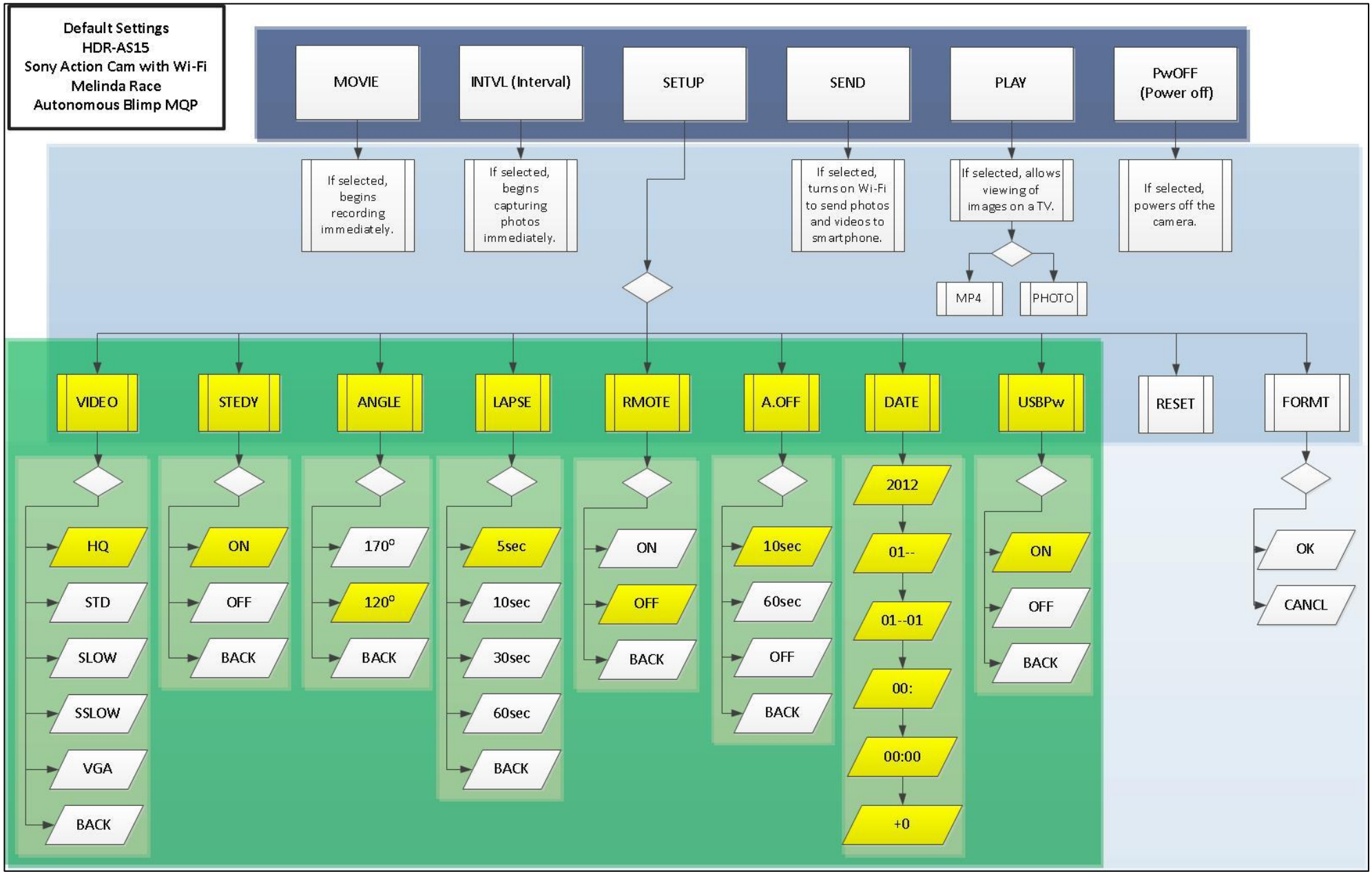
Appendix P: Sony Action Cam Setting Documentation



