ECE/COE 1896

Senior Design

6 Degree of Freedom UAV
Final Report

Prepared By: *Long Vo, Liam Berti, Ritesh Misra, Levi Burner*

April 25, 2018

Contents

List of Figures

List of Tables

1 Executive Summary

The 6 Degrees of Freedom (DOF) Unmanned Aerial Vehicle (UAV) Project sought to improve the agility of large drones that track moving ground targets at close range, and drones that fly in constrained environments. Several example omnidirectional designs found in literature were considered for this project, including truly omnidirectional and partially omnidirectional designs. However, full omnidirectionality was found to compete with better agility. Thus, it was decided that four side rotors would be added to an existing quadrotor frame. This design allowed height holding efficiency to be maximized because the main thrust rotors were always pointed straight down. The side rotors then did not have to be used for lift, and were selected to maximize the rate of thrust change while achieving a maximized side acceleration.

The effectiveness of the design was limited by time and budget. To alleviate time constraints a large portion of the control software would come from the Robotic and Automation Society's (RAS) IARC project which 3 of the project members had been primary contributors to. To stretch the $400 budget, many parts including the frame, main thrust rotors, and propellers were curated from previous RAS projects. This choice limited some design decisions concerning the drone like choosing and interfacing a new sensor suite, designing new controllers, and designing a new power distribution and sensing board. Additionally, a new project was started that created a simple motion capture system to test the onboard localization capabilities of the UAV.

Testing was performed at each stage of the development process. The power distribution board was tested to validate the operation and accuracy of each embedded sensor and the effectiveness of the e-kill circuitry. Sensor interfacing tests were done by verifying sensors measurements and update rates. Controls were tested with a simulation and by demonstrating autonomous flight. Finally, the motion capture system was tested by comparing the measurements to the odometry generated by the UAV's on-board sensors.

The end result is a drone capable of translating without tilting. In theory, it's capable of achieve extremely high translation performance however it was limited by the quality of the velocity estimates and the performance of the height hold controller. Future work will address the quality of the velocity estimates, and damping oscillations in the height hold. Additional work will be done to enhance the motion capture system to allow for more automated comparisons of the real-time odometry to the real-time mocap.

2 Problem Definition

The popular quadrotor unmanned aerial vehicle (UAV) platform is inappropriate for applications which require independent velocity and orientation control. The degrees of freedom (DOF) that they have control of is limited to the three rotational degrees (pitch, roll, yaw), and one translation degree (thrust). Some applications of UAVs require precise positional control while maintaining a particular orientation. Due to its constraints, the quadrotor only offers limited utility in these situations.

A practical example of these limitations would be a quadrotor attempting to track moving ground target with a downward facing camera. Due to the physical constraints of the system, the quadrotor must tilt its camera in a direction opposite of direction of the target in order to propel itself towards the target (a pitch or roll). Thus the quadrotor's tracking will have difficulty locating the target. If the acceleration required is large enough the quadrotor would have to rotate the camera such that the target is not in the field of view. This renders tracking impossible, or inaccurate.

A 6 DOF UAV would solve this problem, and other problems requiring precise motion profile tracking. This project proposes to develop a new drone capable of high performance, 6 DOF control while the UAV remains in a level stance. It will be suitable for precision tracking, payload positioning, and maneuvering in tight spaces at high speeds.

The UAV will be capable of following pre-programmed motion profiles using on-board localization methods and controllers. This will allow it to operate completely autonomously from takeoff till landing. There will also be an option for a human pilot to provide velocity targets through a remote control. Additionally, as a hardware component a power distribution board will be produced that provides energy usage information, and an emergency stop system to enhance the system's safety.

We are confident that this type of aerial platform will not only be achievable, but its enhanced application cases will justify the cost of its additional control surfaces. With the rise of UAVs operating in GPS denied environments, and narrow working environments, precision movement without rotational movement will become a desirable feature. Larger and more expensive payloads will also ensure that lateral movement without perturbing the load is a necessary feature on utility UAVs.

3 Background

Quadrotor UAV's have the disadvantage of having non-holonomic control of their position and orientation. They have only 4 degrees of control, angular pitch acceleration, angular roll acceleration, angular yaw acceleration, and a single translation thrust vector that is normal to the rotational plane of the four main thrust propellers. In order to translate from one position to another, they must first rotate towards their intended destination and then rotate the other way to come to a stop at their destination. This significantly limits the use of UAV's for operations such as tracking ground targets at close range, maneuvering of large drones in constrained environments, and precision pick up and drop off of objects.

There are many ways to achieve a 6 DOF UAV. All involve adding additional rotors so that the achievable propeller thrusts normals span three-dimensional space. Many designs in literature maximize control effort availability regardless of orientation. In [4] [5] [6], hexacopter designs are proposed that employ three pairs of rotors oriented such that the normals of the planes span three-dimensional space. Another approach is to use tilting rotors as seen in [7] [8]. A completely omnidirectional design using a novel propeller configuration is demonstrated in [9] [10].

These designs all attempt to solve the problem of achieving 6 Degree of Freedom (DOF) control for a UAV, regardless of its orientation. This results in inefficient usage of control effort as the rotors are often placed at significant angles to the desired thrust direction. This is necessary in order to maximize the possible orientations and translational velocities achievable by the platform. However, it is not necessarily useful for the above listed applications of 6 DOF UAV's.

A promising, but not well researched, design consists of a quadrotor with four additional rotors mounted perpendicularly to the main thrust plane. This rotor configuration will allow control of velocity while allowing the UAV to remain level. Additionally, the control effort required to hover does not increase past that required by a quadrotor. While the side rotors will require additional power, it is predicted that the energy usage of the side rotors compared to the main rotors will be minimal.

The concept of side rotors has been explored in [11] [12] where single side-rotor was added so that a multicopter can accomplish a specific task. The idea of using four side rotors for holonomic control was also explored in [13]. However, the results did not clearly demonstrate superiority or lack thereof

of the design compared to a quadrotor design.

An expected advantage of the proposed 6 DOF design is greatly increased accuracy and speed of translation performance compared to a typical quadrotor. Localization accuracy will also improve due to the drones sensors remaining in a fixed orientation compared to the region of interest used for observation. In light of recent studies and the above reasons, we think the side rotor design is neglected and deserves another look.

In order to achieve this goal, an Aerial Platform, Power Distribution Board, Localization Method, and Controller need to be designed an implemented. The Aerial Platform will encompass the design of a quadrotor with the additional side propellers. It will carry the onboard computer and sensors require for autonomous operation and 6 DOF control. The Power Distribution Board will handle the emergency stop components and provide independent power usage monitoring for each motor. The Localization method will allow the drone to operate in an indoor environment without global reference points, e.g. GPS denied operation. Finally, the controller will handle the task of tracking setpoints or motion profiles.

A quadrotor with side propeller's at first seems like an excessive idea. However, after reviewing the applications and advantages it is clear that it can easily achieve goals that regular quadrotors cannot. The goal of this project will be to design such a UAV and demonstrate its superiority for certain applications.

4 System Requirements

This section contains the overall requirements for the 6 DOF UAV. Overall, the requirements reflect the intended usage of the vehicle to demonstrate utility for certain UAV applications. It will not be a commercially viable product when completed, however it will be reasonably easy to use for a trained individual.

Four major subsystems have been identified. The Aerial Platform, Power Distribution Board, Localization, and Controller. Each will be outlined in their own section.

4.1 Aerial Platform

The Aerial Platform contains all the components required for a typical human piloted quadrotor. It also contains the additional hardware components required for 6 DOF actuation and hardware required for autonomous operation. It does not include the power distribution board as that is a separate, specifically designed system component.

4.1.1 6 Degree of Freedom Actuation

The Aerial Platform will be able to apply acceleration along all 6 degrees of freedom of a rigid body.

4.1.2 High Maneuverability

The Aerial Platform must be sufficiently maneuverable in order to track moving targets moving at speeds up to a couple meters per second and rapidly pick up and place payloads.

4.1.3 On-board Computer

There will be an on board computer capable of running the localization and flight control system.

4.1.4 Altimeter

There will be sensors capable of estimating height and accurate enough for use in an indoor environment.

4.1.5 Optical Flow Sensor

A sensor capable of optical flow estimation will be available and will provide reasonable velocity estimates.

4.1.6 Flight Controller

A commercially available flight controller capable of orientation stabilization will be available.

4.1.7 Battery

The Battery will be a Lithium Polymer Battery capable of a high enough discharging rate to power all motors simultaneously.

4.1.8 Flight Time

The drone should be able to fly for a minimum of 5 minutes. However, 10 minutes is the target.

4.1.9 WiFi Connectivity

The onboard computer will be equipped with Wifi connectivity.

4.1.10 RC Connectivity

The flight controller will accept input from a human pilot. The pilot will be able to take over, preferably by flipping a switch on his remote.

4.1.11 Downward Facing Camera

The downward facing camera will provide the capability to autonomously identify objects in a frame, and provide frames to the state estimator in lieu or in addition to the optical flow board.

4.2 Power Distribution Board

The power distribution board provides a central interface that connects the hardware subsystems of the design together. This interface includes connectivity between the power supply batteries and the rest of the hardware within the system. This custom PCB will also include safety features such as an isolation barrier and remote emergency stop functionality.

4.2.1 Power Distribution

The power distribution board will provide power to all hardware components within the system. The distribution will allow for operation over the full range of input voltages of the LiPo battery supply. The board will provide voltage regulation to produce supply voltages to any components that do not operate off the nominal voltage of the battery. The board and its components will also be properly rated for all maximum voltages and currents including a maximum continuous current draw of 30A for each motor.

4.2.2 Electrical Isolation Barrier

The power distribution board will provide an electrical isolation barrier between the high and low voltage sections of the system. This isolation barrier will provide the control system with immunity from any noise and high voltage transients generated by the motors.

4.2.3 Current Monitoring

The power distribution board will provide current monitoring capabilities for each individual motor within the system.

4.2.4 Voltage Monitoring

The power distribution board will provide voltage monitoring capabilities for the LiPo battery supplying power to the system.

4.2.5 Remote Kill

The power distribution board will provide the ability to remotely cut power to all motors within the system. The control signals for the emergency kill will be received from an external wireless receiver.

4.3 Localization

The localization system provides the controller with estimates for position, velocity, and acceleration depending on inputs from some of the sensors outlined in the Aerial Platform section.

4.3.1 Update Rate

The Localization system will efficiently gather input from the sensors and fuse them to create its estimates at a sustainable rate.

4.3.2 Filtering

The Localization system will fuse together all of its sensor readings into the single most likely estimate for all of its output parameters, most likely with an extended Kalman filter.

4.3.3 Motion tracking system

The Motion Tracking System will provide position estimates of greater accuracy than the onboard sensors to allow for more rigorous testing of the controllers.

3D cameras such as the Kinect, or Intel Realsense cameras will be used to provide marker-less motion capture. Markers will be used if this method proves unreliable.

4.4 Controller

The controller allows the system to be autonomous. It depends on the localization system generates control outputs based on the desired motion set-points.

4.4.1 Inherently Controlled Degrees of Freedom

The controller will attempt to directly correct errors in velocity and position from a desired set point. This will allow arbitrary paths in 3D space to be followed while achieving a different velocity at each position.

4.4.2 Acceptance of Motion Profile Input

The controller will accept Motion Profile's as input. The motion profile must contain position, velocity, and acceleration targets as well as a time stamp associated with each target. The set-points can be dated in the future and will be executed when appropriate.

4.4.3 Acceptance of Human Input

The controller will allow human input of target velocities and vertical heights. It will attempt to achieve these setpoints using the maximum acceleration limits.

4.4.4 Autonomous Takeoff and Landing

The controller will be capable of autonomous takeoff and landing.

4.4.5 Existence of Rated Limits

Rated limits of jerk, acceleration, and velocity will be available. The controller will guarantee hitting the targets with a minimum accuracy statistic provided the input motion profile respects the limits.

5 Design Constraints

5.1 Aerial Platform

5.1.1 Size

The platform will be based on Robotics and Automation Societie's (RAS) X525 drone frame. It's max outer dimension in any direction is to be 1 meter.

5.1.2 Weight

The target weight is limited by the current thrust capabilities of RAS's X525 drone. It's max takeoff weight is 2.5 kg.

5.1.3 Minimum Ratio of Side Rotor Thrust to Model Weight

The minimum ratio of side rotor thrust in grams to model weight will be 0.6.

5.1.4 Ratio of Main Lift Rotor Thrust to Model Weight

The minimum ratio of side rotor thrust in grams to model weight will be 1.5.

5.1.5 On-board Computer

The On-board Computer will run Linux. It will have at least a quad core Cortex-A53 ARM processor clocked at 1 GHz with 1 GB of RAM. The computer will be capable of running Robot Operating System (ROS) with some threads updating as high as a couple hundred hertz.

5.1.6 Altimeter

Two laser based rangefinders will be used. The reason for two is twofold. Two rangefinders provides redundancy in the case of rangefinder failure. They also will have different, possibly complementary measurement ranges.

5.1.7 Optical Flow Sensor

The optical flow sensor will be ASIC based, capable of 5 rad/s measurements or higher with a 50 Hz or higher update rate.

5.1.8 Flight Controller

The flight controller will run the PX4 firmware. It will be capable of running its control loop at 400 Hz as well as forwarding IMU measurments to the on-board computer at the same rate. It will interface with the on-board computer via USB.

5.1.9 Battery

The battery should be between 3 and 6 cells and capable of at least 150A continuous discharge and 250A peak discharge.

5.1.10 WiFi Connectivity through On-board Computer

The WiFi connectivity will be provided using the on-board computers default hardware. It will be suitable for streaming of live video feeds and sensor data.

5.1.11 RC Connectivity

The RC Connectivity will be achieved with RAS's Taranis X9D radio and a Taranis XSR receiver.

5.2 Cost

In total, with the components required from the Power Distribution board the Platform must not be built with more than $400 of Swanson School of Engineering ECE funds.

5.3 Power Distribution Board

5.3.1 Supply Voltage

The power distribution board will provide supply voltages to all components within the system. The high voltage motor section of the system will be able to operate using up to a 6 cell battery, and the lower voltage section of the system will operate using a 3 cell LiPo. The input voltage range of a single cell of a LiPo spans from 3.7V - 4.2V.

5.3.2 Assembly

All components used on the power distribution board will be able to be placed and soldered by hand. This restricts the package style of components available for use on the board, and prevents components without leads such as BGAs or QFN packages or any components with lead pitch less than 1mm from being used.

5.3.3 Current Draw

The power distribution board must be able to provide up to 30 amps of continuous current to each individual motor. The copper weights and trace widths of the PCB, along with any components interfacing with this subsystem of the board must be able to operate at this maximum current without being damaged.

5.4 Localization

5.4.1 On-board localization

The UAV must localize itself using only the optical flow sensor, on-board camera, IMU, altimeter, and other sensors available on the flight controller. This is so the UAV is capable of being fully autonomous without external sensors.

5.4.2 Update Rate

The update rate shall be fast enough to provide accurate localization, without sacrificing computational time. The extended Kalman filter must run quickly enough to achieve an update rate of more than 300 Hz.

5.4.3 Motion tracking system

The motion tracking system must provide an accurate ground truth (+/-5% of actual position) using no more than two (depth) cameras.

5.5 Controller

5.5.1 Control of Height and Vertical Velocity

The controller will use the localization system to calculate actuator output and compensate for position and velocity error in height. It's specific output will be an average throttle value to be achieved by the four main thrust ESC's.

5.5.2 Control of Horizontal Position and Velocity

The controller will use the localization system to calculate actuator output and compensate for position and velocity error in the level plane. It's specific output will be a throttle value to each of the side rotor ESC's that achieves the desired 2 dimensional thrust vector through superposition.

5.5.3 Control of Orientation through a Flight Controller

Control of absolute orientation in the local frame will be achieved through an onboard flight controller such as Pixhawk V1 running the popular PX4 firmware. The flight controller will adjust the individual throttle percentage of the main thrust ESC's based on internal orientation sensors, desired orientation, and the average thrust value desired.

5.5.4 Update Rate

The controller will update at the highest rate accepted by the flight controller. For a Pixhawk V1 running PX4, this typically around 400 Hz.

5.5.5 Advertised Maximum Jerk Limit

The controller will have an advertised and available maximum jerk target it can achieve when executing a motion profile. This will be made available to a motion planner.

5.5.6 Maximum Acceleration Limit and Goal

The controller will enforce a separate maximum acceleration limit for the translational velocity and the vertical velocity. This is to help decrease the chances of untested stability issues.

The goal is to achieve accelerations of 6 m/s^2 for translation control and 4.9 m/s^2 for vertical control.

Ultimately the maximum acceptable acceleration will be based on the controllers ability to maintain such an acceleration and maintain the velocity tracking accuracy requirement.

5.5.7 Maximum Velocity Target and Goal

The controller will enforce a separate maximum velocity limit for the translational velocity and the vertical velocity.

The goal is to achieve velocities of 5 m/s for translation control and 1.5 m/s for vertical control.

Ultimately the maximum acceptable velocity will be based on the controllers ability to maintain such a velocity and maintain a useful the positional tracking accuracy requirement.

5.5.8 Rated Maximum Velocity Error

The controller will have a rated velocity accuracy derived from automatic testing. The statistic will only be guaranteed to be useful if the jerk, acceler-

ation, and velocity limits are respected in commanded motion profile.

5.5.9 Rated Maximum Positional Error

The controller will have a rated positional tracking accuracy derived from automatic testing. The statistic will only be guaranteed to be useful if the jerk, acceleration, and velocity limits are respected in commanded motion profile.

5.5.10 Intended Hardware

The controller will be designed to run as a ROS node on a Linux based computer.

5.5.11 Debug Information

Extensive debugging information will be available through ROS topics so that the current state of any variable can be displayed and correlated with another state.

5.5.12 Open Source Software Usage

The controller will be built on top of RAS's current Controller framework intended for the IARC project. This software already handles many of the corner cases required for safe operation and provides a safe and easily extensible framework for continued modification.

The controller will also depend on RAS's general purpose flight controller communications ROS package that allows communication with multiple flight controller firmware packages including Cleanflight, Crazyflie, and soon PX4.

5.5.13 Maintaining Compatibility with Existing Software

The control system will remain compatibility (e.g. no forking) of the current RAS IARC stack.

6 Evaluation of Design Concepts

6.1 Aerial Platform

6.1.1 X525 Frame

A UAV frame was available from a previous RAS project. It was already equipped with 4 motors and ESC's. Plant modeling revealed that each propeller was capable of providing over 1 kg of thrust each. The frame was modified to make room for additional rotors. While the X525 was not the ideal base frame for the design, it was the only that could be used due to budget restrictions.

6.1.2 Side Rotor Location

6.1.2.1 Side Rotors Mounted to Main Thrust Arms

The drone was required to have 4 side rotors with thrust normals perpendicular to the main thrust rotors. Figure 1 outlines a possible configuration found in literature. This design benefits from reusing the same arms used for the main thrust rotors for the side rotors. This saves weight in the air frame design.

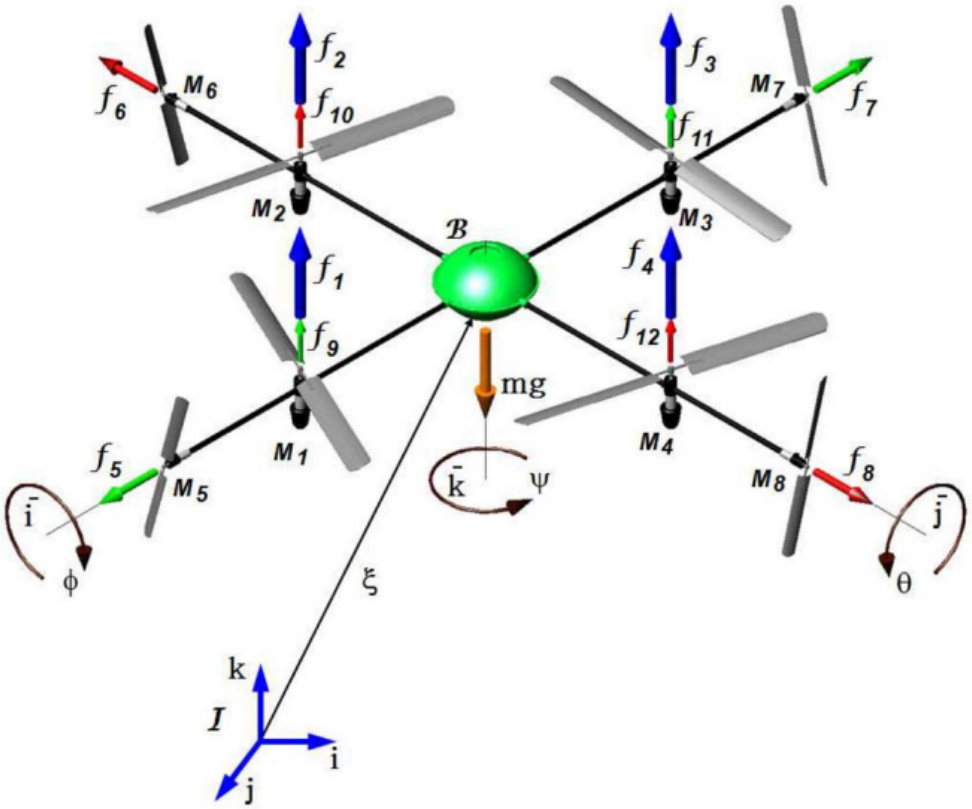

Figure 1: Example Configuration of Side Rotors [10]

6.1.2.2 Side Rotors Mounted in-between Main Thrust Rotors

Another configuration considered was placing the side rotors on separate arms that ran between the arms for the main thrust rotors. This added weight to the frame but was expected to reduce the turbulence caused by interfering rotors. It was also simpler to construct and allowed the vertical position of the side rotors to be adjusted so that the thrust normals intersected at the center of gravity. This can be seen in Figure 2.

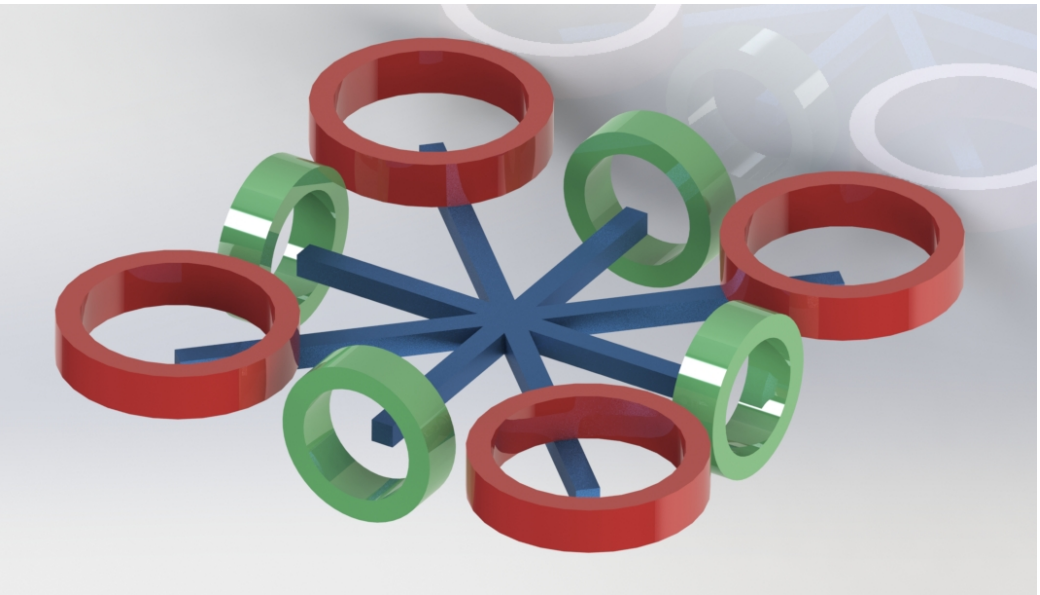

Figure 2: Side rotors in-between main thrust rotors

In the end it was found that the placing the side rotors as shown in Figure 2 did not offer enough clearance between the side rotors and the main rotors. To avoid this issue the rotors were shifted further from the center frame. The results of this adjustment can be seen in Figure 3.

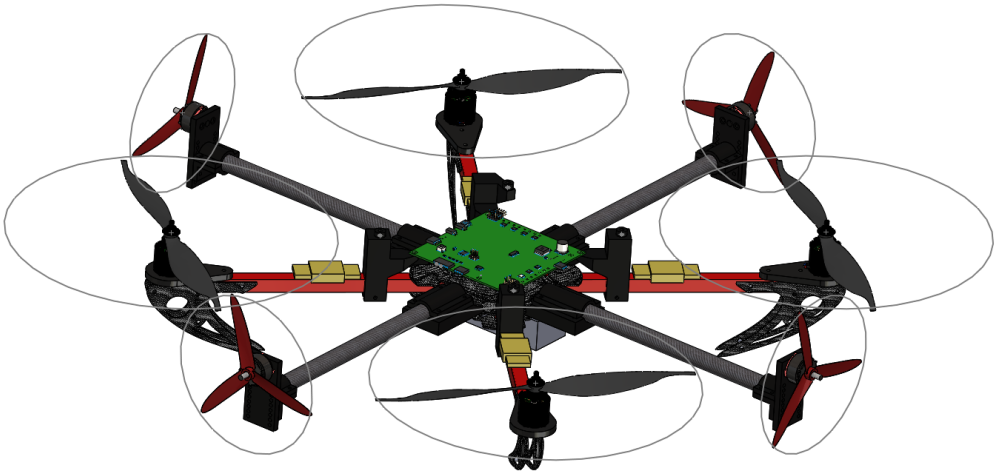

Figure 3: Actual side rotor placement used

6.1.3 FPV Racing Motors and ESCs

It was decided that the side prop motors would be FPV racing drone motors in order to maximize agility. Racing drones are known for their cheap price and high performance. They are designed to quickly change thrust. A few candidates were identified basic on a list of high performance motors identified in a blog post "Best FPV Motors of 2017" by a reputable FPV drone blog, "Oscar Liang" [14].