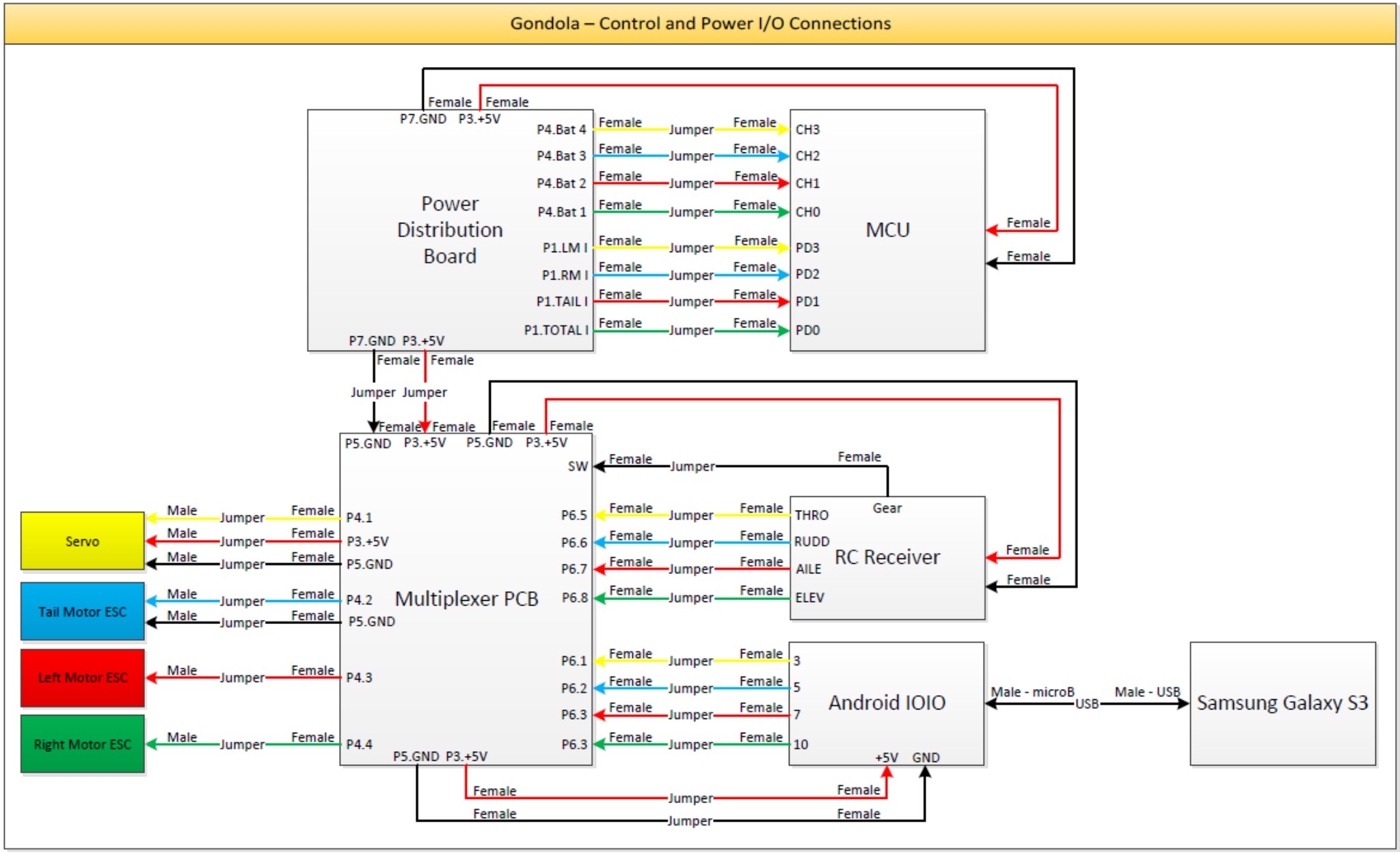
Appendix Q: Sony Action Cam – Wireless Settings



Appendix R: Sony Action Cam – Default Settings

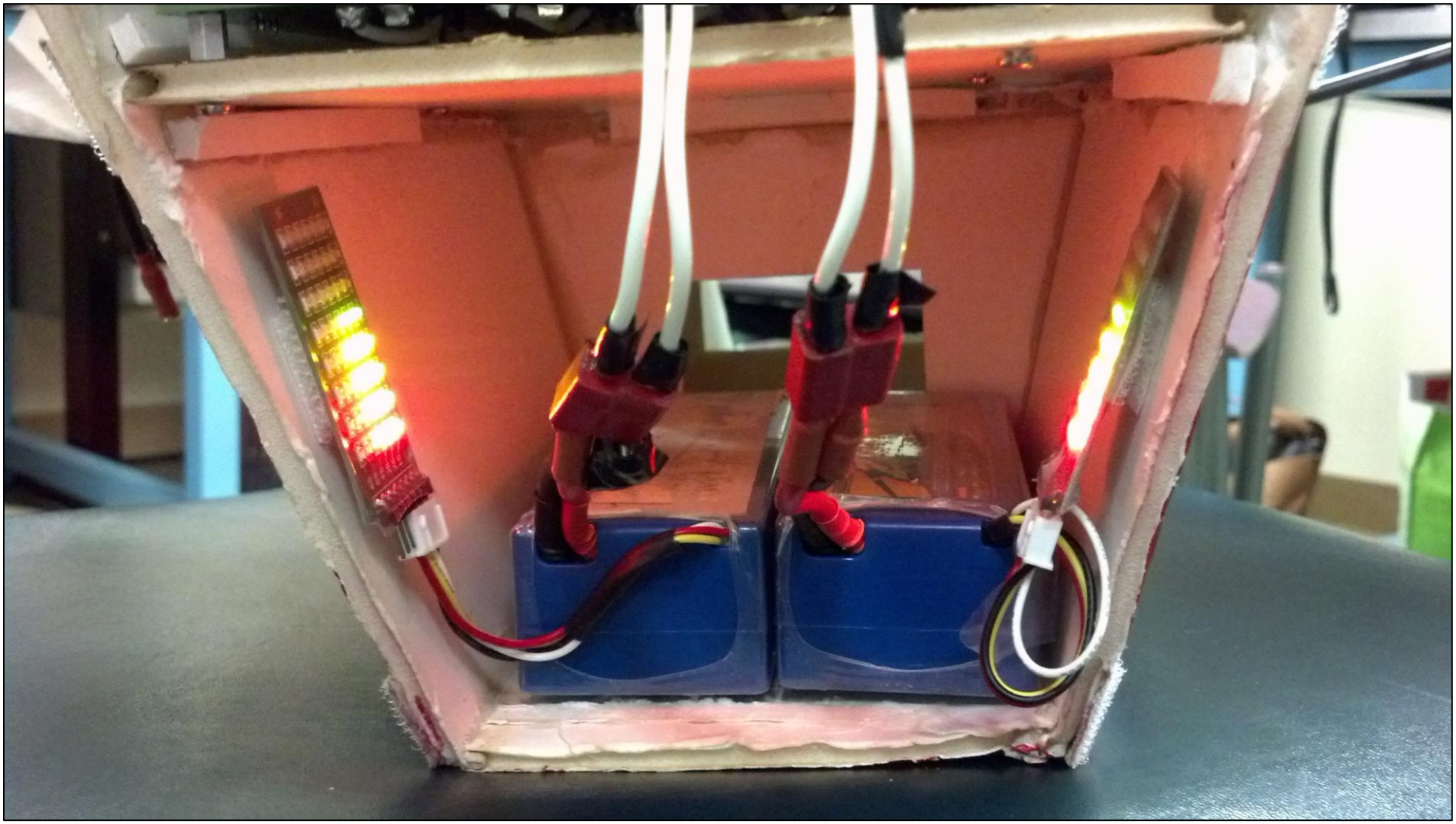


Appendix S: Control and Power I/O Connection Diagram