Two propeller and ESC combinations were seriously considered from an initial list of 5. The initial list was narrowed down to two based on obvious comparison of price and thrust ratios. The EMAX RS2205S 2300 kV with EMAX 30 A Bullet ESCs and EMAX RS2306 2400KV with Wraith32 V2 ESC's were considered. The RS2205S combination was selected because it was the cheapest at $97 vs $148 for a set of 4 and since benchmarks from an independent testing website showed that the performance was similar [15], [16]. The RS2205S was predicted to make 1500 grams of thrust with a 4S LiPo and the RS2306 was predicted to make 1687 grams of thrust with a 4S LiPo.

The motor chosen can be seen in Figure 4.

Figure 4: RS2205-S FPV Motors [1]

In the end these motors performed as expected. Producing up to 0.8 kg of thrust with a 3S LiPo. This peak thrust is lower than previously quoted due to a final propellers used being different than those used by the independent testing website and the battery voltage being lower.

6.1.4 Side Rotor Propellers

It was clear from the Mini Quad Test Bench results that there was a range in the size, pitch, and number of blades that could be used. Specifically, from 4-

6” width with 4-6” pitch between 2-3 blades. At the beginning of the project, RAS acquired such a range of propellers with different airfoil shapes for the IARC project. This set of propellers was tested and the most desirable pair based on total thrust, efficiency, and peak current consumption was chosen. In total 6 different propellers were chosen. From this testing became clear that larger propeller universly created more thrust. Additionally, it was found that all propellers had similar efficiency in terms of grams of thrust produced per watt. The lightest propeller was then chosen that got as close to the ESC current rating of 30A at 100% throttle without exceeding it. The resulting choice was the Master Airscrew 6x4.5 dual bladed propeller.

A graph showing the filtered and curve fitted Thrust to Amp plots can be seen in Figure 5.

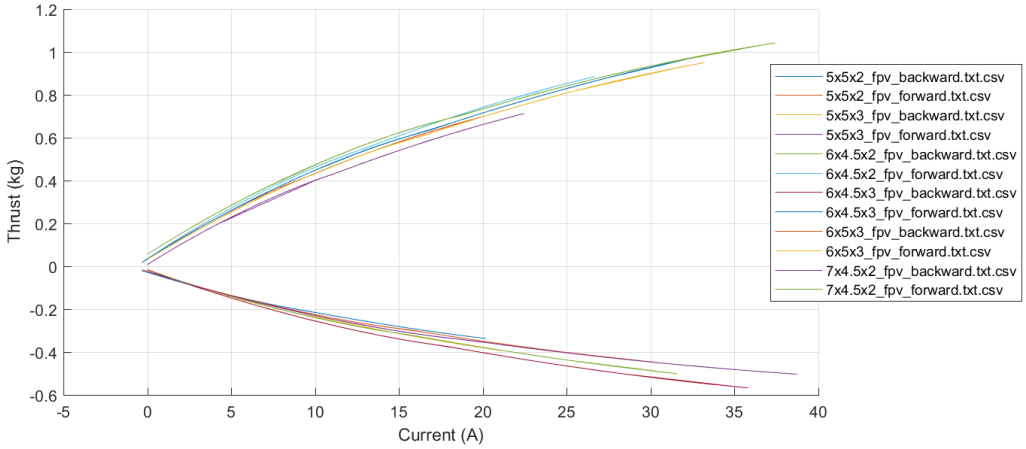

Figure 5: Thrust to current graphs for 6 different side propellers on the EMAX RS2205 motor with a 3S battery. Thrust to amp curves are shown for forward and reverse thrusts.

An interesting revelation from the above graph is that the only about half the thrust could be produced when running in the reverse direction. This is due to the propellers not being designed to provided thrust in the reverse direction. Because running in reverse would result in air blowing on the center drone (resulting in further thrust loss) it was decided to not run propellers in reverse and to only run all side rotors in a pusher configuration.

6.1.5 On-board Computer

The on-board computer needed to run embedded Linux so that ROS packages could be used. Originally, it was also required to have SPI/I2C/USB interfaces so that input can be accepted from sensors. Wifi connectivity was required so that video and sensor data can be streamed for debugging. Finally the computer needed to be able to run the ROS system at high and stable update rates in order to enable high performance motion.

The Raspberry Pi 3 was considered first. It had SPI and I2C lines that could be used with certain sensors, such as the LidarLite. It also had USB ports that can be used with cameras. Ths four ARM Cortex-A53 cores that run at 1.2 GHz each and accompanying 1 GB of RAM were considered fast enough to run all applications. There was also a 2.4 GHz 802.11n WiFi support.

The Jetson TX1 has also considered. Like the Raspberry Pi 3 it had SPI, I2C, and USB ports, the exact configuration of which is determined by the carrier board. It has four ARM Cortex-A57 cores that can run at a maximum of 1.9 GHz and 4 GB of RAM. It has a GPU with CUDA support that could be used to speed up image processing with OpenCV. The Jetson is also advertised to have 5 GHz gigabit WiFi connectivity.

The Jetson's main disadvantage over the RPi 3 were its added weight and greater cost. However, RAS has a Jetson TX1 and a corresponding breakout board, so those disadvantages are null.

The original design recommendation was to use a Jetson TX1. However the Orbitty carrier board originally intended to be used was damaged in an early prototype wiring harness when a ground wire broke loose and scraped along the bottom of the board. An attempt was made to swap the lightweight Orbitty Carrier Borad with the NVIDIA's standard development board. However this was found to be too heavy. The team did not have the ability to replace the Orbitty board and remain within budget. Thus a rapid switch mid semester was made to a Raspberry Pi 3. This was not the ideal single board computer but the team made do.

This switch in on-board computers impacted the wiring harness since the Orbitty Carrier board was originally intended to provide 5V for the low voltage side.

6.1.6 Altimeter

The altimeter needed to measure the height of the quadcopter from the ground. This was useful for position estimates on the z axis, which made the height controller possible.

In the system requirements it was specified a short and long range lidar were needed. The VL53L0X was the only feasible unit for the short range lidar and it was used succesfully. However, the long range lidar had two options.

The Lidar Lite V3 was a commonly used long range Lidar unit used in heavy lift drones. However, it was found to offer no advantages to its industry alternative the TFMini due to this project being restricted to indoor use. The TFMini's update rate was as fast as the Lidar Lite (100 Hz) and the maximum range was 12 m which was more than enough for an indoor environment. Also the TFMini was significantly cheaper ($40).

The TF Mini worked reasonably well. It reported height that was accurate down to 10 centimeters and it combined with the VL35L0X allowed height hold to function very well.

6.1.7 Optical Flow Sensor

An optical flow sensor has the ability to estimate velocity in the x and y direction. The PMW3901 interfaced with our AVR processor over a 2 MHz SPI bus. It is a very small, light, surface mountable part with very low power consumption.

In the case of optical flow sensors, support, documentation, NDAs, and price made the PMW3901 the only viable flow sensor on the market.

Once we more actively investigated the issue of estimating a velocity from the optical flow sensor, we found that the provided documentation was not as good as previously thought. It did not list the width in pixels of the window that the sensor itself observed, which meant that we had to find this number experimentally. We did find that another company had already used this sensor for a miniature drone called the "Crazyflie" and had found that the sensor had a pixel width of about 30 square pixels. In the end, all we had to do was verify those readings with measured distances. However, we did find a variance of about 5-7 pixels in our readings, which would result in an error of about 0.5 to 1 m/s. This resulted in decisively mediocre velocity

estimates. Overall, using an optical flow sensor was a good decision, but not a perfect one.

6.1.8 Flight Controller

The Pixhawk V1 running PX4 is the defacto flight controller for research UAV's. It is rather expensive at approximately $100. RAS and SERC both had Pixhawk's that could be used so this cost was void. Alternative flight controllers are also prohibitively expensive which leaves the Pixhawk as the only option.

For these reasons the Pixhawk was the de-facto choice. However, the team ran out of time to implement support for interfacing PX4 with the IARC flight stack used for the project. This was recognized early in the semester, thus there was time to switch to a spare flight controller board the RAS had on hand that was compatible with the IARC software stack. In the end, a Seriously Pro Racing F4 EVO flight controller was used.

6.1.9 Battery

RAS had a 3S 40C 5000 mAh battery that was used for powering all the rotors. LiPos that are appropriate for this drone range from $40 - $60. In order to save money the RAS LiPo was used. Eventually two more 3S 5000 mAh Gens Ace batteries were purchased so that the team could use batteries and charge at the same time while testing.

6.1.10 Downward Facing Camera

The only requirement for the bottom facing camera was that it existed. Originally it was the intention to use an Intel R200 from RAS's supply. However, when the on-board computer was switched from a Jetson TX1 to a Raspberry Pi 3 this was no longer possible since the R200 requires USB3 in order to function. A Pi Camera could have been used to replace the R200 however at that point in the design process it became clear that weight savings were an issue. Since the bottom facing requirement was not used by any subsystem directly it was decided that no camera would attached.

6.2 Power Distribution Board

6.2.1 Battery Interface

The power source for the system was split into two separate LiPo batteries. One high capacity 2 cell LiPo was used to provide power to the motors, and a second 2 cell LiPo provided power to the control system. The 3 cell LiPo had a voltage range of 11.1V - 12.6V. and the 2 cell LiPo had a voltage range of 7.4V - 8.4V.

A 3 cell and 2 cell LiPo were primarily chosen because the voltage ranges were large enough that they could be converted to 5.0V and 3.3V on board via the LDO and switching regulator. A larger battery with more cells was not used due to the extra weight that it would add to the system, which would decrease the overall thrust of the system.

6.2.1.1 Motor Power Interface

The battery interface provided a direct connection to the battery packs of the system. Since each motor had a maximum current draw of 30A, these connections had to be properly specified. In the hobby electronics world, common battery connector types include XT60, EC3, EC5, and bullet connectors. The system used XT60 connectors to interface to the motors and a deans connector to interface to each of the LiPo batteries.

XT60 connectors were chosen to interface to the motors mainly because of their suitable current ratings. The maximum current draw of each motor was specified to be 30 Amps, and the XT60 connector is rated up to 60 amps, making it a suitable connector for this use case. Also, an existing stock of XT60 connectors available from the RAS club room were also available. Other connectors such as the EC3, EC5, and bullet connectors would have added additional cost to the project, so they were not considered for the motor power interface.

Figure 6: XT60 Connector [2]

A deans connector was used for the connection battery because it would differentiate the connection from all the other motor connectors. Since a different connector was used, it would be obvious within the system which connection plugged into the LiPo batteries, which made system integration easier. A stock of deans connectors were also available in the RAS room, so it was used over other connectors such as EC3, EC5, and bullet connectors due to costs constraints.

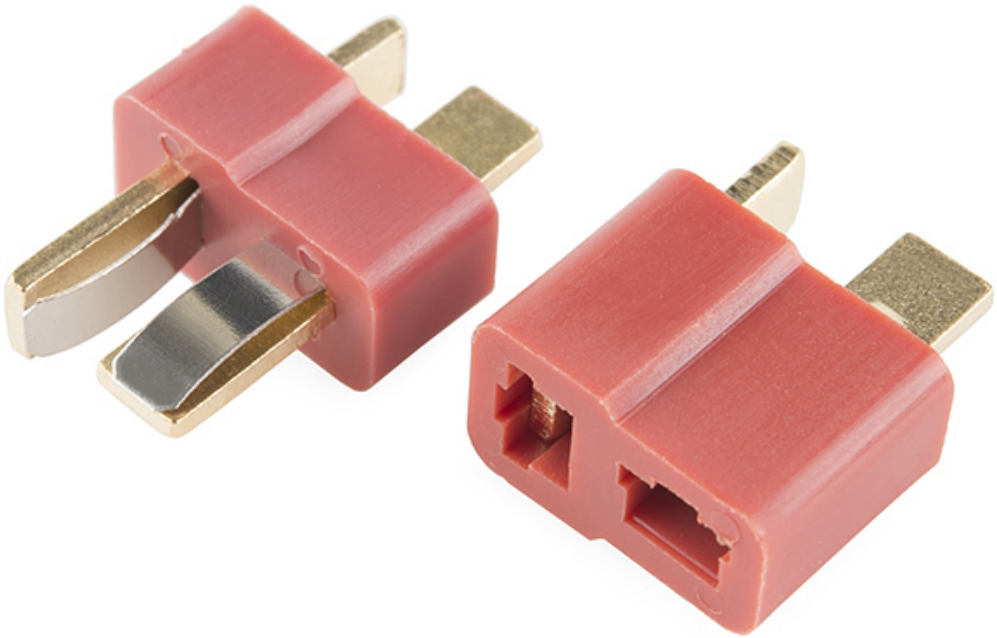

Figure 7: Deans Connector [3]

Another factor that was considered in the battery interface to the motors is the copper weights and trace widths of the connections to the motors. The maximum current draw that each motor could pull individually is 30A, so the traces were to be specified accordingly to ensure that the traces do not burn off. Using the advanced circuits trace width calculator, A copper weight of 2 oz/sq. ft. and a trace width of roughly 229 thou was required to allow up to 30 amps of current with a maximum temperature of 85° C [17].

The ESCs on the system that drive the motors will also require a 5V input. This 5V supply was be provided by a DC-DC buck converter. The LM2596 switching regulator was used to provide this 5V input. The LM2596 has a

maximum input voltage of up to 40V, so the supply will function over the range of a 6 cell LiPo. This supply was available in a TO-220, so it could be easily soldered onto the PCB.

The MAX17557 synchronous step down converter was also evaluated for use within the system, but was not chosen due to its package size and pin out. The chip was only available in a TQFN package, which would have been extremely difficult to solder. Also, the package had a large ground pad located under the package of the IC, which would not be easily soldered, which could lead to issues due to bad grounding.

6.2.1.2 Control System Power Interface

The low voltage control system was powered off of the smaller capacity 2 cell LiPo. A Common Power Module was used to interface between the 2 cell battery and the Raspberry Pi. The Common Power Module provided a stable 5.37V output with current capabilities of up to 2.25 amps [18]. This common power module was used instead of a dedicated, on board regulator because the on board computer changed half way through the prototyping stage. Before, when a Jetson TX1 was used, it could interface directly to the LiPo, but this was not the case with the Raspberry Pi.

Since a Raspberry Pi was used for the on board computer within the design, its own 5V regulator was used to produce 5V to supply to control side of the power distribution board.

A Jetson TX1 with an Orbitty Carrier Board was also considered to be used in the design for the on board computer. The input voltage range of the Orbitty Carrier Board spans from 9V to 15V, so it could have been connected directly to the 2 cell LiPo [19]. However, this was ultimately not used in the design due to the large weight of the Jetson carrier board along with the higher cost relative to the Raspberry Pi.

The flight controller used was a cleanflight. The cleanflight also was provided power by the common power module, so nothing was required within the control system power interface for the flight controller.

Originally, a Pixhawk flight controller, which requires an input voltage of 4.1V to 5.7V [20], was being considered. However, the Pixhawk did not provide a simple way to interface with 8 motors. Because of time constraints of the project, the cleanflight was used instead.

6.2.2 Voltage Monitor Circuits

The voltage levels of both LiPo batteries were monitored within the system. The common power module that plugs into the low capacity LiPo for the controller subsection of the system included a voltage monitoring signal that. This module was sufficient for monitoring the voltage of the 2 cell battery.

For the higher capacity battery, the battery voltage was monitored by one of the ATMEGA328P's ADCs. Since the battery voltage was susceptible to high transients from current draws from the motors, the ACPL-C87B analog isolator was used to protect the microcontroller from the noise produced by the propellor motors. For this chip, the 3 cell LiPo voltage had to be stepped down using a resistor divider to limit the maximum voltage to 2.0V.

An external ADC that communicated via I2C was also considered. In a similar fashion, a resistor voltage divider would have been used to step the voltage down to suitable input voltage levels to the ADC. The ADS1015QDGSRQ1, a 4 channel, 12 bit ADC was considered. In the end, this option was not considered since it would still require a digital isolator with a high enough data rate to keep up with the ADC, which unnecessarily increased the complexity of the system without any added benefits.

6.2.3 Current Monitor Circuits

The current consumption of each individual motor was monitored using Hall Effect current sensor ICs. The ACS781LLRTR-100U hall effect current sensors were used. This sensor allowed unidirectional current monitoring of up to 100 Amps, with a nominal supply voltage of 3.3V. Despite this, the sensors were supplied power from the 5V DC-DC buck converter. The analog voltage output was then be sampled by the 12 bit ADC after passing through an MC14051B analog multiplexer and communicated to the on board computer.

The ACS780LLRTR-100B was considered and originally selected for the current monitoring circuit. These sensors were able to bidirectionally monitor currents up to 100 amps with nominal voltages of 5.0V. These hall effect sensors were more suitable for our application, but due to an error in ordering from Digikey, the ACS781 ICs were used. This mistake ultimately did not cause to a loss of functionality, so no changes had to be made in the design.

The ACS758LLRTR-100B was also considered. These sensors can sample up to 50 amps of current, which is more suitable for the maximum current draw of 30A of each of the motors. This would give more resolution to the output analog voltage of the sensors overall for more accurate readings. However, these chips were not in stock anywhere online, so the 100 amp versions were used instead. In future designs, these chips would be more suitable.

6.2.4 Electrical-Isolation

An electrical isolation boundary between the high voltage sections of the system which include the motors and ESCs, and the low voltage sections will be included in the design. A system diagram of the isolation boundary can be seen in figure 8.

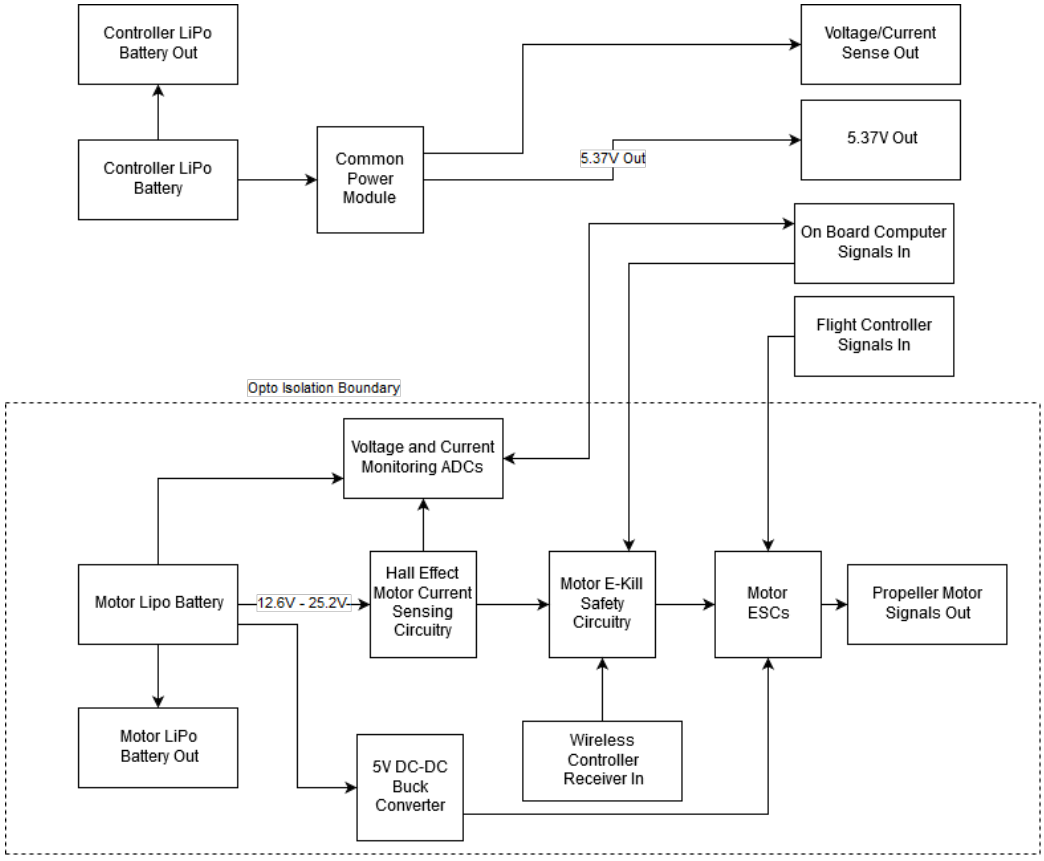

Figure 8: Electrical Isolation System Diagram

Since the ATMEGA's ADCs reading the current and voltage levels will communicate with components on the motor power side of the isolation boundary, some form of digital or optical isolation was required. For the current readings, since the sensors were purely magnetic as hall effect sensors, no isolation was required. As mentioned before, The ACPL-C87B was used for the isolation of the analog signal of the battery level.

Before the analog isolator was chosen, a special purpose, bi-directional isolator was considered for use with an external I2C ADC. The TI ISO154X IC was considered since this chip could operate off the 5V line provided by the DC-DC buck converter, and it could provide two channels for the SCL and SDA lines of the ADC. The chip is also a 8SOIC package, which would cause no issues during assembly. However, this was not required since an external ADC was not used.

For the control signals interfacing between the flight control and ESCs, and the on board computer and the E-Kill circuitry, a standard digital isolation IC was used. These signals were all digital signals operating at 50 Hz frequencies. The ADuM1410 4 channel digital isolator was used for these signals. This IC was chosen due to its number of channels, which allowed for a separate chip for the main motor and side motor controls.

The Silicon Labs SI8660BA-B-IS1R IC was also considered to be used to isolate the PWM signals between the flight controller and the ESCs, and the on board computer and the E-Kill circuitry. This chip was 6 channel 16SOIC package. however, a multiple of 6 channels was not required, so this chip was ultimately not used in the design.

6.2.5 E-Kill Circuit

The E-Kill circuit that was implemented by having high powered FETs in line with the power signals of the motors, controlled by a signal from the wireless controller receiver input . PSMN1R0-40YLD N Channel MOSFETs were used to tie the negative terminal of each motor to ground through the source and drain. These FETs in particular were chosen due to their high Rds on of 1.4m Ohms, and their high current capabilities of 280 amps. The LFPAK56 package of these FETs also made assembly easy even though they were surface mount parts. This gave the E-Kill the ability to be controlled remotely by a wireless receiver to safely disable power to the motors while remaining in a suitable temperature range with minimal power losses.

Other FET options that were not chosen included Vishay SQM40014EMGE3

and Infineon IPB180N04S4H0ATMA1 FETs. These FETs were not chosen due to their higher prices, lower maximum currents, and packages. The Vishay part has a max Id of 200 A while the Infineon part has a max Id of 180A. The Vishay part also has an Rds on of 1.1m Ohm while the Vishay part has an Rds on of 1mOhm. These FETs were both TO263-7-3 packages, which would have been harder to solder onto the board by hand. These chips were also $3.10 and $3.09 respectively, which In the signal processing literature, the use of non-causal (symmetric) filters is commonplace, and the exponential window function is broadly used in this fashion, but a different terminology is used: exponential smoothing is equivalent to a first-order Infinite Impulse Response or IIR filter and moving average is equivalent to a Finite Impulse Response or FIR filter with equal weighting factors.

were significantly more expensive compared to the $2.21 price point of the PSMN1R0-40YLD FETs.

6.3 Localization

To obtain a relative location of the UAV, and a ground truth location of the UAV several techniques were employed. Local estimation was performed using sensor fusion, and ground truth estimations were taken with a ground station camera.

6.3.1 Sensor Fusion

An extended Kalman filter previously used on the Pitt RAS 2017 IARC drone was used to accomplish sensor fusion and state estimation. This was done in order to save time and effort, as the algorithm worked in real time on the Raspberry Pi. The program is not without issues however, and suffers from a non-severe memory leak after long periods of usage. This problem has yet to be solved in the upstream, but did not hinder development to a large degree.

6.3.2 Motion Tracking

6.3.2.1 Camera

The camera chosen for the ground truth motion capture was picked with

several qualifications in mind. The camera had to have a depth image available, work well enough with the ROS stack, and have a high enough FOV and pixel count such that a marker could be spotted easily.

Originally, the Intel R200 and the Microsoft Kinect were to be used in conjunction, but the R200 proved more complicated to interface, so the Kinect was selected as the main camera.

However, it was found that the Kinect does not have manual camera settings (exposure, white balance, etc.), but the R200 does. This made the Kinect less usable for more advanced algorithms that leverage different camera settings to make certain markers easier to track. These algorithms were never used, however, so the Kinect remained the final camera for the prototype.

6.3.2.2 Markers

The choice of markers for tracking the position of the UAV came down to whatever was the most convenient for a given algorithm and camera pairing. Both the R200 and the Kinect have the capability of doing markerless tracking; using a prototypical 3D model, and determining a matched objects 3D location in space based off the of model. This method would have been the easiest in terms of markers used–there are no markers–but the algorithm had mutiple issues in its implementation.

The second choice of markers tried were colored foam balls. While these balls were easier to track by some algorithms, and could be easily differentiated from one another, their size made it difficult to track them at a distance, limiting their usability. Additionally, the colors of the balls were sensitive to room light, making tracking them variant with the lighting conditions of a room.

Another iteration used medium sized white balls, which worked marginally well. Eventually it was determined that a large white foam ball had the best balance between weight and visibility at all distances, and was used for the positional marker in the final prototype.

Another marker prototype was researched, but not finished. The markers in question were high power RGB LEDs diffused by a "ping pong" ball housing. This marker had many of the advantages of the original RGB foam ball markers, but the exposure on a camera could be lowered. In this way only bright light sources–such as the LED markers–would be greatly visible. In initial tests these markers worked adequately, but were not used in final tests

due to time constraints.

6.3.2.3 Algorithm

The initial choice of algorithm was a markerless tracking algorithm based on the AR Toolkit. The software worked by registering a known prototype 3D model to an object in an image. This algorithm seemed simple at first, but proved to be difficult to implement as it needed a specially made 3D model of the UAV, and required calibration on start-up. Likewise, tracking even simple objects such as boxes was slow and somewhat noisy.

The next choice of algorithm investigated was blob detection using specific HSV values. This algorithm searched for and labelled blobs in an image with a given HSV value. While this method was fast, it was prone to errors due to its strict thresholds in HSV values, and noise in the image. Dark and light spots in an image would often be classified as a marker, and markers that are too bright or dark would not be in the HSV range to be classified.

The final prototype algorithm sought to do pure positional tracking by doing routine background subtraction and labeling circles in the binary foreground image. Initially circle labelling was done using the Hough transform, but this proved to be too slow for real-time labelling, so a custom algorithm was written. This method was the most stable, and had minimal tracking issues.

An additional algorithm was developed near the end of the term which combined the flexibility of the color based design with the stability of the binary circle detection algorithm. This algorithm did background subtraction, and then ANDed the binary foreground image with a filtered HSV image (filtered in ranges for red, yellow, blue, green). It then labelled found circles as red, yellow, blue, or green based on which filter was used. With three different colored markers a plane could be reconstructed which tracks the pose and position of the UAV. While promising, this algorithm was not thoroughly tested enough to be included in the final design.

6.4 Controller

The Controller was largely built on top of the last year and a half work accomplished by the RAS IARC Team. Thus the Design Concept of the controller was largely fixed as it was be built on top of existing controllers. However two

distinct design concepts were proposed for the large scale controller design that could be built on the existing software.

6.4.1 Single Axis Control Model

PID Control with a static thrust model could have been used to form 3 single input single output controllers that controlled each of the position axis and the corresponding rotors. This was already demonstrated to work for vertical height control in an indoor environment by the RAS IARC project. This control system would have been "copy and pasted" to the other 2 axis to achieve positional control.

Figure 9 shows the basic layout of such a controller. The side rotor controllers are not shown for simplicity. While the control system takes a position setpoint as a target, it is a velocity controller as the PID controller acts on the velocity term.

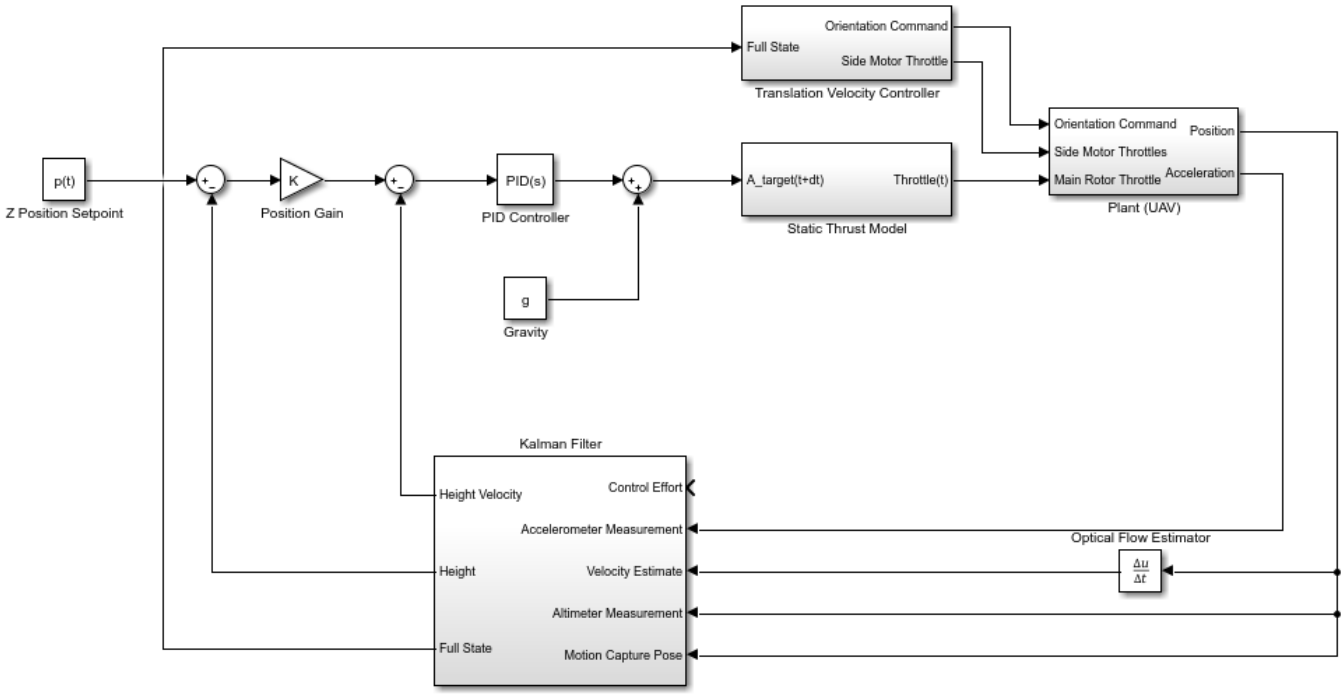

Figure 9: A PID Controller that uses a static thrust model to achieve height control

A more complicated PID Control using Feed Forward and Time Based Thrust Model was also proposed. It was designed as an extension of the previous controller. The time based model allowed the rotors to be treated as a second order system as long as the control loop was run one timestep ahead of real-time. Essentially, the time based model allowed the spin up time of rotors to be anticipated. The development of the time-based thrust model also resulted in a quantified understanding of the maximum jerk limits and maximum thrust for each axis. Figure 10 pictures such a controller being used for the vertical axis with the side axis controller abstracted from view.

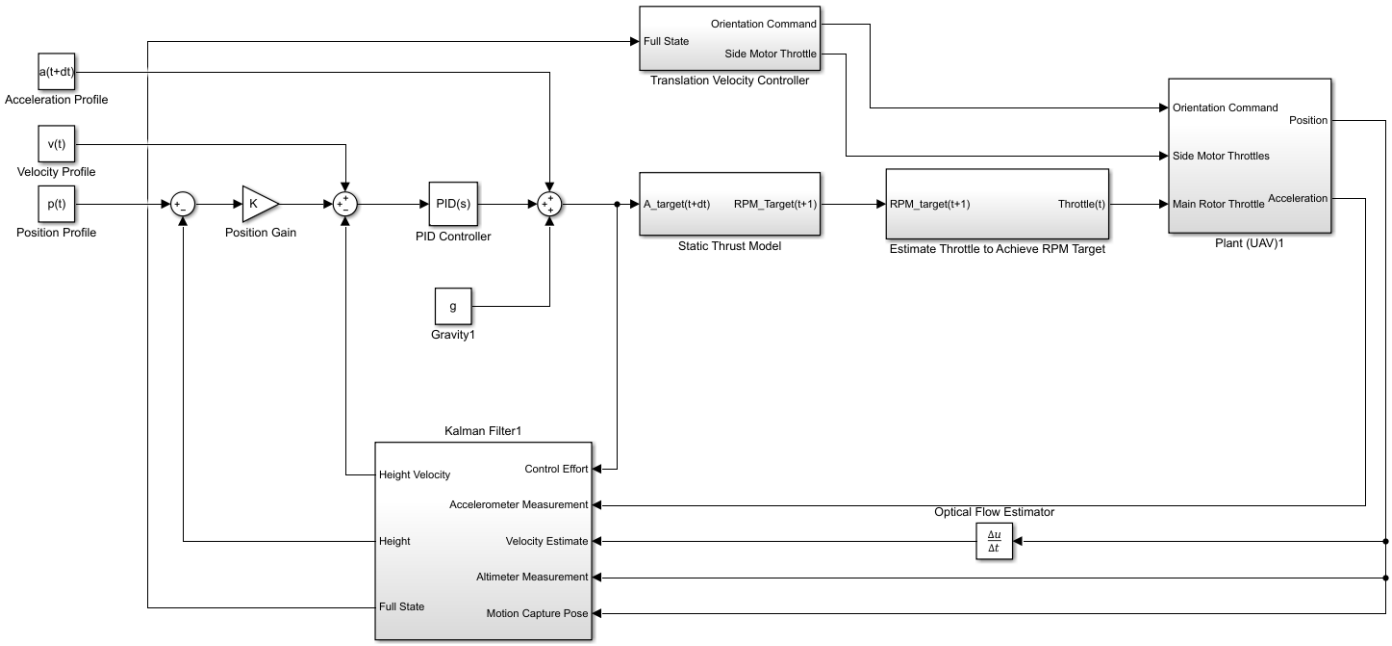

Figure 10: A PID Controller with Feed Forward that uses a time based thrust model to achieve control

The primary advantage of this controller was the ability to anticipate rotor response time. It was necessary that the IARC stack be extended to project motion plans forward in order to use this model. However this re-work was fairly easy to do. By doing this the target position, velocity, and acceleration could be known ahead of time. And the time based thrust model could be taken advantage of. Without this feature the controller would have to be perfectly reactive to error created by rotor response time.

When comparing the two proposed controller design the obvious difference

was complexity. If the project had started with no prior work, building the PID Control with Static Thrust Model design would be more than enough work. However, because this design has already been implemented for the vertical axis, in a modular way, by the RAS IARC project, it would have been trivial to re-implement this controller for multiple axis.

With prior work in mind, the superior performance of Feed Forward PID Control with Time Based Thrust Model was considered to be both desirable and attainable within the scope of the project.

It should be noted that both controllers required a thrust stand and the measuring equipment to develop a thrust model. RAS had the materials required for this so this was not anticipated to be a problem.

In the end the controller design shown in Figure 10 was used with one modification for the height controller and for the side rotor controller. For the height controller, the PID control was set to run directly on the error between the desired height and the target height. Thus, the velocity setpoints were not used for height at all. For the side rotor controller, the position setpoints were not used at all because the position estimates from the kalman filter were not bounded by an absolute position measurement. Thuse the XY position profile estimates were not used at all.

6.4.2 Human Input

One way to receive human input was to interpret the RC Channels from the flight controller and accept them as control inputs. This would have had similar latency to manual control and the human pilot did not need to learn to use a new controller. This setup allowed the same controller to be used taking over manual control at the Pixhawk level and flying in velocity controlled mode using this projects controller.

A gaming joystick could also have been used to provide input to the drone. This would have been simpler to integrate into the controller stack since a USB gamepad could be connected to any PC or Laptop on the same network as the On-board computer and communication could occur through ROS. Additionally, the controller used for manual takeover and for human input would have been separate which would have allowed for a less complex automatic takeover hierarchy.

The original design recommendation was to use the RC Control method as it was expected to be significantly easier to implement from a communica-

tions perspective because an interface with a USB gamepad would not have needed to be written. However, because a Pixhawk was not used as the flight controller this turned out to be quite difficult. So a wireless XBOX 360 controller was used instead.

6.4.3 Autonomous Takeoff and Landing Controllers

Autonomous takeoff and landing was expected as easily achieved using a special purpose controller made by RAS for the purpose. One hiccup is that RAS began to rewrite the height controller extensively during the duration of this project. This required quite a bit of effort from the senior design team to support. In the end, this is what caused the height controller to end up using position as input instead of velocity. Nonetheless, this design concept worked rather smoothly and did not need to be altered.

6.4.4 Thrust Model

The thrust model was not previously discussed in the Conceptual Design however it became one of the biggest parts of this project. A brief description of the design concept for the thrust model will follow.

It was recognized early on in the project that the non-linear 3rd order system formed by a propeller and motor could be made into a linear 2nd order system with the appropriate thrust model and by running a controller ahead of real time. The Thrust Model was then designed based on inspiration from the ideal motor equations, verification from physical testing with a thrust test stand, and resulted in a series of python scripts that could automatically output the constants for the thrust model. Code was also written to interpret and test a thrust model on the test stand to verify the performance improvement of the thrust model.

The resulting thrust models revealed that having a rated jerk limit would be impractical due to the non-linearity in the resulting thrust model. However acceleration limits could still be enforced easily.

6.4.5 Motion Profile Generator

The Motion Profile Generator, like the Thrust Model, was not previously rigorously discussed in the Conceptual Design however it became an integral part of the controller. The Motion Profile Generator generated discrete

position, velocity, and acceleration targets ahead of real time that the controller could track. It enforced the acceleration limits as required in the requirements.

During the mathematical derivation of this generator an attempt was made to limit the jerk as well as the original requirements stated. However, it was proven that limited a state two derivatives down was a none trivial planning task best handled by a motion planner. Thus it was decided to drop the requirement that jerk limits be respected.

7 Team

Long Vo, Liam Berti, Ritesh Misra, and Levi Burner will be the members of this Senior Design team. They each have 2-3 years of hands-on experience designing embedded systems for robotic and industrial systems.

7.1 Long Vo

Long is a senior electrical engineering student. Along with the standard ECE Pitt curriculum, his electives include electromagnetics, embedded design, and digital and analog filter design. He has been a member of the Robotics and Automation Society for 3 years. He has also been employed by Rockwell Automation as a hardware engineering co-op for 2 years. At his time at Rockwell, he has gained experience developing, verifying, and testing industrial control systems. He brings experience in analog and digital circuit design, isolated and safety system design, and PCB design experience.

Long was responsible for designing the power distribution board. This included the design of the circuitry, the PCB design, and the soldering of components for final assembly.

7.2 Liam Berti

Liam is a senior Electrical Engineering student with a minor in Computer Science, and pursuing a Signals and Systems concentration. He has been a member of the Robotics and Automation Society at Pitt since 2015, and has worked on several successful projects. Additionally, Liam has experience with machine learning and image processing from research that he has done with ECE and the LRDC. He brings experience with image processing using traditional techniques and machine learning, system software and embedded programming, digital and analog electronic design, mobile power systems implementation, PCB design, and GNU/Linux System administration.

Liam was responsible for system localization and system monitoring using off-board cameras. He also assisted with system programming, and electronics when necessary.

7.3 Ritesh Misra

Ritesh Misra is a senior Computer Engineering student. He has been a member of the Robotics and Automation Society for 2 years. He brings experience with low-level embedded systems from co-oping at Rockwell, image processing, systems software, software engineering, and ROS.

Ritesh was responsible for sensor integration and writing C++ for the Pi to estimate velocity using optical flow sensor readings.

7.4 Levi Burner

Levi is a Senior Electrical Engineering student pursuing a Signals and Communications concentration. He has been a project lead in the Robotics and Automation Society of the University of Pittsburgh since 2015 and has lead multiple robotics projects to successful completion. Additionally, he has worked at Rockwell Automation in their Embedded Software Engineering department. He brings experience with UAV controls, multiple flight controller platforms, localization of autonomous drones, ROS, software engineering, embedded systems, embedded Linux, robot power systems, PCB design, and image processing.

Levi assumed responsibility of the Control System and interfacing with the flight controller. He also handled constructing the aerial platform.

8 Final Prototype

8.1 Aerial Platform

8.1.1 Frame

An X525 frame was used as the based. This provided everything required for a standard quad rotor from the start. A picture of the original frame used can be seen in Figure 11.

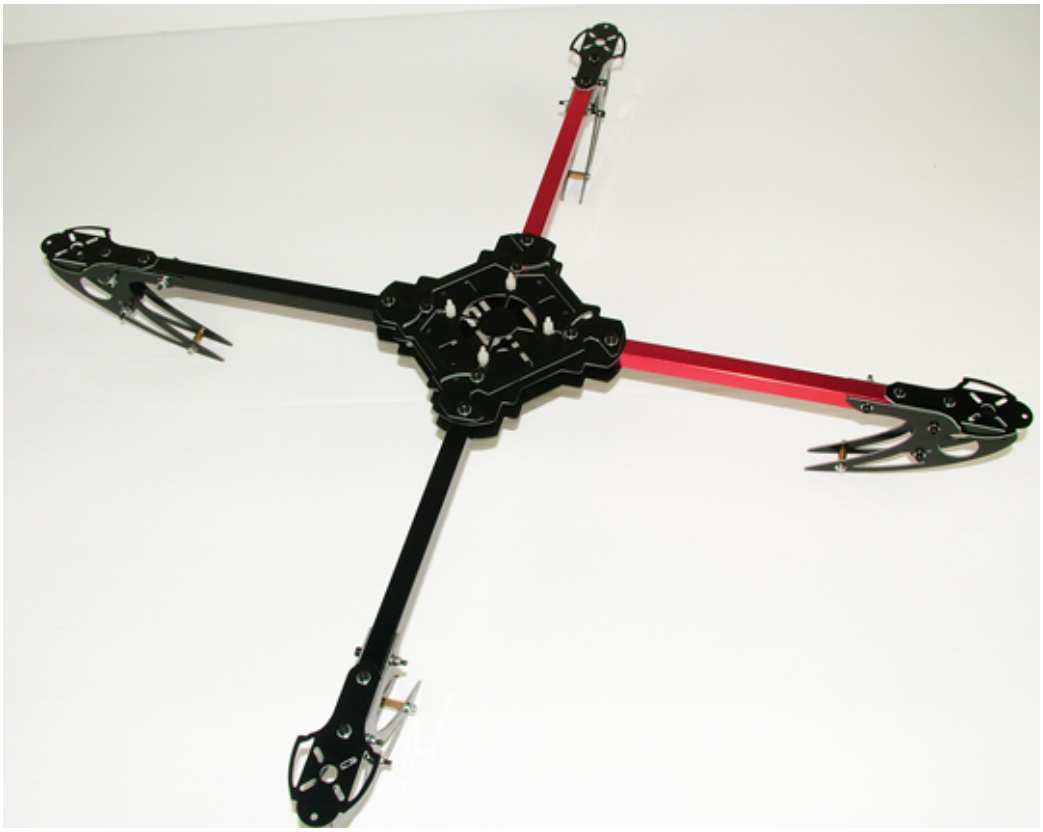

Figure 11: Original X525 Frame

Modifications were made to the center fiberglass plates that allowed the attachment of the side rotor assemblies. A model of the side rotor assembly is shown in Figure 12.

Figure 12: Side rotor mechanical assembly

The side rotor assembly used 3D Printed brackets (pictured in black) that sandwiched on a poltruded carbon fiber tube. Sandwiching was necessary because holes cannot be placed in poltruded carbon fiber without completely compromising the tube. 3M Very High Bond (VHB) tape (commonly used for attaching skyscraper windows) was layered between the 3D Printed bracket and the carbon fiber tube. This prevented the tube from slipping out of or rotating in the 3D printed brackets.

The motor attachment bracket was adjustable up and down with a series of screw holes. This allowed the height of the side motors to be adjusted easily after assembly so that they aligned with the center of gravity. This minimized pitching and rolling when the side rotors came on.

8.1.2 Electronics mounting

The flight controller, and power distribution board, were mounted using 3D printed brackets in the center of the frame as seen Figure 13.

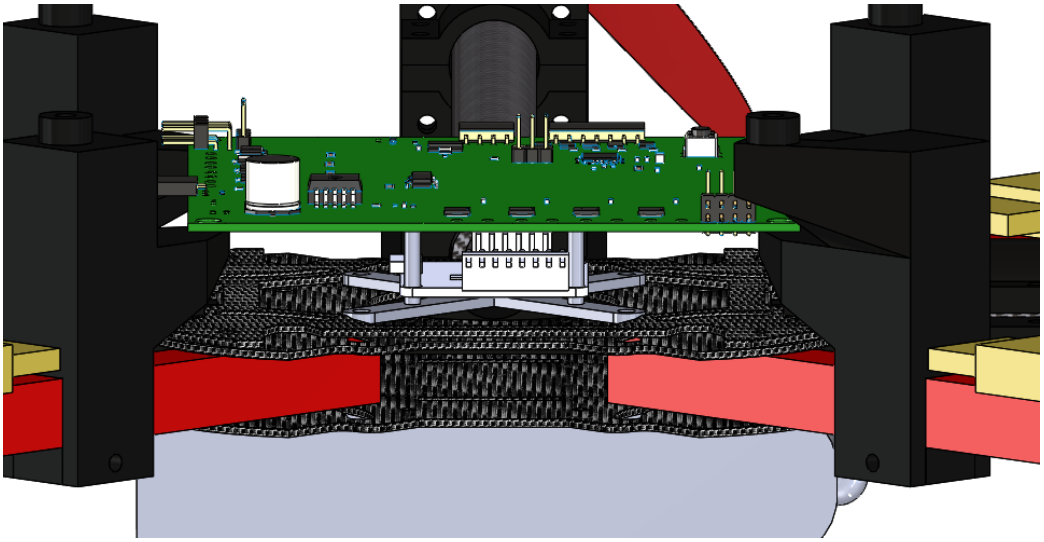

Figure 13: Center stack showing the flight controller and power distribution board

The flight controller was placed to so that it was as near to the center of gravity as possible. This was so that it would be as close to the center of rotation as possible. This prevented the accelerometer measurements from being influenced by rotation and translation and allowed for more accurate velocity estimates.

Because the switch to Raspberry Pi was not intended, an elegant mounting solution was not design for it. It was simply attached with velcro to the top of the power distribution board.

The TFmini, VL53L0X short range altimeter, and PMW3901 flow board were attached on the bottom of a the drone using a combination of VHB tape and zip ties as can be seen in Figure 14.

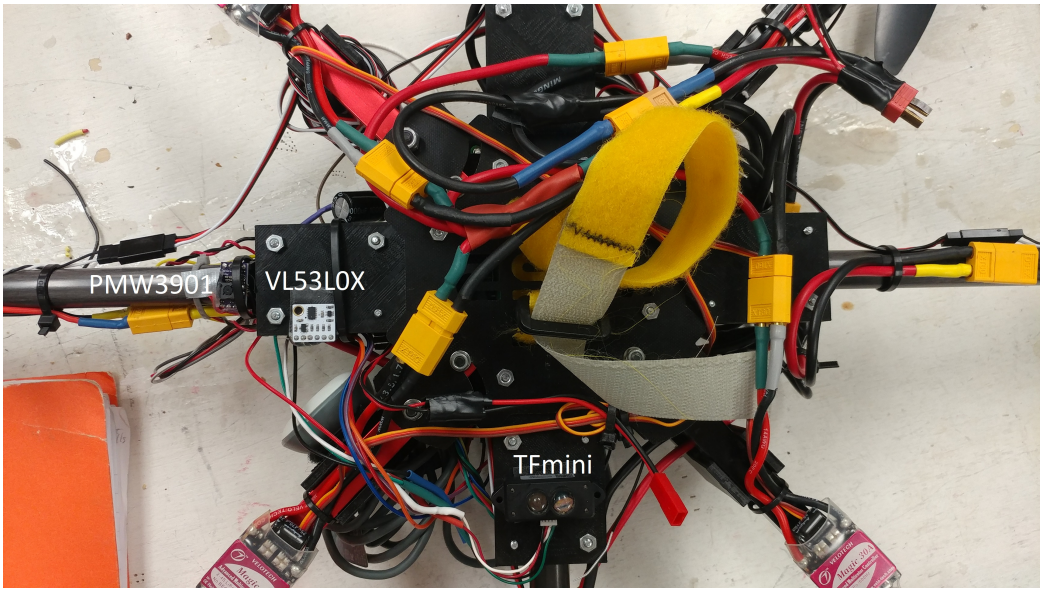

Figure 14: Placement of altimeters and flow sensor on the bottom of the drone

The high and low voltage batteries were mounted with a single velcro stap as can be seen in Figure 15. Because of the relatively small size of the low voltage electronic battery compared to the motor battery this was possible.

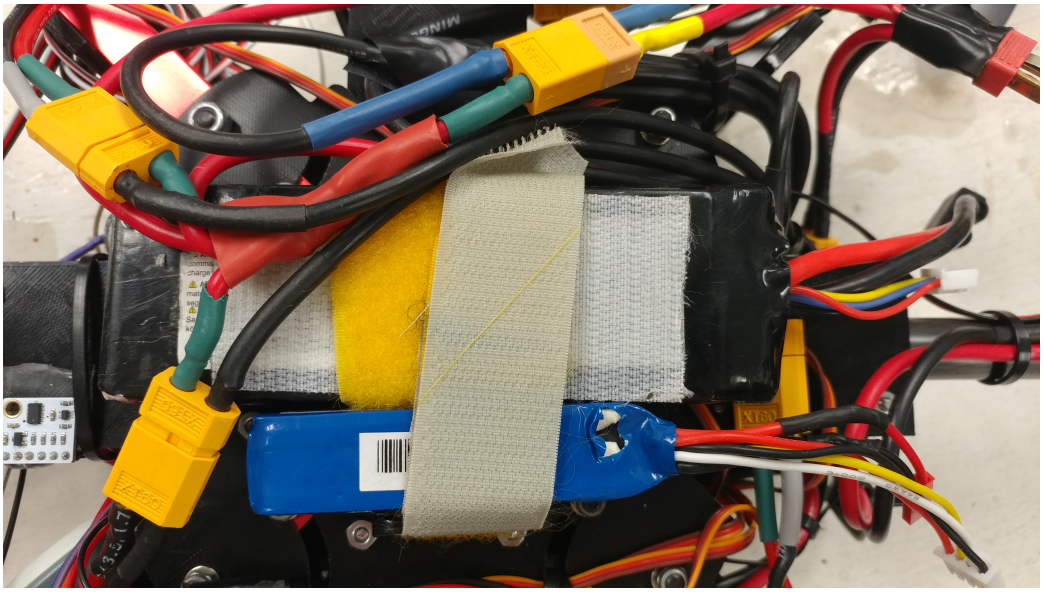

Figure 15: Mounting of batteries on the bottom of the drone using a single velcro strap

Miscelaneous other electronics were mounted as follows. The XBOX 360 receive was mounted with velcro to one of the 3D printed power distribution board mounts. The ekill radio receiver was mounted with zip ties to one of the original X525 arms. All ESC's were zip tied to the frame arm corresponding to their motor. The side ESC's were wrapped in electrical tape so that they would not conduct but coming into contact with the poltruded carbon fiber arm. Finally, the flight controller radio receiver was attached with VHB to the center frame.

8.1.3 Propulsion

The main motors were Sunnysky X2212 980 kV motors. These motors came originally with the X525 frame donated by RAS. They were paired with Velotech Magic 30A ESC's and genertic 10x4.5 propellers. These components were very cheap and very outdated compared to modern UAV technology. However, they held up and were free which allowed the project remain mostly within budget.

The side rotors as were EMAX RS2205's paired with EMAX Bullet 30A ESC's and Master Airscrew 6x4.5 inch propellers. They are pictured in Figure 16. A 1000 uF Panasonic low ESR cap was used to filter the supply

voltage to the bullet ESC's since their current draw could spike from 0-30A within a few hundred milliseconds.

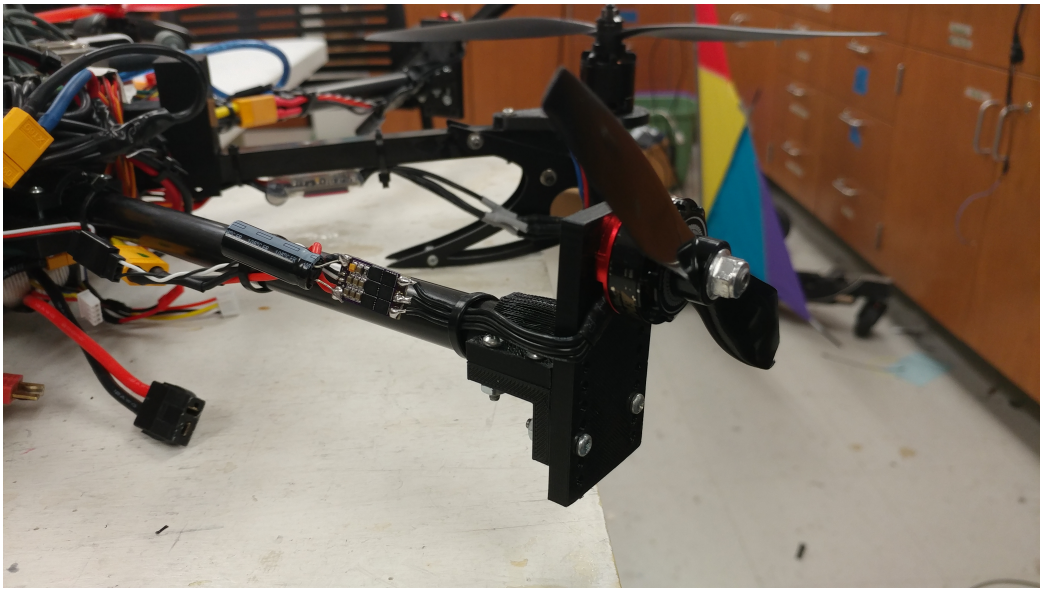

Figure 16: Side motor, ESC, and propeller assembled and mounted on the drone

8.1.4 Final Assembly

The final assembled prototype is pictured in Figure 17. The drone weighed approximately 2.3 kg in this configuration and could hover for approximately 7 minutes. The main motors could provide approximately 3.2 kg of thrust when fully loaded. This was lower than the peak thrust recorded in thrust modeling due to the increased voltage drop of the battery when 4 motors were being run instead of one as tested on the thrust test stand. The side rotors could provide 0.8 kg of thrust each. This allowed for a peak sideways acceleration of approximately 0.3 g which was one half of the targeted 0.6 g of acceleration. This was due to the frame being heavier than intended and the fact that 3S batteries were used to save money instead of 4S batteries.

Figure 17: Final assembly of the aerial platform

8.2 Sensor Integration

8.2.1 Architecture

We used an asynchronous update loop to continually cheIn the signal processing literature, the use of non-causal (symmetric) filters is commonplace, and the exponential window function is broadly used in this fashion, but a different terminology is used: exponential smoothing is equivalent to a first-order Infinite Impulse Response or IIR filter and moving average is equivalent to a Finite Impulse Response or FIR filter with equal weighting factors.

ck our sensors for updates to send to the main computer. Since we know our update rates for each of our sensors, we can check that a certain amount of time has passed before we actually check the registers/buffers for new data.

8.2.2 Interrupt Issues

Using software serial with the TFMini caused issues with a servo library that used the same hardware interrupt as the software serial library, so we were forced to write to individual registers instead of using the servo library,

since we could not sacrifice long range distance measurements. This was accomplished relatively easily with PORTC statements. However, not having access to any hardware interrupts slightly hurt our update rates for the rest of our sensors.

8.2.3 Velocity Estimates

The optical flow sensor continually sends us a change in pixels in the x and y direction between the current time and the last time that we checked it. Since it gives us some indication of how much we move parallel to the ground plane, we can use it to estimate our velocity. We originally tried to include corrections for change in pitch and roll that we thought would be important for obtaining good velocity estimates. However, upon comparing the raw velocity estimate that only used the drone's current height and optical flow measurements with the corrected velocity estimates, we found that the corrections often made the velocity estimates more inaccurate. We decided that it would be better to implement a digital filter for both the x and y velocity estimates to prevent the unwarranted zero crossings that we were getting. We used a 5th order Kaiser window to obtain a linear phase dropoff so as to not lag our measurements significantly.

8.2.4 ESC Updates

The AVR processor is connected to the Pi over serial. A ROS package called rosserial reads one of the topics output by low level motion and sends the commanded motor PWM over the serial connection. After the AVR processor receives these commands, it then runs a function that sets local variables equal to the commanded PWM duty cycles. In the asynchronous loop, the PWM cycles are sent out at the very end (at a point where the other sensors will still have no new data available if they were checked in that same loop).

8.3 Power Distribution Board

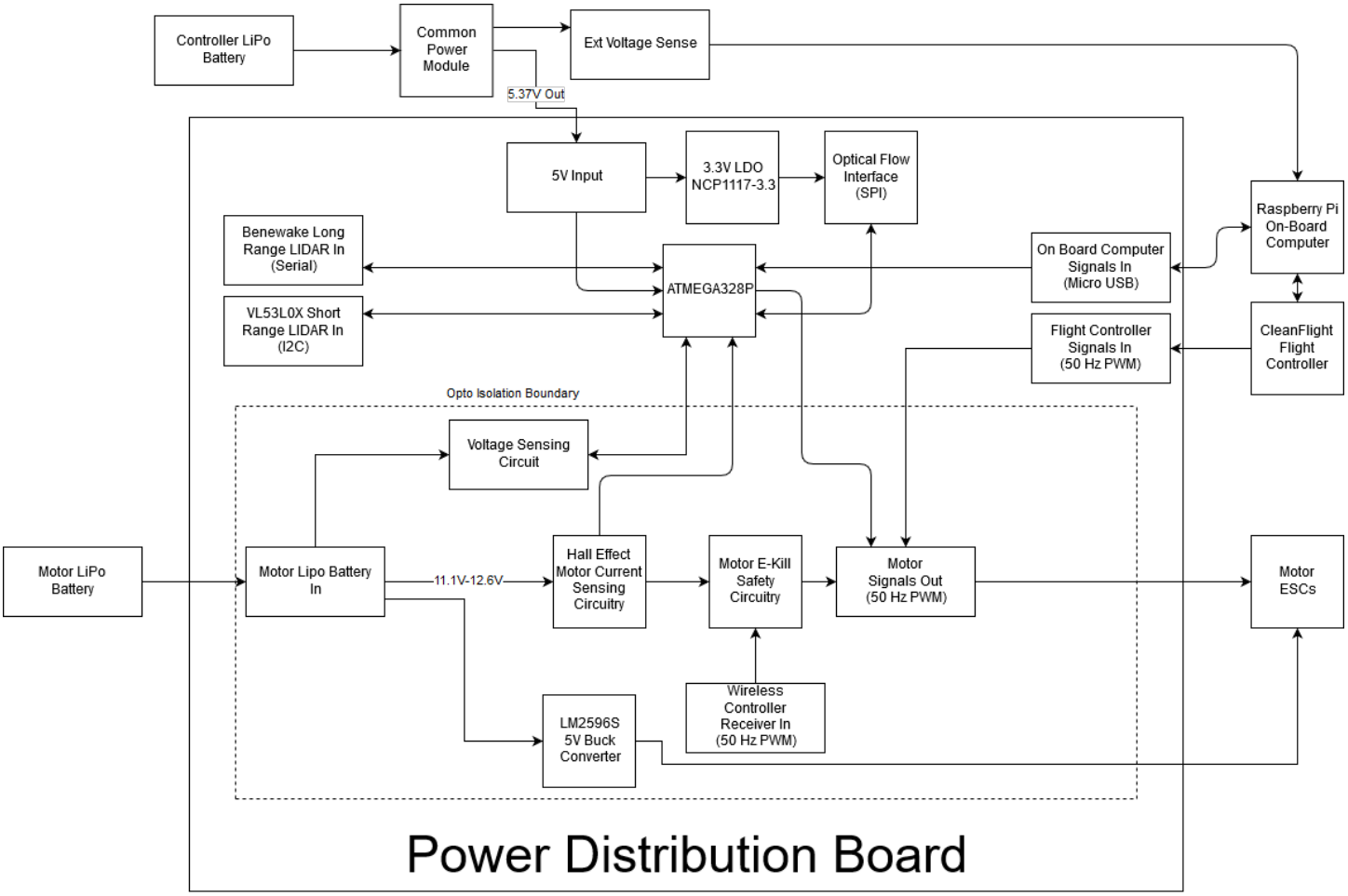

Figure 18: Finalized System Diagram of Power Distribution Board

A finalized block diagram of the power distribution board can be seen above. The board interfaced with both batteries, the on board computer, the flight controller, the motors, and the sensors which include both long range and short range LIDARs and the optical flow board. The board itself was split into two separate isolation regions. These regions were separated by digital and analog isolation ICs.

8.3.1 Low Voltage Isolation Region

The low voltage isolation region included the interface between the sensors and computing subsystems of the UAV, which included the ATMEGA328P microcontroller, Raspberry Pi on board computer, and CleanFlight flight controller. This part of the system was isolated from the high voltage isolation region so that the sensitive components would not be affected by the large amounts of noise generated by the motors.

8.3.1.1 5V Input Circuitry

The 5V input interface for the low voltage isolation included a 1725656 phoenix terminal block that allowed input voltage wires to be securely screwed into the board. This ensured that the hot wires would not come undone during aerial operation of the drone. On the positive input terminal of the connector, a schottky diode was included to protect the region from reverse polarity inputs. The input circuitry also included a status LED to indicate power to the board. The majority of components within the low voltage region used this protected input voltage for Vcc during operation.

8.3.1.2 3.3V LDO Circuitry

3.3V was required on the power distribution board to interface with the optical flow interface. The NCP1117-3.3 LDO was chosen to step down the 5V input voltage to 3.3V for this use case. 10uF caps were used on the input and output of the LDO to decouple the input and output. A status LED was also included on this output to indicate that the LDO was producing a stable output.

8.3.1.3 ATMEGA328P Microcontroller

The ATMEGA328P Microcontroller included functionality for general purpose digital output, analog input ADCs, and dedicated communication hardware for SPI, IIC, and serial. This allowed the microcontroller to act as the main interface between the on board computer, LIDARs, optical flow board, and voltage and current monitoring circuits. The ATMEGA was also used to generate the 50Hz PWM to drive the 4 side motors within the system.

The general purpose digital IO was used in multiple different ways. The

output was bit banged to produce a 50 Hz PWM with varying pulse widths to control the amount of thrust of each of the side motors. These four 50 Hz PWM signals then had to interface with the side motor ESCs on the high voltage region of the board, so they were passed through and ADuM1410 digital isolator. On the other side of the isolation boundary, the PWM signals were routed to a 12 pin, 4x3 connector that connected directly to the side motor ESCs. The additional 8 pins of this connector supplied power and ground to the four side motor ESCs.

Serial communication to the Benewake long range LIDAR was also bit banged since its single dedicated serial port was already being used by the communication interface to the Raspberry Pi. The digital IO was also used to control the multiplexer which selected one out of eight of the motor current sensors.

The analog input ADCs of the ATMEGA were used to read in the voltage value of the 3 cell motor LiPo and the individual current values of each of the motors. Only one ADC was required for all 8 of the motor currents because an MC14051B 8:1 analog multiplexer was used.

The dedicated SPI interface was used to communicate with the optical flow board, the IIC communication lines were used to communicate with the VL53L0X short range LIDAR, and the dedicated serial lines were used to communicate with the Raspberry Pi via USB. an FT232RL UART IC was used to translate between the ATMEGA's serial TX and RX lines and a micro USB UART connection to the Raspberry Pi.

The ATMEGA328P auxiliary circuitry included an external 16MHz resonator, an ICSP program header to allow for loading of the boot code, and status LEDs on unused pins to be used for debugging purposes.

8.3.1.4 Benewake Long Range LIDAR Interface

The interface to the Benewake long LIDAR included a serial TX and serial RX connection to the ATMEGA, along with 5V and ground connections to power the LIDAR. a 4 pin standard 0.1" right angle female header was used to this connection, and the TX and RX lines connected directly to the hardware dedicated serial ports of the ATMEGA.

8.3.1.5 VL53L0X Short Range LIDAR Interface

The interface to the VL53L0X short range LIDAR included an I2C communication line consisting of SCL and SDA lines, along with 5V and ground connections to power the VL53L0x. The SCL and SDA lines included 1k pull up resistors to 5V for I2C communication. A standard 4 pin 0.1" right angle female header was used to tie the dedicated I2C lines of the ATMEGA to the short range LIDAR mounted on the bottom of the drone.

8.3.1.6 Optical Flow Interface

The interface to the optical flow board included SPI signals which consisted of MOSI, MISO, CLK, and CS signals, along with a SPI reset and power and ground signals. However, the optical flow board operates on 3.3V, so a voltage translation circuit was required to interface between the ATMEGA328P and the optical flow board.

To translate the voltages between 5V and 3.3V, a BSS138 N channel MOSFET was used. The configuration used to act as a level shifter can be see in the figure below. By including pull up resistors on both the drain and source to 5V and 3.3V, along with tying the gate to 3.3V, the FET will not conduct as long as 0V is not applied on either the source or drain. However, if 0V is applied to either side, the FET will conduct and the signal will be pulled low appropriately. This ensures that the ATMEGA and optical flow board can communicate with one another even though they operate at different voltages.

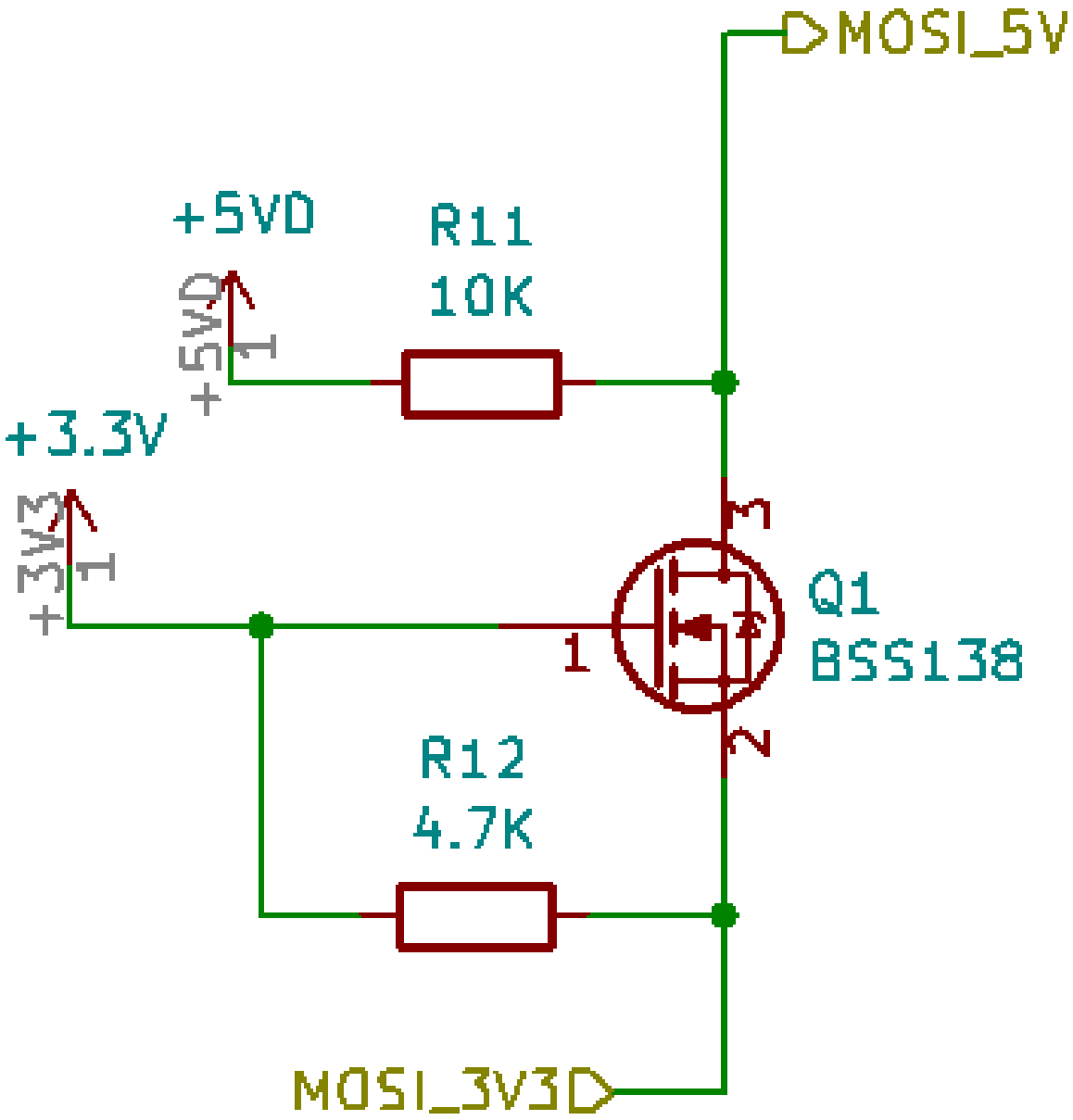

Figure 19: N Channel MOSFET Level Shifter

The connector used to interface to the optical flow board was a 1x7 standard 0.1" right angle header.

8.3.1.7 On-Board Computer Interface

The interface between the ATMEGA and Raspberry Pi included a serial UART connection between the two along with 5V and ground. Specifically, The FT232RL IC was used to convert the serial TX and RX signals of the ATMEGA328P to UART micro USB UART signals for the Raspberry Pi. This allowed the microcontroller and on board computer to communicate to one another serially via USB. Along with the serial connections, 5V and ground connections were also included on the micro USB port. The shell of the USB connector was left floating to prevent any incidents of accidental shorting.

8.3.1.8 Flight Controller Interface

The flight controller interface of the power distribution board included a standard 0.1" 4 pin connector to connect the four 50 Hz PWM signals to control the main motors, which was generated by the flight controller. The interface also included a 2 pin standard 0.1" header to connect Vcc and ground of the flight controller to the power distribution board so that it could be used with a digital isolator.

These four control signals had to interface with the ESCs on the high voltage region of the board, so an ADum1410 digital isolator was used to cross the isolation boundary between the regions. On the other side of the isolator, these four main motor PWM signals were then routed to a 12 pin 3x4, which included power and ground for each of the ESCs as well. This 12 pin connector then interfaced directly to the ESCs controlling the main motors in charge of vertical movement.

8.3.2 High Voltage Isolation Region

The high voltage isolation region included the current and voltage sensing circuits, along with the E-Kill safety circuitry to allow a safe shut down of the propeller motors. This region also included the output interface for the control signals of the motor ESCs. All signals within this region that interfaced with the low voltage region were isolated in some way.

8.3.2.1 Motor LiPo Input Interface

The 3 cell motor battery input interface included a 1725656 phoenix terminal block that allowed input voltage wires to be securely screwed into the board. This ensured that the hot wires would not come undone during aerial operation of the drone. This battery voltage was then routed throughout the board to the 5 Volt generation and voltage sensing circuits.

8.3.2.2 5 Volt Generation Circuit

The 5V generation circuit on the high voltage isolation region consisted of a LM2596S switching converter to step down the input battery voltage to an output voltage of 5V. A large 470 uF electrolytic capacitor was included on the input to ensure a relatively smooth signal on the input of the converter, and an output tantalum capacitor of 220 uF was included to produce a stable output voltage. The feedback loop of the regulator used 0.1% tolerance and the length of the feedback loop on the routing of the board was minimized to minimize any undesired fluctuations at the converter's output. A status LED was included on the output to indicate that voltage was being properly generated at the output of the converter.

8.3.2.3 Voltage Sensing Circuit

The primary component of the voltage sensing circuit was the ACPL-C87B analog isolator. This component was used to provide a direct, isolated connection between the motor battery and an ADC of the ATMEGA328P. The battery voltage was stepped down so that the maximum voltage at the input of the ATMEGA328P was 2V using a voltage divider. A 6.49K and 36.5K resistor were used to achieve a gain of 0.151. The maximum value of the ADCs of the ATMEGA328P was also set to 2.5V by connecting the AREF pin of the microcontroller to 2.5V. This voltage was generated using a voltage divider on the 5V of the low voltage region, and was done to decrease the quantization error of the ADCs.

The output of the ACPL-C87B was a differential signal with a common mode voltage of 1.23V. The positive output of the IC was connected to an ADC of the ATMEGA328P, and the input voltage was calculated by subtracting 1.23V from the output and multiplying it by 2. This allowed for the use of only 1 ADC connection even though the output of the chip was a differential signal, at the cost of some noise immunity. This method was compared to

measuring the voltage across the digital output, and the overall outputs were similar.

8.3.2.4 Hall Effect Current Sensing Circuit

The current sensing was done using ACS781LLRTR-100U-T hall effect current sensing ICs. The power distribution board included 8 hall effect sensors, which allowed individual monitoring of each of the 8 motors on the drone. Since the hall effect sensors measured magnetic fields rather than electrical characteristics of the motor connections, isolation for these parts were not required. Each of the outputs of the 8 hall effect sensors were connected to the inputs of an MC14051B 8:1 analog multiplexer. The output of this multiplexer was then connected to an ADC of the ATMEGA328P so that the currents of each of the motors could be monitored. The selects of this multiplexer were controlled by 3 digital IO signals of the ATMEGA328P.

8.3.2.5 Motor E-Kill Circuit

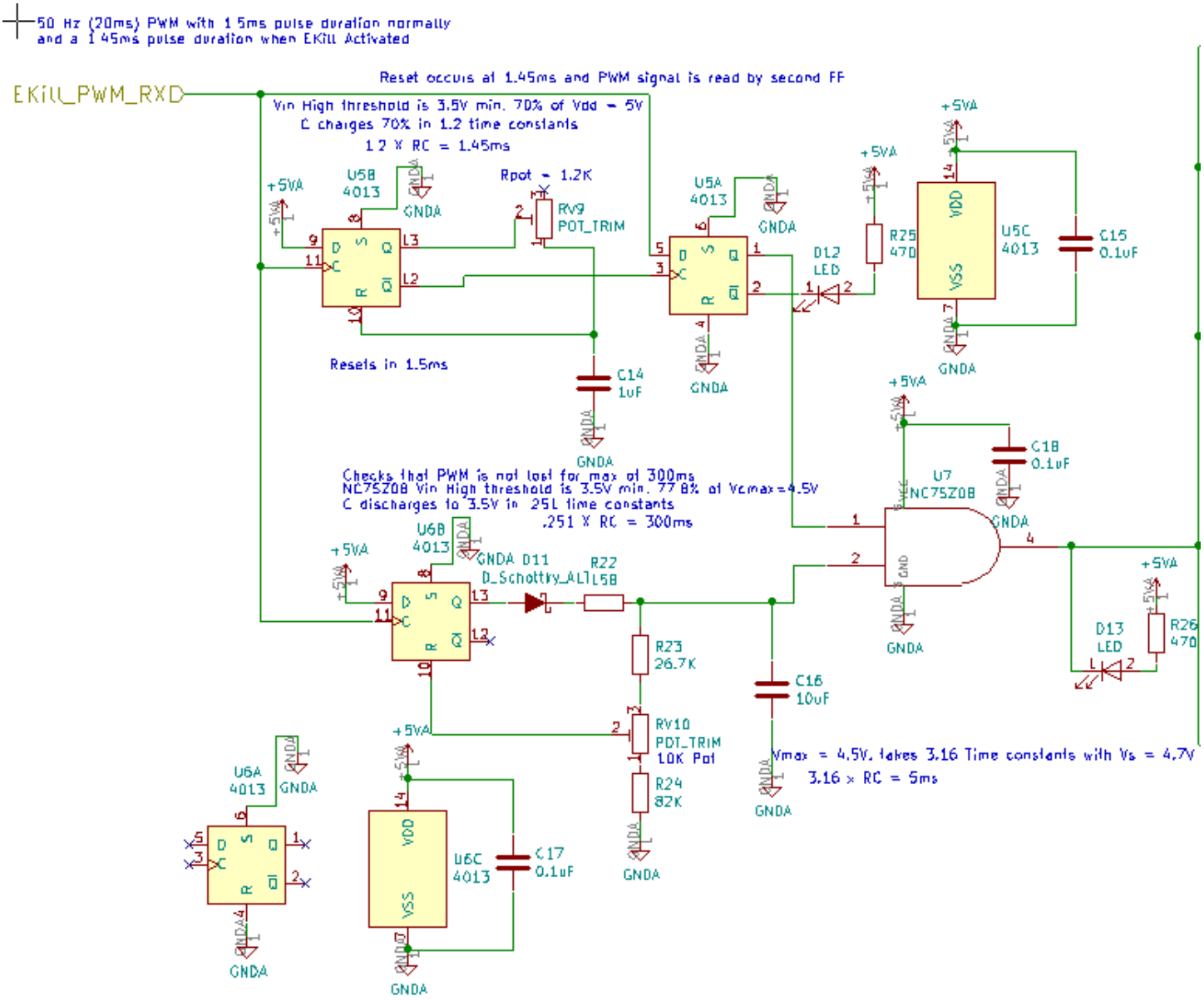

Figure 20: Wireless Safety E-Kill Circuit

The Motor E-Kill circuit receives 50 Hz PWM from an external wireless receiver to set the gate voltage of 8 PSMN1R0-40YLDX NMOS transistors. This transistors are used in a digital configuration as a switch to either connect or disconnect each motor to power. This wireless receiver interfaces with the board via a 1x3 pin standard 0.1" right angle header that includes 5V, ground, and PWM in signals.

During the on configuration, which allows power to be applied to the motors and the drone to operate, the wireless receiver receives a 50 Hz square wave with a pulse width of 1.5ms. When the wireless receiver outputs a signal to turn off the motors, a 50 Hz square wave with a pulse width of 1.45ms is received. The E-Kill circuit was designed to determine whether the pulse width of the received signal is 1.5ms or 1.45ms, and also if the receiver is still receiving a signal.

The E-Kill circuit can be seen in the figure below. The circuitry connected to pin 1 of the AND gate U7 is used to determine whether the pulse width of the received square wave is 1.5ms or 1.45ms, and the circuitry connected to pin 2 of U7 is used as a time out to determine if PWM signal has been received in the past 300 ms.

To determine the pulse width of the received signal, an RC network is attached on the output of D flip flop U5B. The time constant of the network is set with a potentiometeter so that the flip flop will reset sometime between 1.45ms and 1.5ms. This reset will trigger the latching of the second D flip flop U5A, which reads in the status of the received PWM signal. If the pulse width of the signal is 1.5ms, the flip flop will pass a high signal, and if the signal is 1.45ms, the flip flop will pass a low signal.

For the time out circuitry connected to pin 2 of U7, two separate RC time constants are set for charging and discharging of the output capacitor C16. The charging time constant is set by R22 and C16, and is low enough so that C16 can charge up higher than the input high threshold of the and gate within the 1.45ms-1.5ms duration of the received signal. The discharging time constant is set by C16, R23, RV10, and R24 and is large enough so that the input voltage of the and gate will remain high for longer than multiple 20ms periods of the received signal, in the event that the signal is lost. Based on the time constant chosen, a time out is set for 300 ms, which is 15 periods of the received 50 Hz signal. After 15 periods, the output capacitor C16 discharges enough so that it is below the input low threshold of the AND gate. However, if the wireless signal is received again before 15 periods, C16 would be allowed to charge up again and reset the duration of the timeout. D11 is placed so that the capacitor does not discharge back into the flip flop.

Both the pulse width condition and the time out conditions have to be satisfied for the E-kill to be disabled and for it to continue to allow motors to draw power from 3 cell LiPo. If at any time one of the E-kill conditions are not met, the MOSFETs in series with the motors stop conducting and the motors turn off until the E-Kill disabling conditions are met again.

8.3.2.6 Motor Signals Out Interface

The motor output interface, as mentioned above, included two 4x3 connectors. Both of the connectors contained four 50 Hz PWM connections, four 5V connections, and four ground connections. The connectors were separated to distinguish between connections to the main motors used for vertical movement and the side motors used for horizontal translation. These signals were generated by the CleanFlight and ATMEGA328P respectively, and were digitally isolated by two ADum1410 ICs.

8.3.3 Final Prototype Required Hardware Modifications and Future Improvements

Several issues on the final prototype required hardware modifications to ensure proper operation of the power distribution board. This section will outline unforeseen issues along with their resolutions.

8.3.3.1 ATMEGA328P Held in Reset

The ATMEGA328P was constantly held in reset in the original configuration on the board. This was caused because a status LED was placed on the reset pin of the chip. This caused the maximum voltage on the reset pin to be clamped to approximately 1.2V by the LED, which permanently held the chip in reset. Once the LED was removed, the chip was able to come out of reset.

8.3.3.2 ATMEGA328P Fails to Load Bootloader

Two issues made the ATMEGA328P unable to load bootloader code. The first issue was that the reset connection of the programmer was not connected to the reset pin of the microcontroller. The second was that the external 16MHz clock's pinout was defined incorrectly in the layout. Once the reset connection and the resonator were connected properly, the bootloader and any firmware could be loaded on the ATMEGA328P.

8.3.3.3 3.3V LDO Outputs 4.0V

The 3.3V LDO of the low voltage region was found to output 4.0V instead

of 3.3V. This was due to the fact that the 5V output version of the part was placed instead of the 3.3V output. This was resolved by placing a second order and placing the NCP1117-3.3 part on the board. During the time that the incorrect LDO was placed, no components were damaged by the higher voltage.

8.3.3.4 Measured Battery Voltage Innaccuracy

Several factors were caused innacurate readings of the battery voltage monitor. The first was caused by the fact that the negative output of the differential voltage was being shorted to ground in the design. The second and third reasons caused larger quantization error that lead to high levels of innaccuracy. The resistor divider on the input of the sensor originally set the 2V maximum voltage of the input to correspond with a 4 cell LiPo's voltage range of 14.8V-16.8V. The maximum voltage of the ATMEGA328P's ADCs were also set to 5V, when the maximum read voltage on the board was closer to 2.5V

The quantization error was reduced from 110mV to 50mV by setting the maximum 2V input to the voltage range of 11.1V-12.6V of a 3 cell LiPo and by setting the ADCs maximum voltage to 2.5V using the AREF pin.

Once the chip was no longer being shorted and the quantization error was reduced, the quantization error per bin was reduced from around 120mV per bin to 50mV to pin, which increased the accuracy of the readings significantly.

8.3.3.5 Current Sensor Output Invalid

The current sensor output was found to be invalid since the SN74LS151 8:1 multiplexer originally used on the board was a digital IC. This problem was resolved once the analog multiplexer MC14051B was ordered and placed on the board to replace the original digital multiplexer.

The hall effect sensors on the board were also incorrectly ordered as the unidirectionally, 3.3V nominally operating chips. These chips still produced readable outputs, but the current readings may become more accurate if the ACS780LLRTR-100B0T hall effect sensor is used instead, which is the 5V, bidirectional version.

8.3.3.6 ESCs Remained Powered after E-Kill

The E-Kill shut off the ground connections to the motors properly using the FETs, but each ESC was still connected to ground via the control signals connectors. This caused the motors to remain powered even after the E-Kill was enabled.

This problem was solved on the final prototype by disconnecting the grounds on the control signal connectors, which caused the ESCs to float when the E-Kill was enabled. This lead to other issues since the ESCs and power distribution board were on separate references when the E-Kill was enabled. This caused damage to the 50 Hz PWM digital isolators at times, which required the parts to be replaced. To prevent this damage from occurring, diodes were placed with the cathode connected to the ESC and the anode connected to the isolator, which prevented high voltages from getting through to the isolator pins.

In future designs, this issue would best be resolved by disconnecting the battery voltage from the motors instead of the grounds. This would ensure that the ESCs do not remain floating when the E-Kill is enabled. This would also require the 5V connections to the ESCs from the power distribution board to be disconnected in a similar fashion to the motors when the E-kill is enabled, along with the 50 Hz PWM signals so no unexpected back powering of the ESCs occur through the control signals again.

8.3.3.7 Main Motor Input Connector Incorrect Size

A small error in the layout occurred where the footprint of the 1x4 pin connector was incorrectly selected to have 2.00mm spacing instead of 2.54mm spacing. This did not cause too many problems since the 2.54mm spacing connector could still interface with the holes on the board, but would be corrected in future versions of the board.

8.3.3.8 Motor Signals Out Interface Silkscreen Labeled Incorrectly

An error in the silkscreen caused the main and side motor PWM signals to be labeled incorrectly. The silkscreen will be updated in future versions of the board so that connectivity to the board is properly labeled.

8.3.3.9 Non-functional E-Kill Timeout

The timeout functionality of the E-Kill was not functional original because the discharging time constant of the circuit was not large enough. During the 18.5ms low portion of the period of the square wave, the output capacitor was discharging to a voltage low enough to trip the E-Kill. This was resolved by increasing the output capacitance to 20 uF and increasing the total resistance of the discharging node. This increase caused the discharging time for the output voltage to reach the low input threshold in 42 ms, which means that the E-Kill receiver can miss up to two periods worth of signals before the motors are shut off.

This timeout was much shorter than the originally planned timeout, and caused unexpected power disconnects in rare occasions. In the future, the total resistance of the discharging node will be increased to set the discharging time constant to a larger value, which will increase the timeout before the E-Kill trips.

8.3.3.10 Phoenix Connector Footprint Error

Both 1725656 phoenix connectors for the 5V input of the low voltage side and the motor battery voltage on the high voltage side were rotated 180 degrees facing backwards on the board. This was resolved by removing the plastic strain relief tabs on the bottom of the connectors that interface with holes on the board and soldering them on the proper way. In the future, the footprint would be rotated and oriented properly so the connectors will face the correct way and allow use of the strain relief for a better connection to the board.

8.4 Localization

8.4.1 Sensor Fusion

Sensor fusion was handled by the existing RAS IARC stack, specifically with the "Robot Localization" software. This software employs a 15 state extended Kalman filter, and was already able to fuse existing sensor data (optical flow, IMU, altimeters) into an odometry estimate. No large changes to the parameters of the Kalman filter's covariance matrices, or other parameters, were necessary as the same hardware had already been integrated on RAS' 2017 IARC drone.

8.4.2 Ground Truth Localization

8.4.2.1 Background

The algorithm that produces ground truth localization estimates was derived from prior work done by Baek, Park, Cho, and Lee in [21]. The algorithm developed is similar to the one described in [21], but has several optimizations.

In their paper [21], Baek et al. describe a tracking system that utilizes a Kinect to identify a Styrofoam orb, and to use the depth image of the camera to localize the orb's centroid in 3D space.

8.4.2.2 Implementation Details

Because the prior work done [21] did not include software, a new algorithm had to be written from scratch (with some details used from the prior work). The software package was written in Python as a ROS package.

8.4.2.3 Launch File and ROS Specifics

The package includes a launch file that can be ran directly by the roslaunch program, or potentially ran by other ROS programs. The launch file does preliminary work by starting the camera software, setting global arguments for the names of certain topics to subscribe to, starting the program itself (the ROS node), and performing several static transforms so the a "map" marker designating the center of a room is in the right spot relative to the camera.

8.4.2.4 Driver Program

The running node starts up a driver program that subscribes to an RGB image, a 3D point cloud, and a camera information topic. The data from these topics are sent through function callbacks which return modified data which can be more easily used by the program. The driver then create listener and broadcaster objects to recieve and send transformations to the ROS transformation tree. This is done in order to keep track of different parts of the robot in accordance to other parts. For instance: when the camera tracks the UAV the program will send a transform with the (X,Y,Z) location

and a quaternion rotation of the tracked object. It will then relate that location to a parent transform, such asIn the signal processing literature, the use of non-causal (symmetric) filters is commonplace, and the exponential window function is broadly used in this fashion, but a different terminology is used: exponential smoothing is equivalent to a first-order Infinite Impulse Response or IIR filter and moving average is equivalent to a Finite Impulse Response or FIR filter with equal weighting factors.

"map" which is located at (0,0,0), giving a vector between the "map" and the tracked object.

Once the preliminary setup is done, the driver program initializes a tracker object; updating the object with new images and point clouds continuously.

8.4.2.5 Tracking Class

In order to track input images from the driver program, a tracking object performs several prepossessing steps.

First the object extracts the foreground using OpenCV's MOG2 Gaussian mixture model class. Then a Gaussian blur is applied to the resulting binary foreground image, followed by thresholding the image to remove foreground shadows (MOG2 segments foreground features and shadows in different intensities). Then morphological opening and closing is done on the binary foreground image to clean it up and remove thin connecting pieces.

Once a clean binary foreground image is had, the image is processed by a "circle detector" to find the marker orb. A simple edge detection algorithm is ran on the image to find contours, and each simply connected contour is processed. The area and perimeter of each contour is calculated using OpenCV, and circularity is determined using the expression:

$$circlularity = 4\pi \frac{area}{perimeter^2}$$

Where a perfect circle would a circulatiry of 1.0. Contours that have a circularity within some threshold (we chose 0.7, and 1.2) are selected as potential markers. From these labelled circles the largest area contour is chosen as the marker. The center of this circle is used by the tracker object to find the 3D centroid of the marker. The center is passed to a helper function that validates that pixel to a registered point cloud, making sure the depth at the point is not NAN or infinity. Then the (x,y,z) coordinates from that

point cloud pixel is taken and transformed to the map reference frame, and published to the larger transform tree for other programs to use.

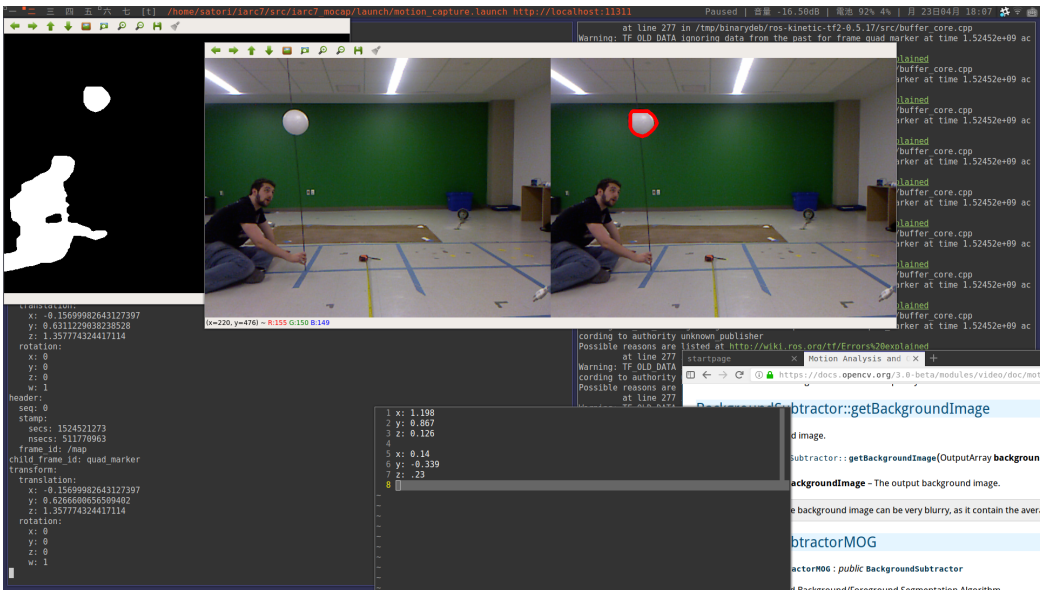

Figure 21: Example of the tracker locating the marker.

Shown in Figure 21 is an example of the marker tracking in operation. The binary image is shown on the top left with the actvie foreground. The original image and an image of the tracked marker are shown side by side.

8.5 Controller

The controller was implemented within RAS's IARC software stack. The complete software engineering behind this piece of this software is beyond the scope of this document as the architecture was not developed as a part of this project. Instead, the development of new features using the already existing architecture was the focus. To give context to the following design components a summarized software design diagram containing only the parts relevant to the controller is shown in Figure 22.

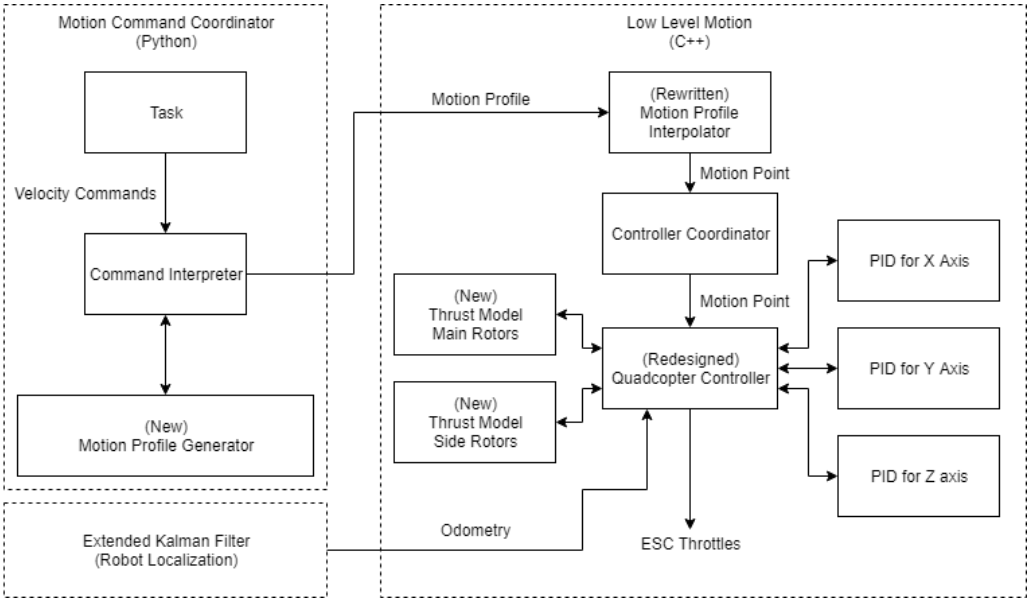

Figure 22: Block diagram of RAS's IARC software stack showing the parts relevant to controller design

A high level overview of the information flow will follow.

The Motion Command Coordinator is a Python ROS node that handles task switching and state transitioning. This node was already written prior to the start of this project. This project added a linear motion profile generator that takes advantage of the fact that tasks in the motion command coordinator can only request velocities. The Motion Profile Generator generates a motion profile 200 milliseconds in to the future and upon every update of the Motion Command Coordinator (every 100ms). The motion profile is sent via ROS to Low Level Motion.

A motion profile is a list of motion points. A motion point is a timestamped position, velocity, and acceleration that is to be achieved by the controller.

Low Level Motion is a C++ ROS node designed to run at a high update rate. In this case 50 Hz. It was also previously written by the RAS IARC team. However, it was heavily modified and new parts were added to support this project. Specifically, this project rewrote the Acceleration Planner into a new class called the Motion Profile Interpolator, redesigned the Quadcopter Controller to support 6 DOF mixing output mixing, and re-designed the thrust model used by the Quadcopter Controller.

When a motion profile is sent from high level motion it is received by the Motion Profile Interpolator in Low Level Motion. The Motion Profile Interpolator has an internal buffer of motion points from previously received motion profiles. In order to accept the new profile the Motion Profile Interpolator replaces all points with timestamps newer than the motion points received with the motion points received.

The next stop is the Controller Coordinator. The Controller Coordinator runs at a fixed update rate (50 Hz) and queries the Motion Profile Interpolator for motion points at specific times. The Motion Profile Interpolator searches its internal buffer and interpolates between the two motion points closest to the desired time in order to generate a new motion point.

The new motion point is then treated as the target motion point by the Quadcopter Controller. It is compared to the Odometry from the extended kalman filter and the error is used for 3 PID controllers. In the case of the PID for Z axis, height is used. In the case of PID for X and Y velocity is used to generate the error. In all cases the outputs of the PID controllers are interpreted as accelerations. The accelerations are used as inputs to the Thrust Model in order to find a target voltage that should be sent to the each of the side rotors and the main thrust motor. The voltage is divided by the current battery voltage to find the throttle percentage and those throttle percentages are sent back out through ROS to the appropriate hardware interfacing nodes.

The rest of this section will focus on the Thrust Model, Motion Profile Generator, and the details of the Controller Implementation.

8.5.1 Thrust Model Design

The thrust modeling is completely covered in the report "6 Degree of Freedom UAV Thrust Model Summary" which can be found as an accompanying file with this document. In the report the complete mathematical derivation behind the equations used for the thrust model is covered as well as how testing was performed and how the model was eventually derived from the test results. The document is over 20 pages so it was not appropriate to include it in this report.

The thrust model can be summarized as follows. A model was needed that could predict the thrust at the next time-step of a controller update cycle based on the currently estimated thrust and the currently applied voltage. To accomplish this the rate of thrust change needed to be characterized

at all operating points. To accomplish this, step impulses of 10% throttle increments were sent to a motor, ESC, and prop combination while it was mounted to a thrust test stand. The thrust test stand recorded the applied voltage, thrust, and current draw at between 50-100 Hz depending on the sensor. In total, over 100 such steps were applied. The output data was then processed with SciPy to find the time constant associated with first order rise. These time constants differed for accelerating and decelerating as well as all different operating points. Least squares was used to fit a polynomial surface to the timeconstants based on the starting thrusts and final voltages. Two surfaces were used, one for acceleration and one for deceleration. The steps were also analyzed to find the steady state voltages and thrust associated with them.

The steady state data was used to generate a steady state voltage to thrust model while the time constant data was used to predict the rise rate from one setpoint to another. Together this data was used to predict the ending thrust for a range of starting thrusts and range of final voltages with a known timestep. The timestep used is determined by the expected update rate of the controller. In this case, the controllers updated at 50 Hz so the timestep used was 20 ms.

Figure 23 is an example of the non-linear thrust model generated. The model was then outputted to a yaml file as a 2 dimensional array. The model was then searched for the lowest voltage that would give the yield the desired thrust based on the starting thrust. Bi-linear interpolation was used to interpolate between the data points.

This implementation of the thrust model in the controller was used so that changes to the thrust model would not require code changes in the controller. Instead a new thrust model could simply be loaded in.

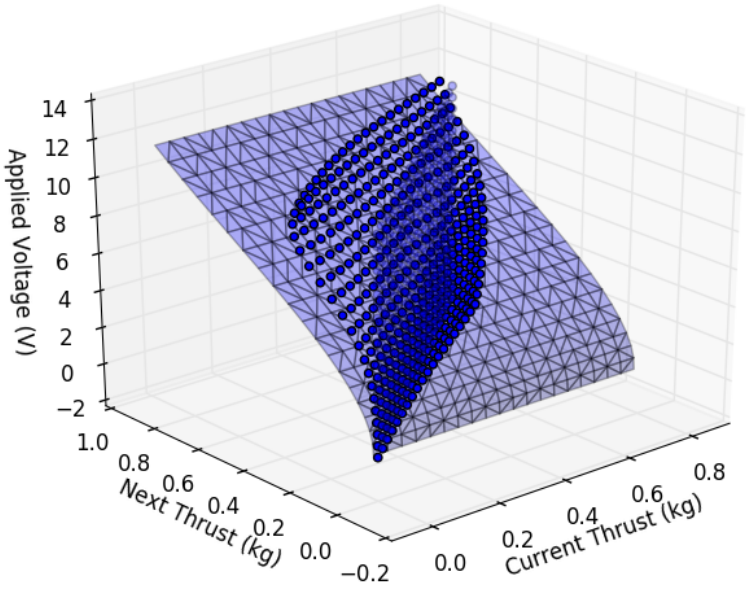

Figure 23: Example thrust model output showing starting thrusts, ending voltages, and expected output thrusts. The blue surface shows the steady state models predictions.

The designed modeling procedure can be used on any hobby motor, ESC, and prop combination. Full testing was done for the main and side rotors. It is shown in the "6 Degree of Freedom UAV Thrust Model Summary" that mechanical lag was significantly reduced and the max thrust slew rate increased by four times.

8.5.2 Motion Profile Generation

The Motion Profile Generator exists to generate future motion profiles based on a desired 3 dimensional acceleration limit. The motion profiles are generated 200 ms into the future and regenerated every 100 ms based on the currently desired velocity. This protects the low level motion from setpoints changing too quickly and makes tasks in the motion command coordinator simple to write. Additionally, Low Level Motion takes advantage of the fu-

ture stamped profiles to run the controllers ahead of real time so that the thrust model can be used effectively.

When the Motion Profile Generator receives a target velocity it searches through the last motion profile generated and finds the two motion points closest to the timestamp of the target velocity. The motion points are interpolated between, and the resulting motion point is used as the starting point for the soon to be generated motion profile. Next, the difference between the target velocity and the currently estimated velocity is used to decide whether or not an acceleration sequence is needed based on the maximum allowed acceleration. If it is required, a series of velocities are generated that ramp to the desired velocity without violating the maximum acceleration limit. If the 200 ms profile is filled, no further work is done. If more time is required to complete the profile, velocities equal to the target velocity are appended to the profile.

The targeted velocities are integrated and differentiated to create the target positions and target accelerations. The resulting discrete acceleration, velocity, and position profiles are then packed into a motion profile and sent to Low Level Motion.

One of the original requirements was the existence of rated limits and the enforcing of limits on jerk, acceleration, and velocity. During the writing of the Motion Profile Interpreter it was mathematically proven that enforcing second order limits on a quantity was a non-trivial motion planning task that was outside the scope of the project. Thus the Motion Profile Generator was designed to work on first order limits. It would accept velocities, enforce acceleration limits, and integrate the velocities into positions. This allowed the task design to remain simple, and allowed sane limits on the input to the controller to be enforced.

8.5.3 Controller Implementation

The controller, at its heart, is three PID controllers. One for each axis in the global frame. The PID controllers are set to control the thrust in each of the orthogonal euclidean vectors of the non-rotated drone frame so that they never interfere with each other. The controller for height was not designed specifically as a part of this project though it was significantly overhauled by RAS members and members of this design team during the duration of the project. It will be described so that implementation of the side rotor controller can be justified.

The height PID controller runs off the error in the target height and the measured height from the kalman filter's odometry. The output is interpreted as an acceleration. The D term of the PID controller uses the kalman filter's estimate of velocity instead of differentiating position. This allows the D term to operate with a low pass filter since the kalman filters estimate of velocity is smooth enough. The output of the height controller is interpreted as a desired corrective acceleration in the map's z axis. It is added to the expected gravitational acceleration of 1 g and the desired current target acceleration. The resulting sum is multiplied by a trigonometric function to correct for the request orientation of the drone. In the case of the 6 DOF UAV the orientation is always assumed to be level with the ground so this corrective factor defaults to 1.

The final acceleration sum for the height controller is fed into the thrust model interpreter for the main rotors. The thrust model uses the currently estimated thrust and the new desired thrust to calculate a voltage that if applied instantaneously will result in the desired acceleration at the current time. Because the controller is running in the future this allows the thrust for the motion point to applied at the time it was actually desired.

The side rotor PID controllers operate in a similar manner. The target velocities in the map frame are fed into the PID controls for the X and Y axis in the map frame. The error between the measured velocity and the target velocity are used as the inputs to the PID controller. Because the PID controller is operating as a first order controller the D term is not used.

The final summed acceleration vectors are rotated by the drones current yaw to find the target accelerations for the actual orientations of the side rotors. The sign of the resulting accelerations are used to determine the side rotor pair to activate. The desired acceleration is fed into the appropriate side rotor thrust model object to determine a voltage to apply to the side rotor.

In all cases the resulting voltages are divided by the measured battery voltage to calculate the throttle to apply to the appropriate rotor. The throttles are then sent out through ROS to the hardware interfacing nodes.

9 Testing, Data Analysis and Results

9.1 Aerial Platform

9.1.1 Performance Criteria

9.1.1.1 Side Rotors Provide at least 0.6 g of acceleration

9.1.1.2 UAV is capable of being flown with a human pilot

9.1.2 Test Cases

9.1.2.1 TST-3.1.4

The altimeter will be verified with by propping the UAV on its side and placing a flat surface in front of the altimeter. The distance measured will be compared to the actual distance. The altimeter's readings will be verified to be within the tolerance of the altimeter sensor.

9.1.2.2 TST-3.1.4-Result

We ran the test as specified with the flat surface a measured 0.35 m from the sensors. We measured 0.35 m from the from the short distance altimeter, and 0.36 meters from the long distance alitmeter. This is within the tolerances for the short distance altimeter, but 2% less accurate than what the long distance altimeter claims.

9.1.2.3 TST-3.1.5

The drone will be placed on a moving cart and dragged at a fixed rate with an automatic winch system. The measured velocity will be compared to the commanded. The readings will be verified to be within the tolerance of the optical flow sensor.

9.1.2.4 TST-3.1.5-Result

This test was never performed due to not having time to find a suitable

cart and make an automatic winch system. For our best effort at a suitable replacement, go to TST-3.3.2.

9.1.2.5 TST-3.1.6

The flight controllers orientation will be tested through a manual test flight with a human pilot. This will require an experienced pilot in order to evaluate performance.

9.1.2.6 TST-3.1.6-Result

When the drone was manually flown with the SPF3EVO in angle mode, it responded appropriately to all setpoint thrusts and orientation commands. The behavior during the test left us reassured that the flight controller would be able to keep the drone level despite disturbances from the side rotors.

9.1.2.7 TST-4.1.3

The thrust of the side rotors will be verified using a thrust test stand. It will be compared to the weight of the model to ensure that 0.6g of acceleration can be provided.

9.1.2.8 TST-4.1.3-Result

We were only able to achieve 0.3g of acceleration on the side rotors. However, no other requirements depended on achieving that specific level of acceleration, and overall our drone was still able to translate horizontally much faster than it previously could. We can currently achieve a maximum acceleration of $0.3g * 1.414 = 0.424g$, assuming that we have two side rotors running at once.

9.2 Power Distribution Board

9.2.1 Performance Criteria

9.2.1.1 The DC voltage of the 2 cell and 3 cell LiPos, along with the 5V output of the DC-DC buck converter and 3.3V LDO are properly generated and routed throughout the board

9.2.1.2 Both voltage levels of the battery and current consumption of each motor are properly communicated to the on board computer

9.2.1.3 Electrical isolation barrier is implemented successfully so noise generated by motors do not affect control system

9.2.1.4 E-Kill functionality allows motor power to be remotely controlled with no significant amount of delay

9.2.2 Test Cases

9.2.2.1 TST-3.2.1

The supply voltages created for the board will be verified by an oscilloscope to measure output voltage. The output voltage will be verified to be within 10% of the intended value.

9.2.2.2 TST-3.2.1-Result

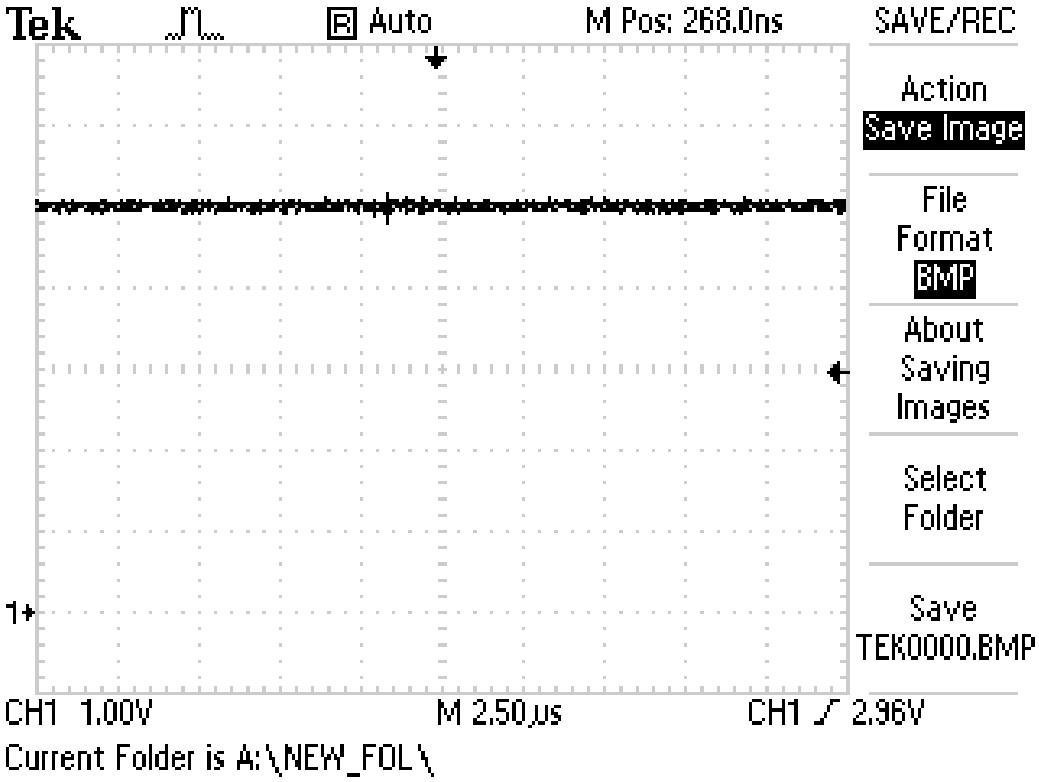

Figure 24: Output Voltage of 5V Switching Converter

The steady state output voltage of the 5V switching converter is seen above. The converter has a nominal voltage of 5.01V, which is within the acceptable limit for all devices within the system.

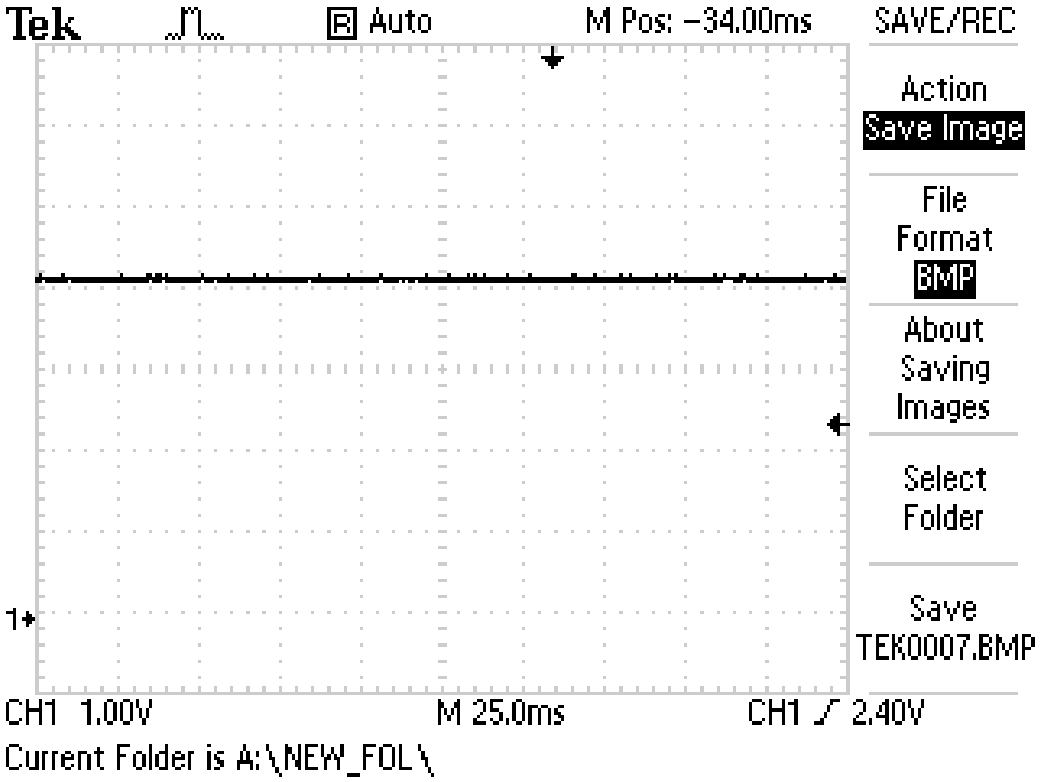

Figure 25: Output Voltage of 3.3V LDO

The steady state output voltage of the 3.3V LDO is shown above to be around 4.0V. This large difference in voltage was discovered to be caused due to the fact that the incorrect version of the NCP1117 LDO was purchased. The 5V output LDO was used instead of the 3.3V output. In the meantime, the 3.3V output voltage of the on board computer will be used to provide 3.3V to the board. Eventually, the LDO will be replaced with the NCP1117-3.3V component.

9.2.2.3 TST-3.2.3

The current monitor circuits will be verified by connecting a channel of the power distribution board to the thrust test stand. Different levels of thrusts with known currents will be set, and these currents will then be recorded by the hall effect sensors and outputted to the ATMEGA328P microcontroller. The microcontrollers ADC value will be compared to measured current values using a current clamp to ensure that the monitor is working properly.

9.2.2.4 TST-3.2.3-Result

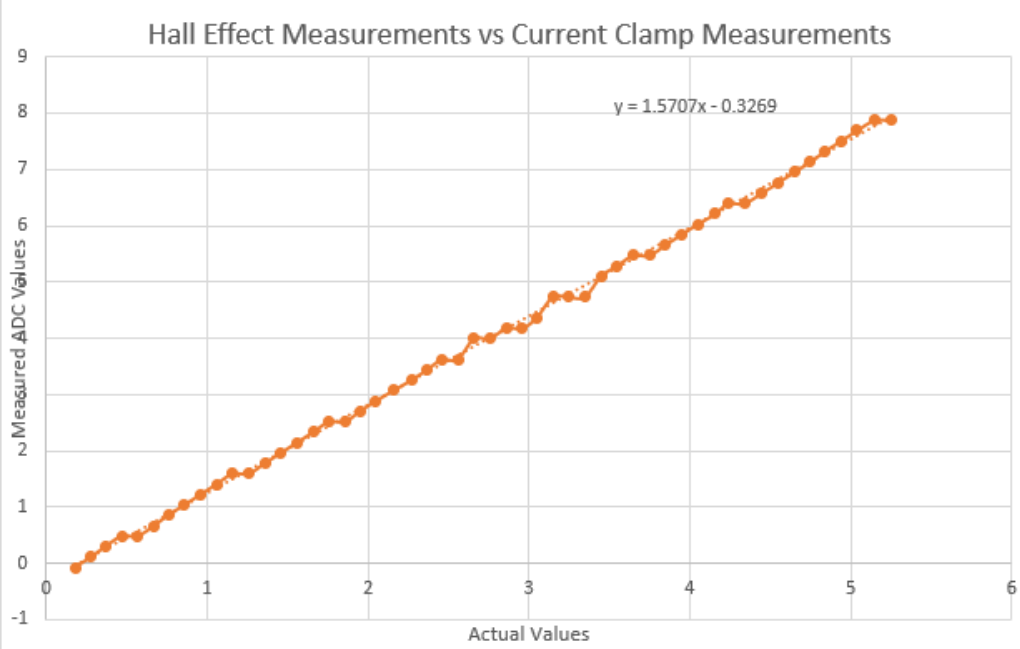

Figure 26: Bench Top PS Currents - Hall Effect vs Current Clamp Measurements

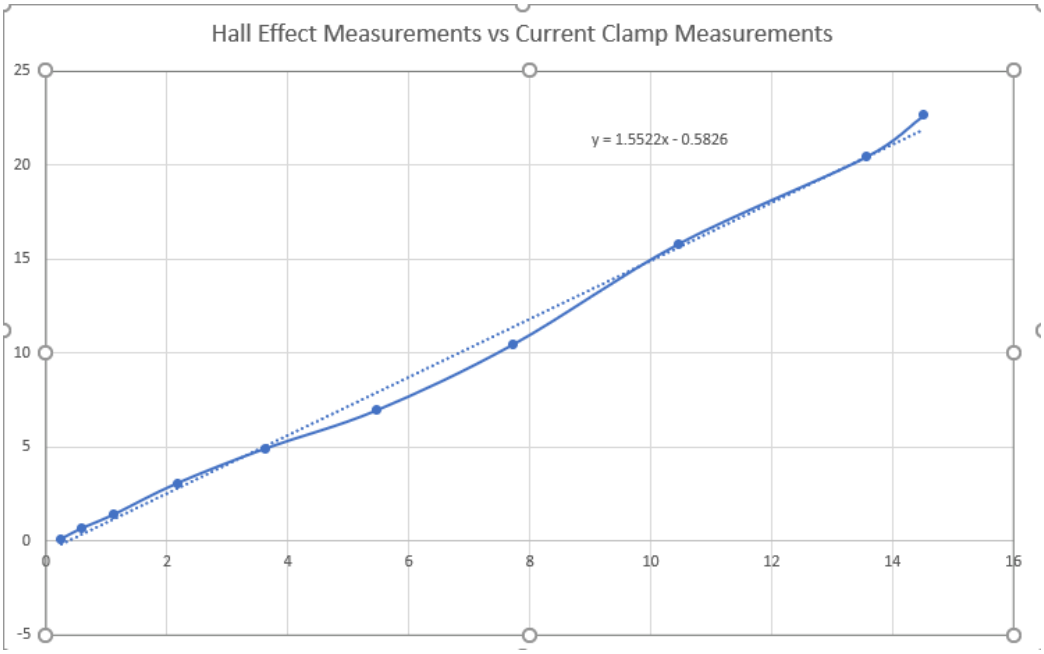

Figure 27: Thrust Model Currents - Hall Effect vs Current Clamp Measurements

Two plots of recorded currents by ADC measurements and actual current values measured by a current clamp are shown. The top plot has values recorded on a test stand consisting of a bench top power supply connected to the power distribution board. The bottom plot shows values recorded on the thrust test stand model with currents drawn by an actual motor.

In both cases, the results were linear. A linear approximation of both graphs was calculated. Both graphs had slopes of around 1.5 and offsets of -.3269 and -.5826. These offsets and slopes can be removed via a linearization operator in software so that we get measured values and current values that are one to one.

9.2.2.5 TST-3.2.4

The voltage monitoring circuit will be verified by inputting known voltages of 11.1v and 22.2V, which are the typical voltages of 3 cell and 4 cell LiPos, using a benchtop supply. The ADC output will then be verified to correspond with the properly scaled voltage that was inputted into the ADC. Voltages ranging between 11.1V and 22.2V will be inputted into the input terminals

of the batteries in 1V increments, and the output of the analog isolator as well as the ADC values of the microcontroller will be recorded. The analog voltage will be measured by a multi-meter. These values will be recorded in a table.

9.2.2.6 TST-3.2.4-Result

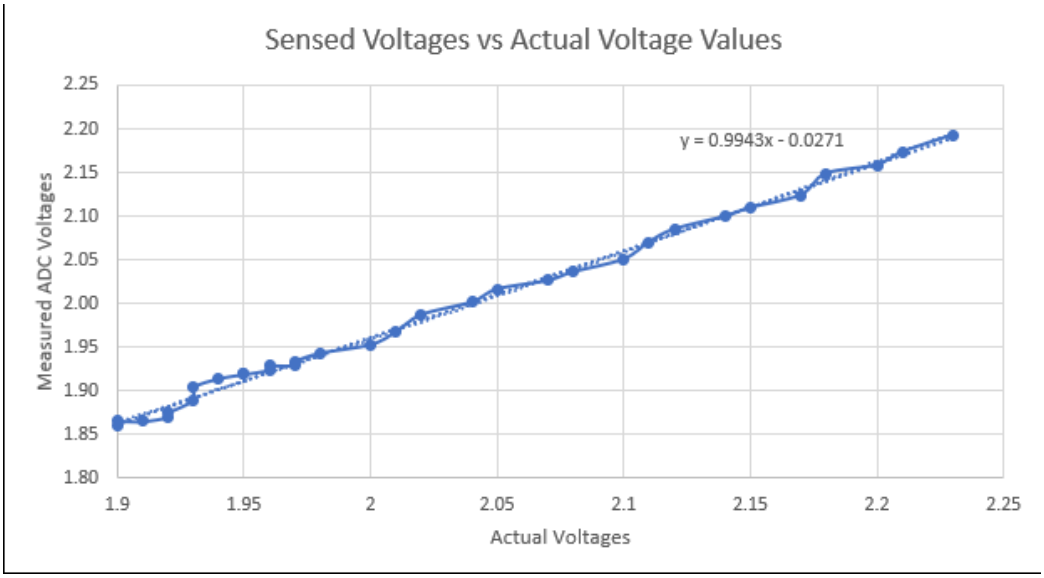

Figure 28: Measured Voltages vs Multimeter Measurements

The plot above shows recorded voltage values using the on board analog voltage sensor and ADC and an external multimeter. The input voltage was stepped through the entire voltage range of a 3 cell and 4 cell lipo.

The linear fit of the data showed a slope that was pretty much unity (0.9943, and an offset of -0.0271. This offset can be removed in software to get a more accurate measurement of the actual battery voltage.

9.2.2.7 TST-3.2.5

The switching characteristics of the power MOSFETs will be verified using an oscilloscope. The FETs will be checked to ensure that they remain

on during normal operation, and switch once the wireless signal is received with less than a 1 ms delay time. LEDs will also be used for visual feedback so that proper tuning of the E-Kill circuit can be verified.

First, the waveforms received by the remote control will be measured using an oscilloscope during both the state when the E-Kill is disabled, and when it is enabled. The proper RC time constant will then be calculated based on the period and duty cycle of the received waveforms. The potentiometers RV9 and RV10 will then be tuned accordingly to disable and enable the motor FETs at the appropriate times. This will be verified by using the installed indicator LEDs D12 and D13 on the board, to ensure that the E-Kill circuitry is tuned properly and the motor FETs are being controlled properly.

9.2.2.8 TST-3.2.5-Result

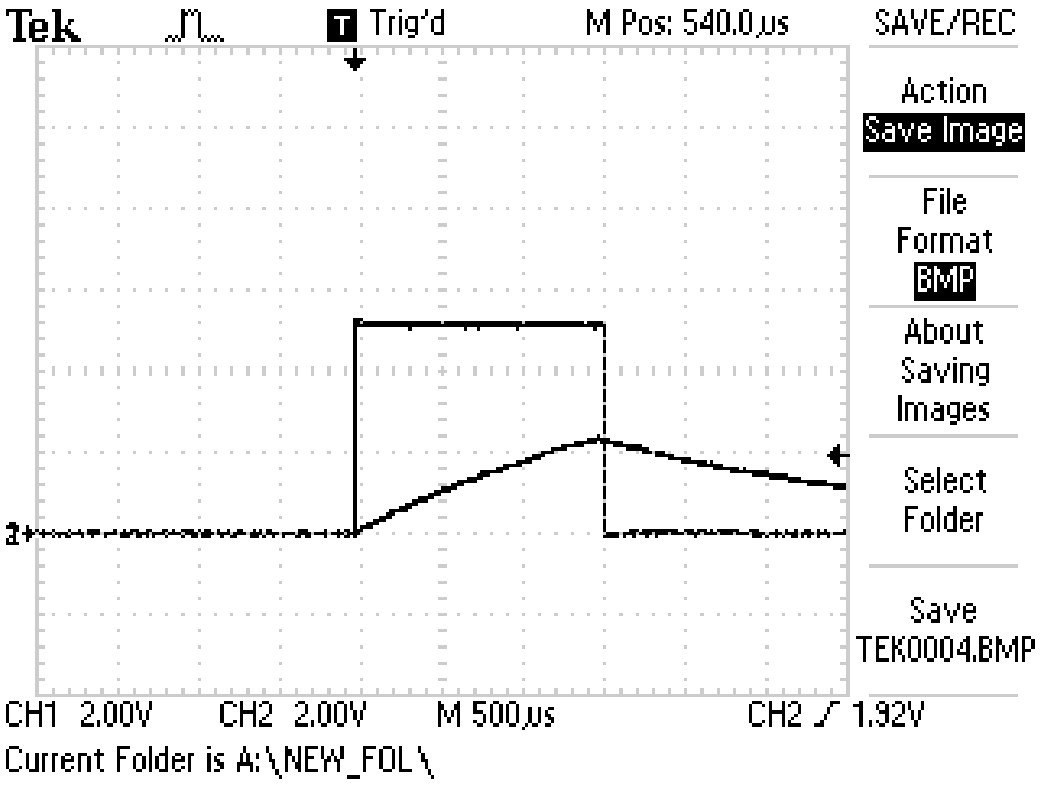

Figure 29: EKill Waveform with FETs Enabled

This graph shows the input PWM signal as a square wave and the output signal of the flip flop U5B during the case when the E-Kill is disabled, which refers to when the systems motors are powered.. At the time that the slope of this signal changes from rising to falling is when the flip flop resets, and the resetting of the flip flop is when the second flip flop. U5A, latches the signal at its input to its output. As shown here, when the e-kill is disabled, the second flip flop latches at the end of the input of the PWM before it goes low, setting its output of flip flop U5A to high. This keeps the motors connected to the battery.

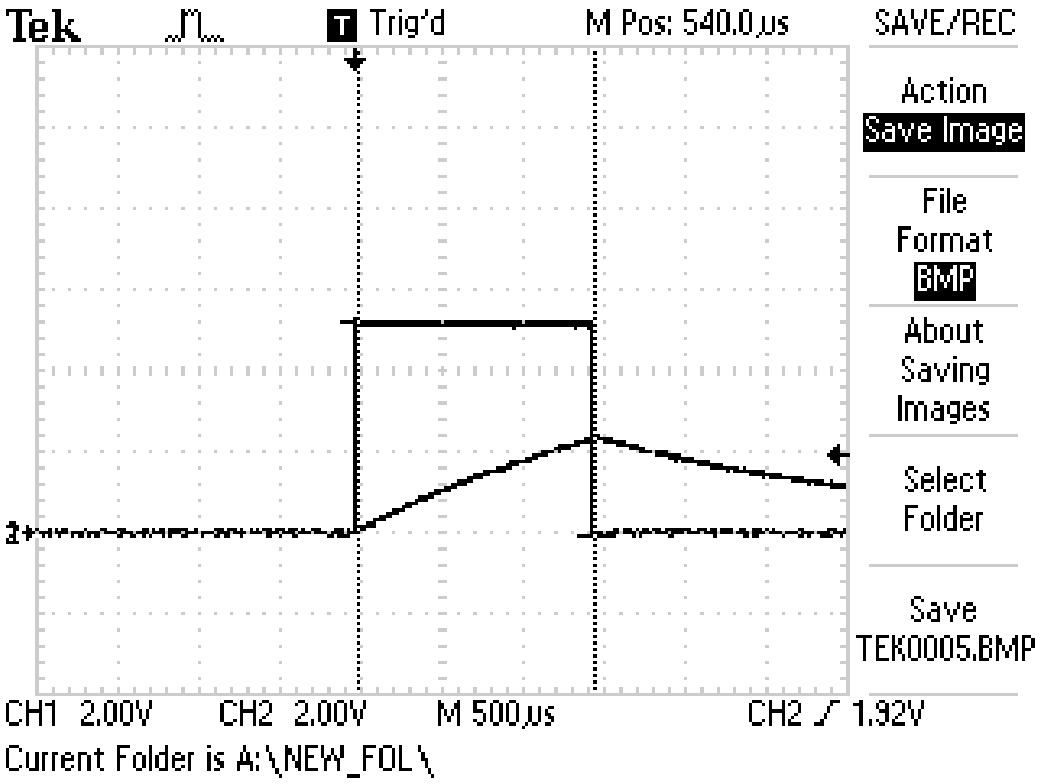

Figure 30: EKill Waveform with FETs Disabled

This graph shows the case where the e-kill is enabled and the motors are disconnected from the main thrust battery. As shown above, the flip flop resets during the time after the high signal of the input PWM. This latches 0V into the output flip flop U5A, which disables the gates of the motor control FETs. This disables the motor function and kills the circuitry remotely, which will be used for safety during testing.

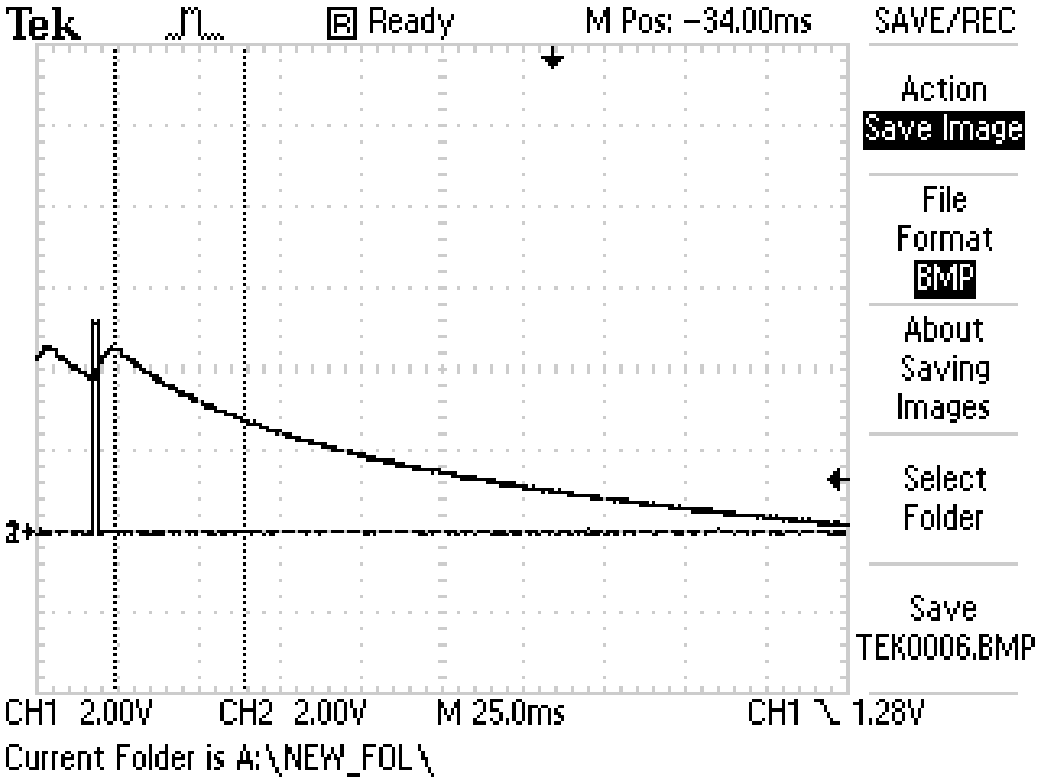

Figure 31: EKill Waveform with FETs Disabled - Time Out Condition

The graph above shows a single input PWM representing a loss of signal, and the discharging of the output of U6B. This portion of the E-Kill circuit checks to see if the external PWM signal is still being received. The minimum voltage at which the and gate input registers as a digital low signal, 2.6V, is marked on the graph using the cursor. This reset occurs at 42 ms, which means that the circuit will output a low signal after 2 periods of the input PWM. This means that the circuitry will detect if it fails to receive more than two periods of the PWM signal from the external receiver. This portion of the circuit acts as a safety mechanism to shut off the drone in the case of a loss of signal from the wireless receiver.In the signal processing literature, the use of non-causal (symmetric) filters is commonplace, and the exponential window function is broadly used in this fashion, but a different terminology is used: exponential smoothing is equivalent to a first-order Infinite Impulse Response or IIR filter and moving average is equivalent to a Finite Impulse Response or FIR filter with equal weighting factors.

9.3 Localization

9.3.1 Performance Criteria

9.3.1.1 The desired update rate is reached

9.3.1.2 The filtering minimizes the amount of error going into the controller and makes it possible for the controller to perform its function well

9.3.1.3 The motion tracking system is able to determine the pose and position of the drone .

9.3.2 Test Cases

9.3.2.1 TST - 3.3.1

The time-stamps of incoming estimates while the drone is manually moved around will be examined to verify that the localization is updating at the necessary speed. However, the ultimate test will be whether the controller functions.

9.3.2.2 TST - 3.3.1 - Result

Instead of the test plan documented, it was found that there was a ROS command (rostopic Hz) that automatically documented the update rate of a sensor. We have posted a printout of the update rates that we were able to achieve for each individual sensor.

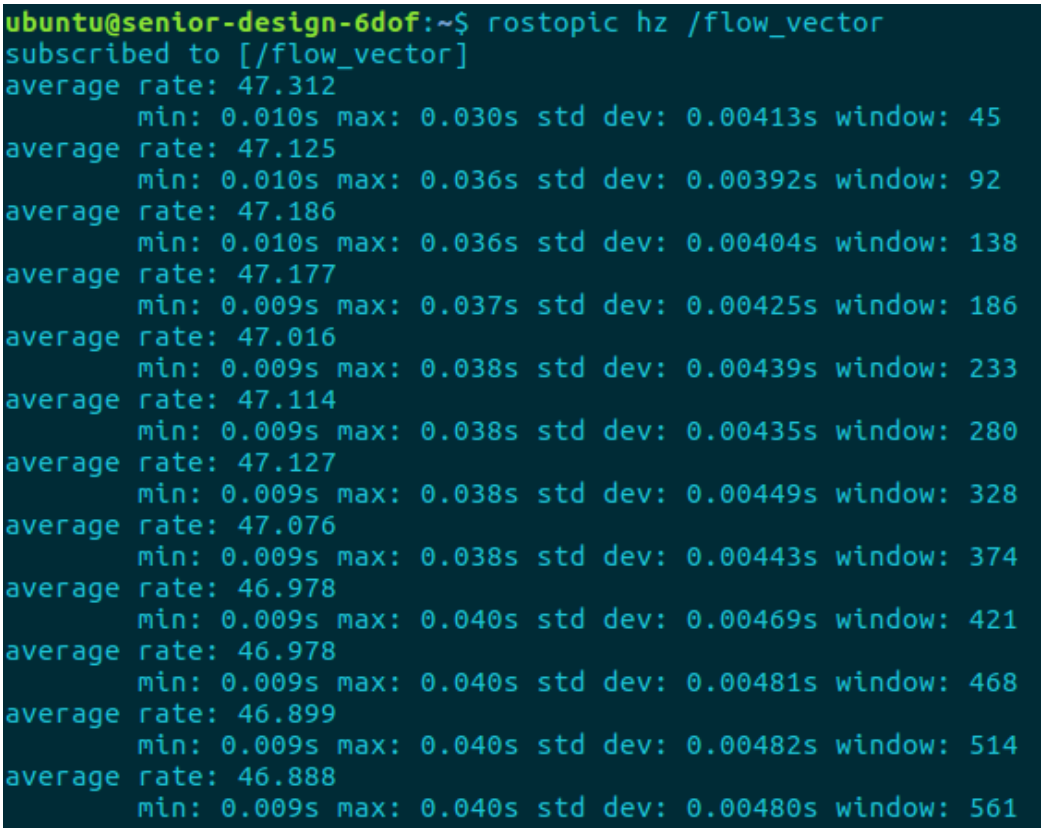

Figure 32: Update rate of Optical Flow Sensor (47 hz)

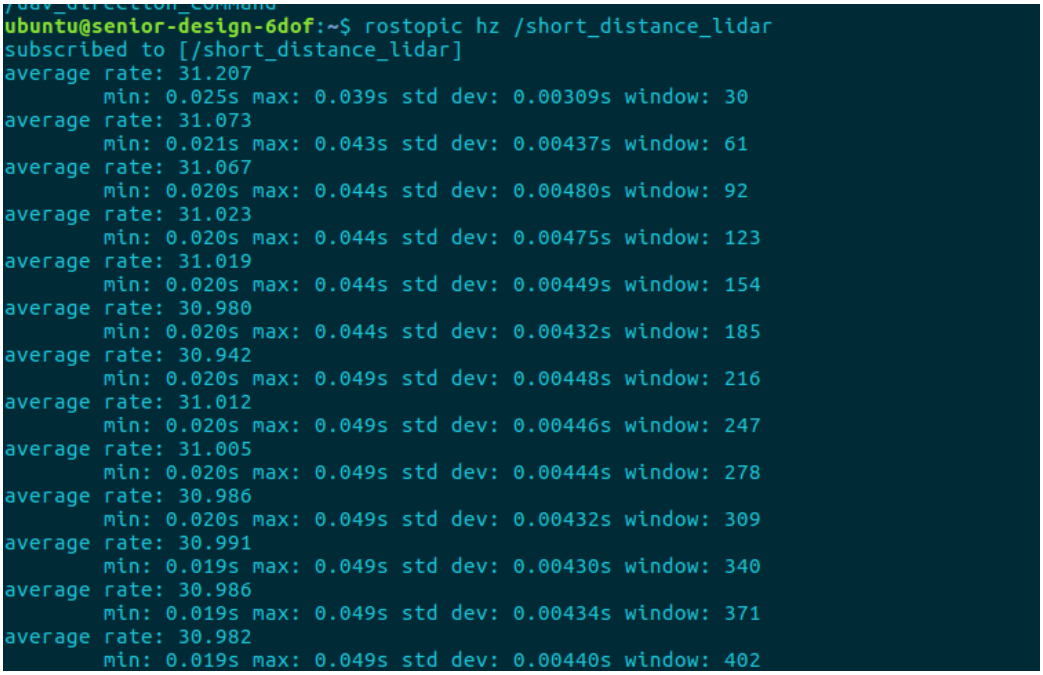

Figure 33: Update rate of Short-range Lidar (31 hz)

```
ubuntu@senior-design-6dof:~$ rostopic hz /long_distance_lidar
subscribed to [/long_distance_lidar]
average rate: 71.926
        min: 0.008s max: 0.037s std dev: 0.00653s window: 70
average rate: 70.018
        min: 0.008s max: 0.038s std dev: 0.00718s window: 139
average rate: 68.858
        min: 0.008s max: 0.043s std dev: 0.00724s window: 206
average rate: 69.394
        min: 0.008s max: 0.043s std dev: 0.00688s window: 277
average rate: 69.687
        min: 0.008s max: 0.043s std dev: 0.00683s window: 348
average rate: 69.164
        min: 0.008s max: 0.043s std dev: 0.00679s window: 414
average rate: 68.935
        min: 0.008s max: 0.043s std dev: 0.00685s window: 482
average rate: 68.982
        min: 0.008s max: 0.047s std dev: 0.00706s window: 552
average rate: 70.034
        min: 0.008s max: 0.047s std dev: 0.00693s window: 630
average rate: 70.163
        min: 0.008s max: 0.047s std dev: 0.00692s window: 702
average rate: 70.391
        min: 0.008s max: 0.047s std dev: 0.00692s window: 774
average rate: 70.150
        min: 0.008s max: 0.047s std dev: 0.00693s window: 842
average rate: 70.737
        min: 0.008s max: 0.047s std dev: 0.00681s window: 920
average rate: 69.931
        min: 0.008s max: 0.047s std dev: 0.00690s window: 980
```

Figure 34: Update rate of Long-range Lidar (70 hz)

Overall,the drone did not manage to hit the roughly 90 hz that was claimed that would be accomplished for all of the sensors. This was actually impossible for the short distance lidar; its maximum update rate was only 30 hz. For the flow vector, it could easily have achieved a higher update rate than the one displayed, but it would have resulted in very small errors causing great fluctuations in velocity estimates. We decided that artificially limiting the update rate of the optical flow sensor (and allowing it to accumulate a certain number of pixels before we checked the change in number of pixels) was a good solution. We experimented with different update rates until we found the one displayed in the test case (roughly 50 Hz), believing it to be a good tradeoff. There is no such excuse for the long distance lidar. Its maximum advertised update rate was 100 Hz, and we could not manage to achieve that rate. However, we did not put too much effort into it because

height hold worked fairly well halfway through the semester and writing code for the optical flow sensor was a more urgent calling at the time.

9.3.2.3 TST - 3.3.2 - Filtering

Localization testing will be performed by manually moving the drone to predetermined set-points and recording the drone's estimate of where it was. Comparisons should show a error of less than 5 centimeters. Future testing will involve trying to hit the same setpoints with autonomous flight.

9.3.2.4 TST - 3.3.2 - Result

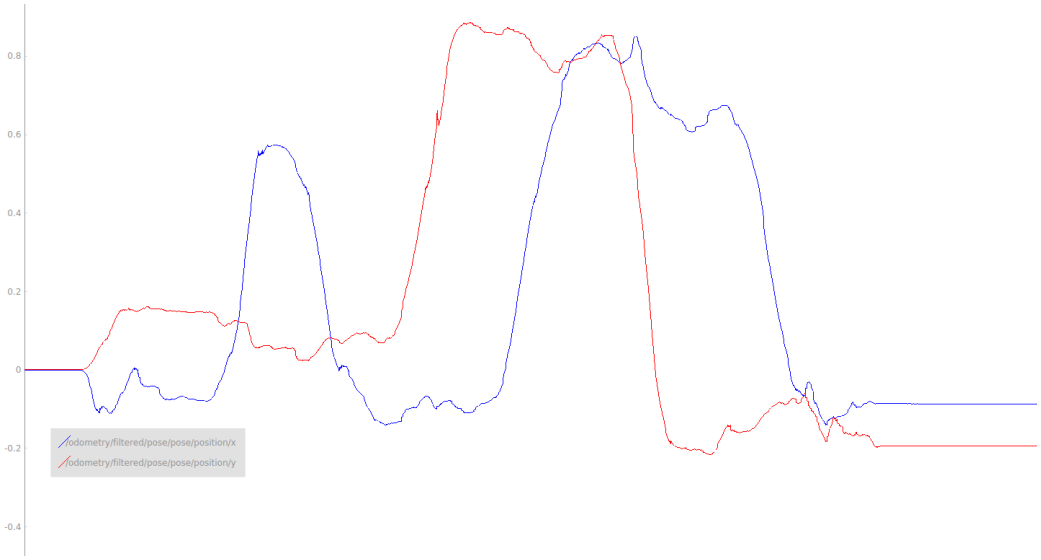

Figure 35: X and Y position outputs from Kalman filter after setpoint test

This is a record of the drones measurements; moving the drone to recorded setpoints and then moving it back to its original position. Even without knowing the position of the setpoints, one can see that the there was a maximum error of about 0.2 meters between the drones initial position and its return to its initial position.

We first attempted to move the drone 0.5 meters in the y direction. The drone itself had an error of about 0.05 meters. we then moved it backwards

0.5 meters, which resulted in an error of about 0.1 meters. we then tested the x axis position estimates by moving it in the positive x direction 1 meter, which resulted in about 0.07 meters of error. Finally, we moved it back to its original position, which resulted in a final error of error of -0.1 meters in the y direction and -0.2 meters in the x direction.

There are a number of sources of error that most likely occurred as we were performing this test, including measured changes in X and Y position that occurred as we were lifting the drone off the ground, similar inaccurate changes that occurred as we put the drone back down on the ground, and slight changes in orientation from a lack of rigidity in my arms causing optical flow readings that were inaccurate with respect to the amount of translation we were imparting upon the drone in that moment.

9.3.2.5 TST - 3.3.3 - Motion Tracking

A square will be laid on the ground with painter's tape. The drone will be placed in each of the 4 corners and the accuracy of the x and y estimates are accurate within 5 centimeters. The x and y positions will be tested again, but with z coordinates corresponding to the highest point the drone can go in an indoor environment. Finally, the drone will be held at different poses ranging from 0-30 degrees and to verify that the motion tracking system detects the poses.

9.3.2.6 TST - 3.3.3 - Result

Rather than directly testing the UAV with the marker attached, due to the variability of the drone's movements and the difficulty of setting the whole stack up repeatedly for such a simple test, we performed the first test by holding the marker on the end of a thin tube and placing it in the 4 corners of a marked area. Then the ball was lifted up to get vertical measurements.

The resulting difference between the measured distances to the waypoints using a ruler, and using the tracking system is shown in Figure 36.

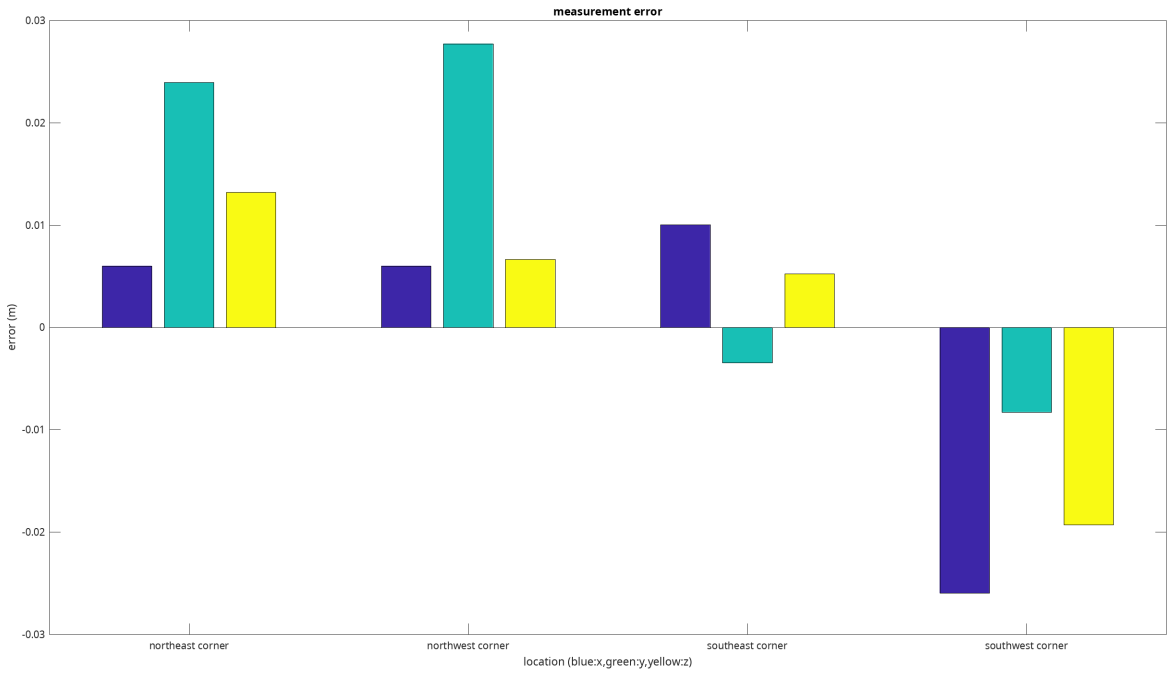

Figure 36: Measurement of the offset from the true marker waypoints and where the tracking system measured the marker to be in meters.

These ground tests performed quite well. The tests show the the measurements are well within a 5cm variance of the ground truth. However, when lifting the markers, shown in Figure 37, there was some variance between the ground truth and measured marker.

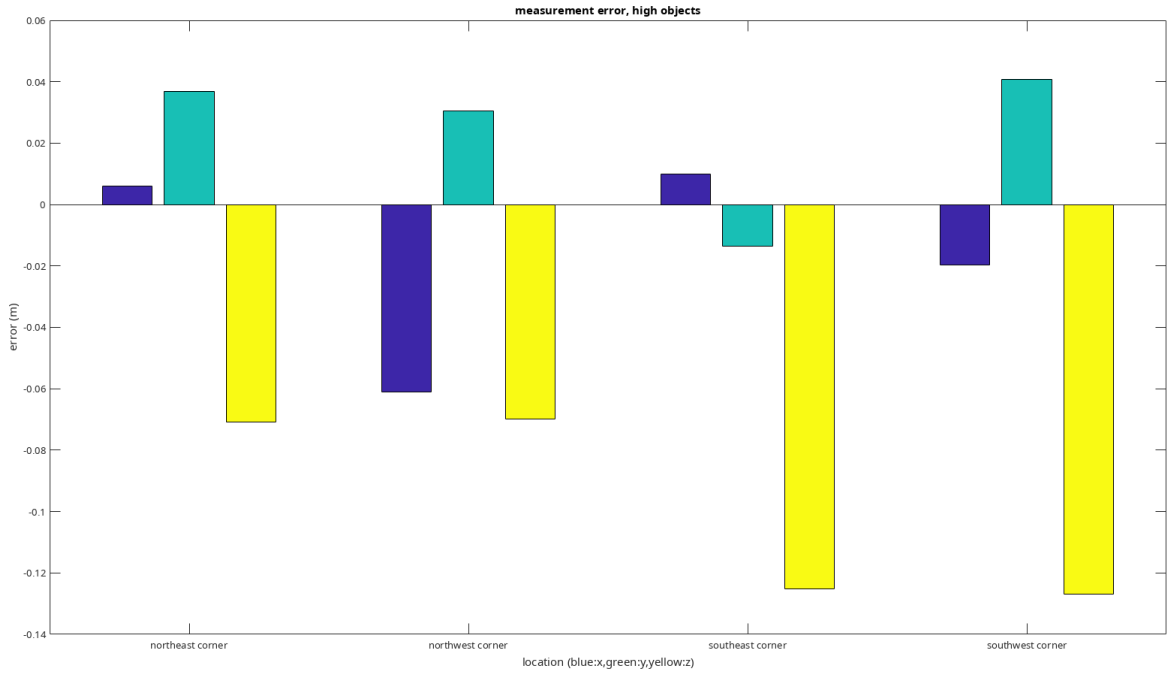

Figure 37: Lifted measurement of the offset from the true marker waypoints and where the tracking system measured the marker to be in meters.

This variance is likely not due to the Kinect's measurements, but rather variance in how the marker was held up. instead of being held completely straight, the marker could have very well been drooping to a side. This explains why in a more stable configuration (on the ground) the marker's position was less variant with the ground truth location.

Overall the tracker was very capable when it came to representing a marker's location in 3D space. Short of the marker going out of the camera's view, something obstructing the camera, or a large change in brightness, the tracker managed to create reasonably precise measurements for a single stereo camera with limited range.

9.4 Controller

9.4.1 Performance Criteria

9.4.1.1 The maximum jerk, acceleration, and velocity targets can be achieved

9.4.1.2 The controller does not destabilize due to setpoint oscillations

9.4.1.3 Automatic takeoff and landing can be accomplished

9.4.2 Test Cases

9.4.2.1 TST-3.4.2

A motion profile will be sent to the controller. The controllers ability to attempt to execute the motion profile will be checked by observing the control effort allocation during simulation.

9.4.2.2 TST-3.4.2-Result

The above test was accomplished as described. In Figure 38 a graph of the X Y and Z controllers tracking in simulation are shown. A screenshot showing the drone in the simulator is also shown in Figure 39.

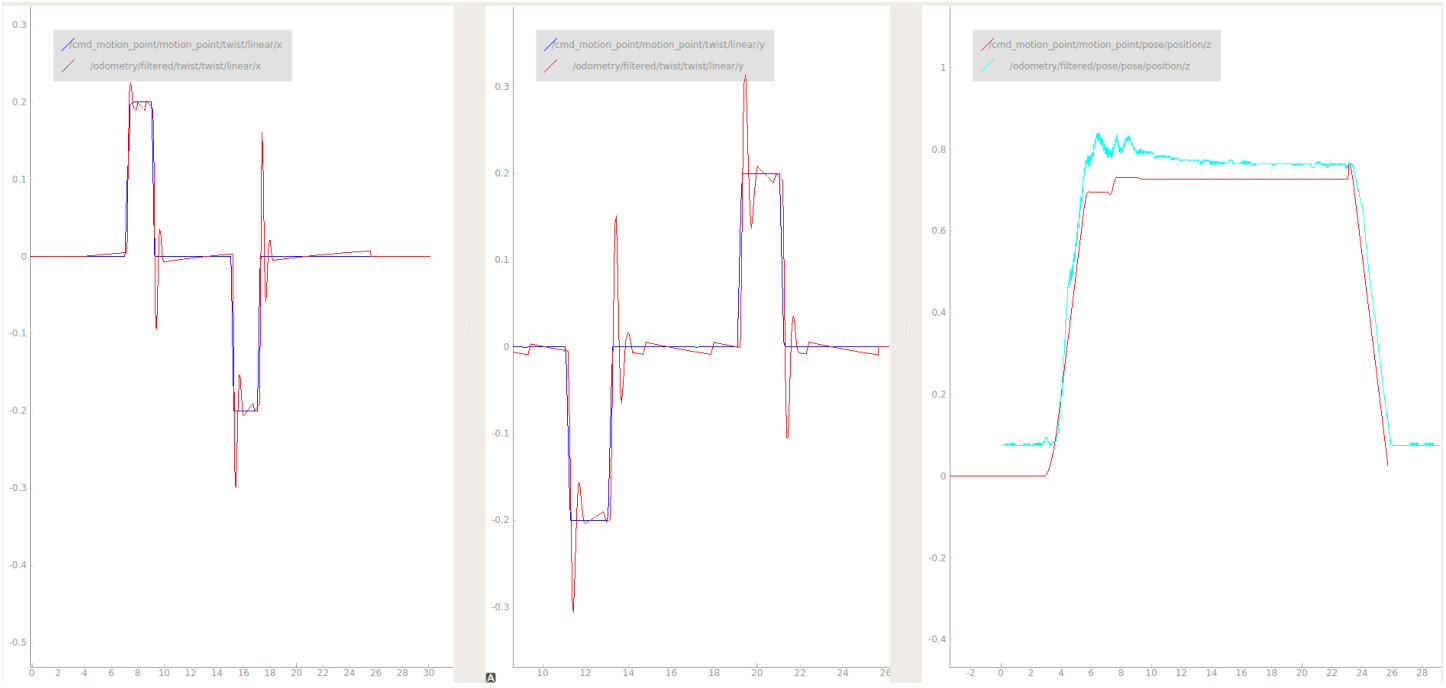

Figure 38: Controller successfully tracking velocity commands for X and Y and position commands in Z. All units are m/s or m as appropriate. The horizontal axis is in seconds.

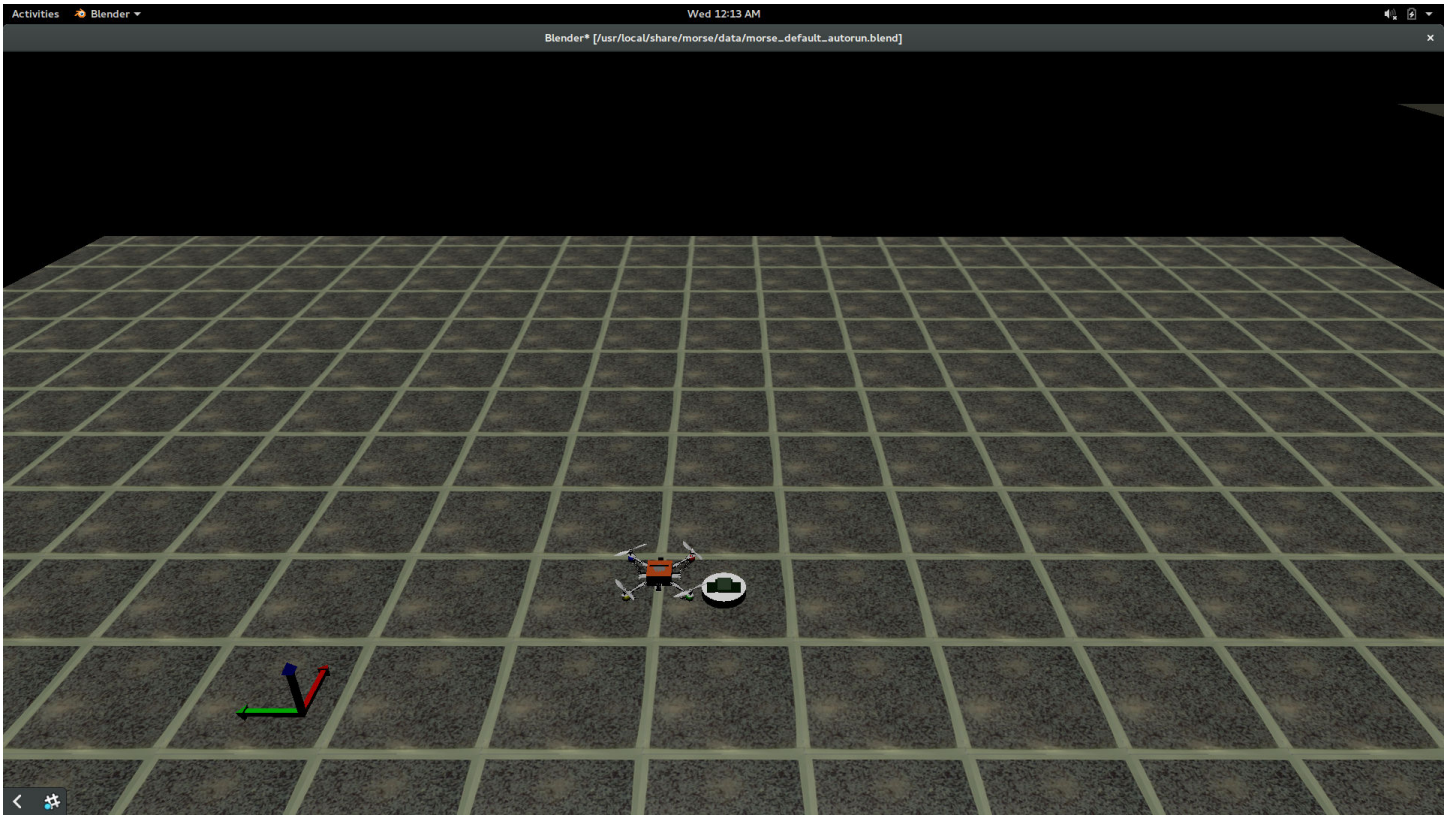

Figure 39: A screenshot of the simulator in action

9.4.2.3 TST-3.4.3-RC input

The RC Controller will be used to provide input to the ROS Stack and verify that Offboard and Stabilize mode can be entered on the pixhawk. While in offboard mode the RC inputs will be recorded in the ROS stack to verify that they can be received an interpreted as inputs.

9.4.2.4 TST-3.4.3-Result

Because a Pixhawk was not used for this project this did not have to be implemented or tested. This functionality was original implemented and tested with a Cleanflight flight controller by the RAS IARC team in 2016.

9.4.2.5 TST-3.4.4-Human Control

The RC Controller will be used to command automatic takeoff. Upon taking off the translation and vertical velocity will be set using the RC Controller. The commanded setpoints and the controllers ability to achieve them will be compared using localization's automatic error calculation statistics and verified to remain below the acceptable error limits.

9.4.2.6 TST-3.4.4-Result

This test was accomplished with the exception of an XBOX 360 controller being used instead of an RC Controller due to system architecture changes. Figure 40 shows the result of such a test. It can be seen that the controller minimizes the error between the target velocity within a fraction of a second with little overshoot. This was considered acceptable performance and confirmed that oscillations or destabilization would not be an issue in the X and Y directions.

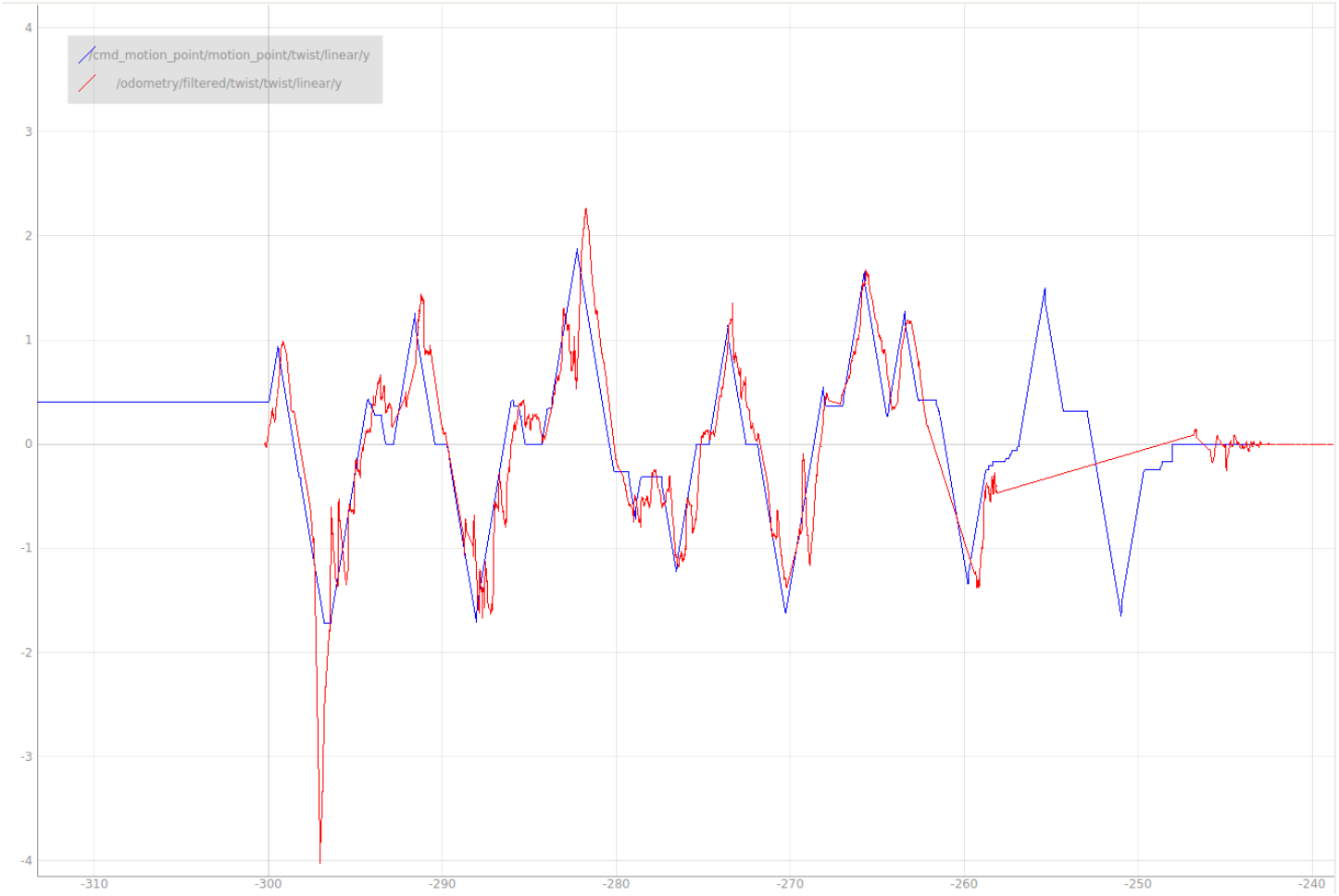

Figure 40: Tracking of velocities in the Y axis when commanded by an XBOX Controller when the drone is in joystick controlled mode

In the end there was not time to find and define the error limits. More post processing code would need to be written. Further, the velocity estimates were qualitatively identified to not be as good as shown in the above graphs. Thus, it was decided that derivation of maximum tolerance guarantees and velocity and acceleration would be a futile endeavour for the time being.

9.4.2.7 TST-3.4.5-Takeoff Simulation

Takeoff and the control outputs will be simulated by manually lifting the drone up and down. The control outputs will need to be proportional to the

velocity error.

9.4.2.8 TST-3.4.5-Result

Figure 41 shows the result of the aboe test. It can be seen that the throttle goes to 100% (1.0) due to the excessive difference between the commanded height and the desired height. Further when the drone is brought back down manually the throttle dips low to account for excessive height compared to the target height. For reference, the typical hover throttle is around 80%.

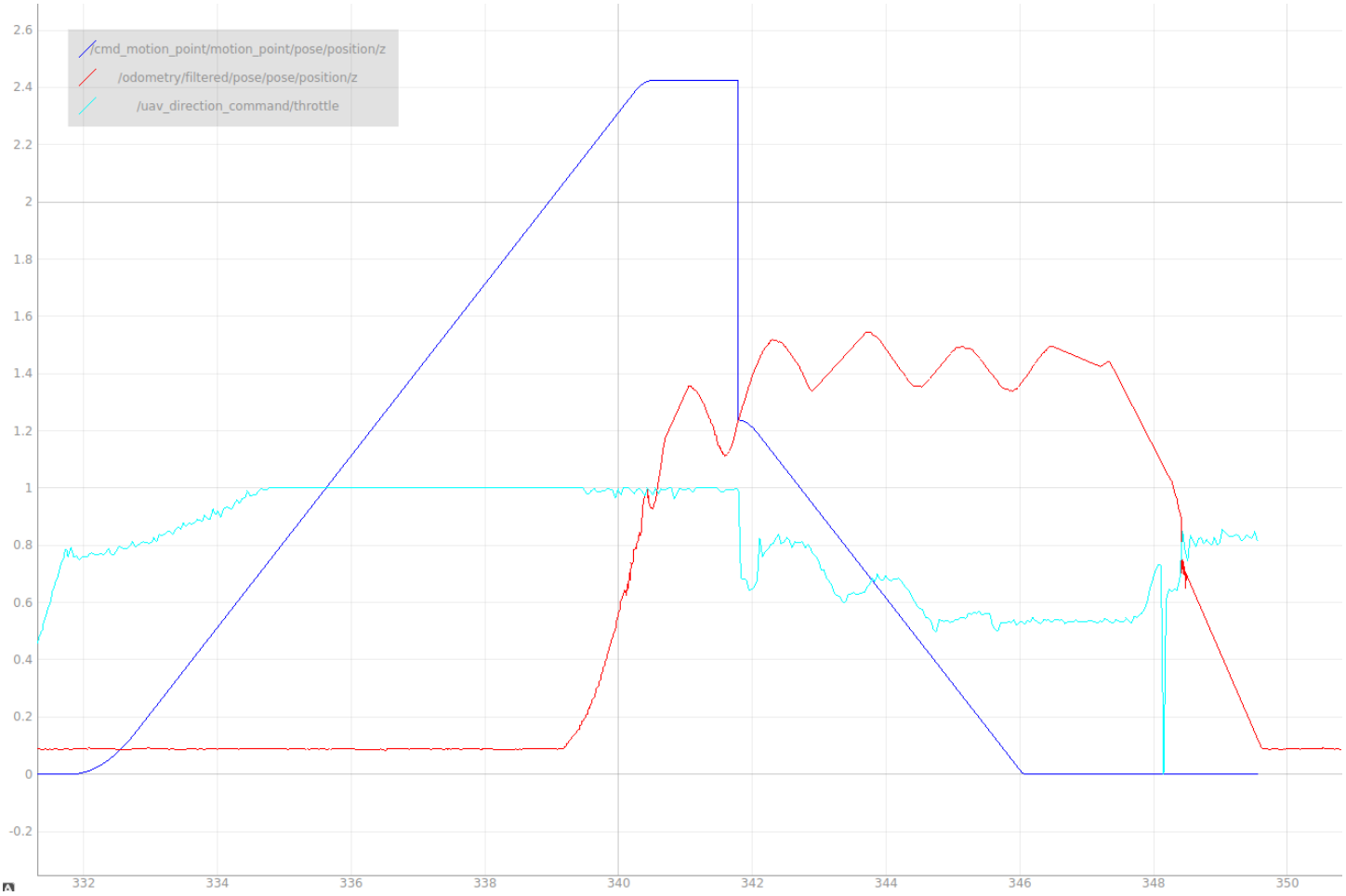

Figure 41: Result of testing height hold with a human lifting the drone up and down when the takeoff and land sequence is run. The intention was to sanity check the commanded throttles.

9.4.2.9 TST-3.4.6-Takeoff and Landing

Automatic takeoff and landing will be commanded. The commanded motion profile will be compared to the resulting localization data. The error will be compared to the tolerance guarantees.

9.4.2.10 TST-3.4.6-Result

Figure 42 shows the target height plotted with the achieved height and the main rotor throttle for an automatic takeoff and landing. It shows that the height controller was successfully able to achieve the target heights with above 0.5 seconds of lag. The source of this lag is currently unknown and will be the subject of future investigations. An interesting note is the height measurements became shaking around 0.8m the source of this oscillation will also be investigated.

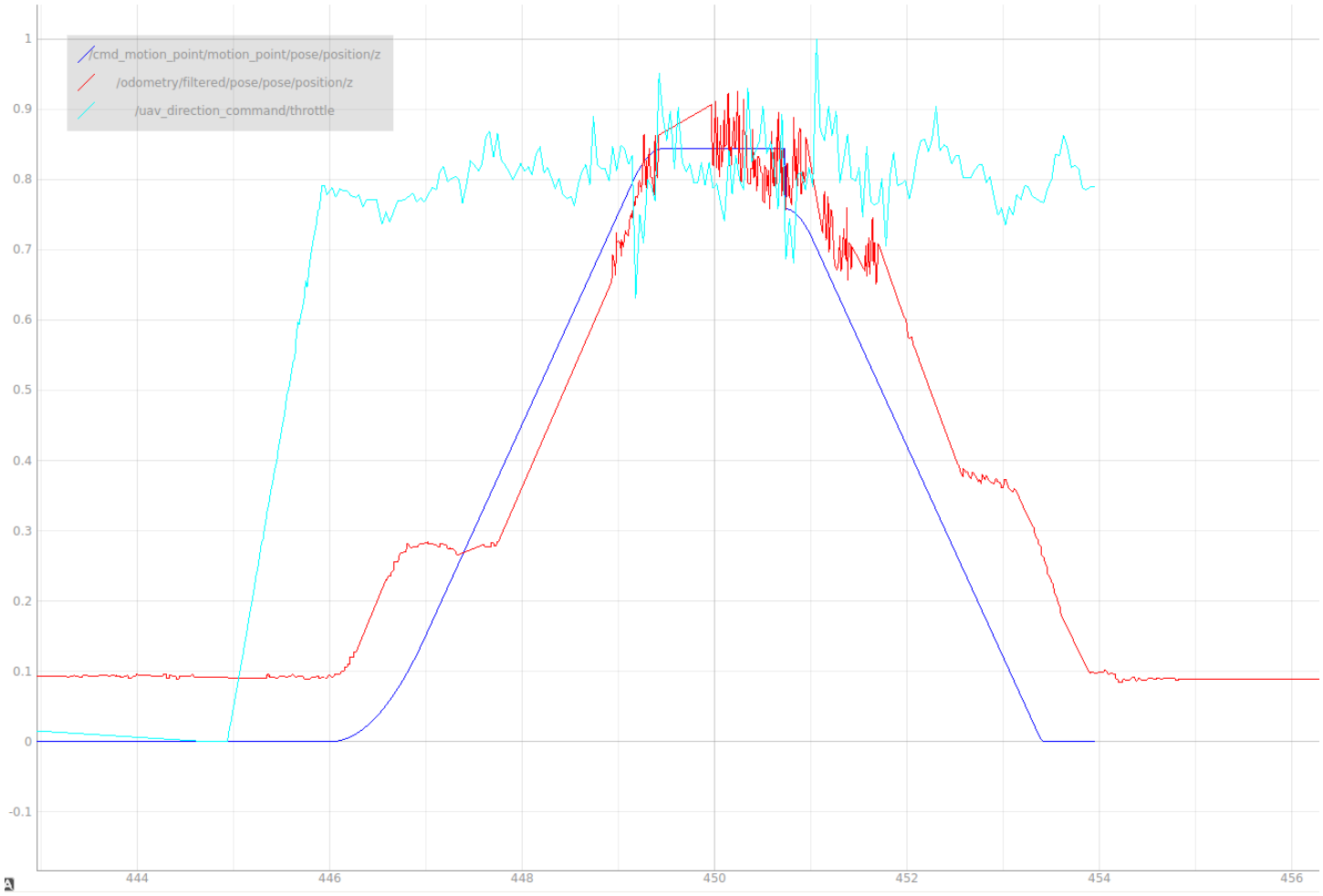

Figure 42: The result of an automatic takeoff and landing. The commanded height is plotted with the measured height as well as the throttle values for comparison.

Again, rated limits were not derived due to time constraints. The performance shown was qualitatively considered acceptable.

This test should have requested that hovering be performed between takeoff and landing. A graph of takeoff, hovering, and landing is shown in Figure 43.

Figure 43: Taking off, hovering, and landing showing oscillation in the height controller.

It can be seen that there is significant oscillation in the height hold. This will be addressed in future work but since the height hold controller was not within the scope of this project it will not be discussed further. The height hold controllers performance was one of the reasons for not eventually deriving rated limits.

9.4.2.11 TST-3.4.7

The rated limits will be tested by commanding motion profiles with progressively higher jerk, acceleration, and velocity. The error will be compared to the tolerance guarantees to ensure compliance.

9.4.2.12 TST-3.4.7-Result

Because the tolerance limits were not derived this test was not necessary.

9.4.2.13 TST-3.4.8-Ensure Craft will not destabilize

A square wave jerk signal will be used to generate a motion profile. The frequency and amplitude of the jerk will determine the peak acceleration, and velocity that the controller attempts to attain. Tests will be run with the amplitude of the jerk set between 0 and the maximum jerk limit. The frequency will be varied at each setpoint between 0 and the frequency that attains a maximum acceleration or velocity limit. The controller needs to remain stable and continue to track the motion profile for all tests.

9.4.2.14 TST-3.4.8-Result

Due to the noise in the velocity estimates it was not possible to use higher velocities or jerks than those shown in the above tests. Because of that, this test would not have revealed any useful information, so it was skipped.

10 Project Timeline

Date	Goal
February 5th	All materials for Aerial Platform were ordered
February 26th	1st Checkoff Switched from PX4 to Cleanflight Board Begin static thrust testing Interfaced VL53L0X and TFMini with the Nano Drone was rebuilt from the ground up with Jetson TX1 Static thrust tests to select propellers were performed Dynamic thrust data was collected Schematic for the Power Distribution Board was completed Added V1.5 drone to the simulator as well as depth cameras
March 14th	2nd Checkoff Designed Ekill and Jetson attachment hardware Rewrote IARC's Low Level Motion to support Motion Profiles Finished automatic derivation tools for dynamic thrust model Developed ground truth algorithm Developed full test plan for the power distribution board Optimized AVR code to save memory
March 28th	Accomplished autonomous takeoff and land Quantified improvements due to the dynamic thrust model Optimized sensor code for timing constraints Wrote initial implementation optical flow translation node Rebuilt drone with Raspberry Pi Finished assembly of power distribution board Finished implementation of camera tracking node Wrote look ahead motion profile generator
April 16th	After an all nighter all bugs finally fixed, drone could fly in a square autonomously Added velocity control using XBOX 360 controller
April 19th	Poster and Design Expo
April 20th	Final Demonstration Demonstrated autonomous height hold and translation with side rotors. Success!

Table 1: Milestone Schedule

11 Conclusions and Future Work

In the final demonstration the drone was able fly autonomously, hold height, and translate without tilting. It was able to operate in both a fully autonomous and a joystick controlled velocity hold. Because this was accomplished the project can be considered a success.

A number number of smaller "stretch goals" were not achieved (such as 10 minutes of flight and live accuracy statistics updates), but all main objectives were accomplished such as the drone was able to fly autonomously in a square without any human input and stable controller operation. Most of the smaller goals were not completed not due to inadequacies in the design but time constraints. The team does not feel the ability to hit the small goals are a reflection of the work put in but due to their inability to size the project appropriately at the beginning of the semester.

Things that should have done differently include having more than one person review the power distribution board before it was sent out for production. There were multiple minor issues with the board (such as the reset mechanism on the AVR chip and the digital isolators malfunctioning after a certain amount of time) that would have cost the team much less time to find initially than it took to fix them after they became a problem. It also would have helped to characterize our optical flow board earlier, as a lot of time was wasted trying to evaluate velocity measurements where the underlying math was flawed.

Things that were done correctly include using simulation to evaluate and test the controller design before testing on the physical drone. This allowed many crashes to be avoided and bugs to be found in the thrust model interpreter. Another thing is that in the case of the marker based tracking system, the literature review done prior to the implementation proved extremely valuable. Without that review the tracking system would not have been nearly as robust.

The biggest lesson learned was that projects have to be scoped appropriately for the time allotted. This project showed the team members exactly what their best work is and how much they could accomplish in a fixed time.

If this project became a commercial product it would need significant R&D resources to be robust and cheap enough for mass production. For instance, the Raspberry Pi would not likely remain, or would at least become an integral part of the circuit board. Additional safety mechanisms such as prop guards would also be necessary. Finally, a smaller circuit board, stronger

motors, longer lasting batteries, and several cameras would be necessary to make this project a fully featured commercial robot.

This project will continue as part of the RAS IARC project. We will focus on achieving more stable control of height, proving the accuracy of sensor measurements with higher rigor, and increasing the accuracy of velocity estimates.

It works! We can't believe it actually works!

12 References

References

[1] "Banggood rs2205s product listing." https://www.banggood.com/Emax-RS2205S-2300KV-Racing-Edition-Brushess-Motor-CW\-CCW-for-FPV-Racing-p-1093090.html?cur_warehouse=HK.

[2] "Marinefe high performance 8mm bullet connectors w/ molded grip plug product listing." https://marinefe.com/product/high-performance-8mm-bullet-connectors-w-molded-grip-plug/.

[3] "Deans connector - m/f pair." https://www.sparkfun.com/products/11864.

[4] E. Kaufman, K. Caldwell, D. Lee, and T. Lee, "Design and development of a free-floating hexrotor uav for 6-dof maneuvers," in *2014 IEEE Aerospace Conference*, pp. 1–10, March 2014.

[5] G. Jiang and R. Voyles, "Hexrotor uav platform enabling dextrous interaction with structures-flight test," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–6, Oct 2013.

[6] S. Rajappa, M. Ryll, H. H. Blthoff, and A. Franchi, "Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4006–4013, May 2015.

[7] M. Odelga, P. Stegagno, and H. H. Blthoff, "A fully actuated quadrotor uav with a propeller tilting mechanism: Modeling and control," in *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 306–311, July 2016.

[8] M. Ryll, H. H. Blthoff, and P. R. Giordano, "Modeling and control of a quadrotor uav with tilting propellers," in *2012 IEEE International Conference on Robotics and Automation*, pp. 4606–4613, May 2012.

[9] D. Brescianini and R. D'Andrea, "Design, modeling and control of an omni-directional aerial vehicle," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3261–3266, May 2016.

[10] S. Park, J. Her, J. Kim, and D. Lee, "Design, modeling and control of omni-directional aerial robot," in *2016 IEEE/RSJ International Con-*

ference on Intelligent Robots and Systems (IROS), pp. 1570–1575, Oct 2016.

[11] D. R. McArthur, A. B. Chowdhury, and D. J. Cappelleri, "Design of the i-boomcopter uav for environmental interaction," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5209–5214, May 2017.

[12] A. Albers, S. Trautmann, T. Howard, T. A. Nguyen, M. Frietsch, and C. Sauter, "Semi-autonomous flying robot for physical interaction with environment," in *2010 IEEE Conference on Robotics, Automation and Mechatronics*, pp. 441–446, June 2010.

[13] H. Romero, S. Salazar, and R. Lozano, "Real-time stabilization of an eight-rotor uav using optical flow," *IEEE Transactions on Robotics*, vol. 25, pp. 809–817, Aug 2009.

[14] O. Liang, "Top 5 best motors for mini quad." `https://oscarliang.com/top-5-best-motors-mini-quad/`, Jul 2017.

[15] R. Harnell, "Emax rs2306 2400kv." `https://www.miniquadtestbench.com/emax-rs2205s-2300kv.html`, 2016.

[16] R. Harnell, "Emax rs2306 2400kv." `https://www.miniquadtestbench.com/emax-rs2306-2400kv.html`, Jul 2015.

[17] "Trace width calculator." `http://www.4pcb.com/trace-width-calculator.html`.

[18] "Common power module." `http://ardupilot.org/copter/docs/common-3dr-power-module.html`.

[19] "Orbitty carrier for jetson tx2 and tx1 user guide." `http://www.connecttech.com/pdf/CTIM-ASG003_Manual.pdf`.

[20] "Powering the pixhawk." `http://ardupilot.org/copter/docs/common-powering-the-pixhawk.html#common-powering-the-pixhawk`.

[21] J. Y. Baek, S. H. Park, B. S. Cho, and M. C. Lee, "Position tracking system using single rgb-d camera for evaluation of multi-rotor uav control and self-localization," in *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1283–1288, July 2015.

[22] "Ekf block diagram." `https://www.researchgate.net/profile/Alexander_Sendobry/publication/313663074/figure/fig21/AS:461371751047190@1487011005696/`

Figure-6-Block-diagram-of-an-Extended-Kalman-filter\
-EKF-without-input-quantities-to.png.