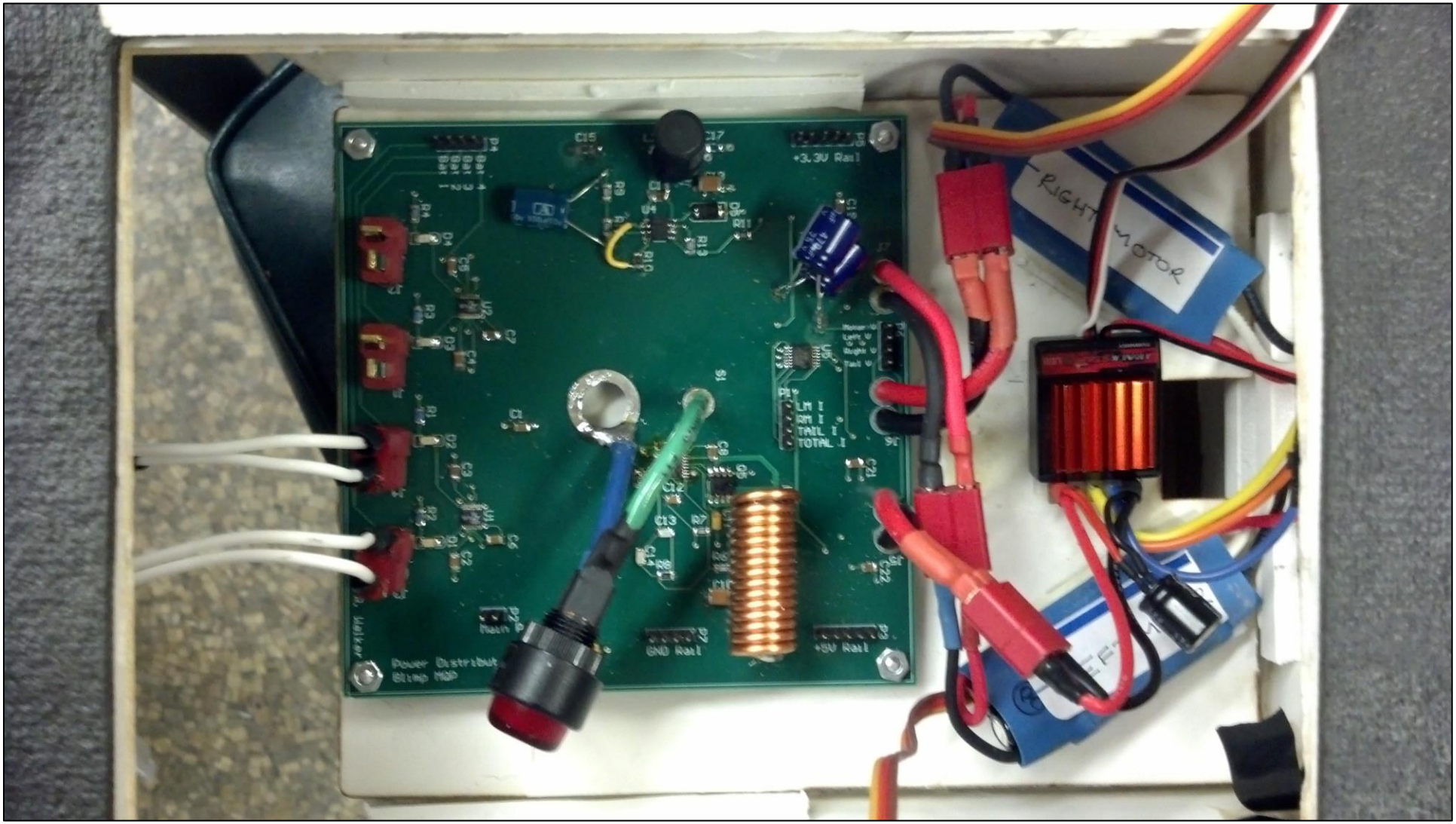


Appendix T: Pictures from Inside Gondola

Battery Compartment:



Power Level:



Control Level: