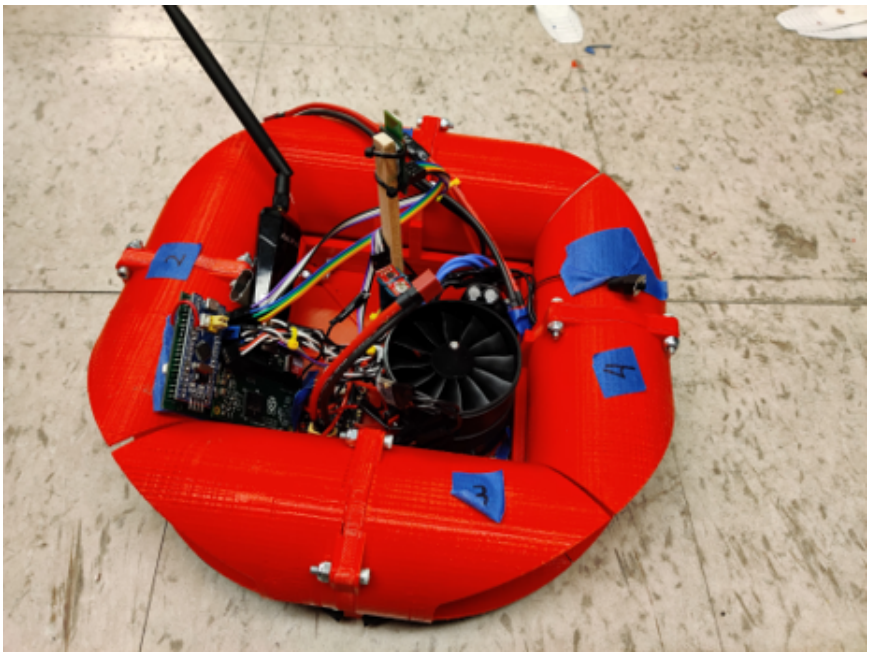
Team #26: Computation, Localization, and Power Systems for Distributed Robotics
ECE445: Final Report

Tanitpong Lawphongpanich (tl6) & Lyle Regenwetter (regnwttr)
TA: Jon Hoff

December 12, 2019



Contents

1 Introduction

1.1 Objective

As technology continues to advance and become more accessible to the general public, drones and autonomous vehicles are becoming commonplace in modern society. For years already, drones, specifically quad copters have been a common and widespread hobbyist item. Quad copters are also the foundation of a lot of swarm coordination research. The widespread

prevalence of quad copters can largely be attributed to the availability of all-in-one drone controllers and electronics packages that can be applied to almost any custom quad copter platform. This allows hobbyists to build quad copters with limited technical experience and allows drone coordination researchers to focus on algorithms and planning rather than physical hardware. Considering the revolution in cost and availability that these all-in-one systems have provided for the quad copter industry, it is surprising that nothing has been made for ground-based robotic systems.

The goal of the Computation, Localization, and Power System for Distributed Robotics project was to develop an all-in-one electronics package for ground-based robotics projects. The functionality of the system is similar to many of the commercially available all-in-one drone electronics systems, supplying power, controlling the motors, and localizing the craft. Unlike drone systems, though, the system is optimized for ground-based robotics. This difference is especially pronounced in the design of the craft's localization system. While many aerial robots use GPS to localize in a large, unbounded space, the ground-based system provides much higher accuracy in a closed, preset space with localization anchors around the area. This will give researchers and hobbyists localization accuracy previously unseen in an all-in-one system. Additionally, we included several features not commonly found in all-in-one systems. For example, we incorporated a backup battery system that keeps the main computer on so that code can continue running and the computer doesn't need to restart.

1.2 Background

This project originated as a research project for Professor Geir Dullerud. He tasked us with updating an old project called HoTDeC, which was an autonomous hovercraft platform from 2004 [1]. The HoTDeC is a perfect example of an autonomous ground-based system developed for research that could hugely benefit from an all-in-one system. Back in 2004, a cutting edge robotics initiative took several researchers many years to develop, and cost thousands of dollars per unit. As discussed, the booming hobbyist drone industry has proliferated consumer-targeted mobile robotics hardware. All-in-one drone controllers like the Emlid Navio2 are now commonplace. Despite this fact, almost all integrated drone systems are designed for quadcopters and applying one of these to a ground-based robotics project, such as Professor Dullerud's HoTDeC's would be inefficient. Existing systems that were developed with ground-based robotics in mind often come with significant drawbacks. The Facebook PyRobot system, for example, runs on ROS, which makes quick development much more difficult, especially for a casual user. More imporotnatly, it is physcially bulky, running on an x86 Intel NUC PC. Another platform, the iBase, lacks localization systems and does not support high speed motors, which drastically limits the applications of the system.

Developing an affordable, open-source, all-in-one unit drastically lightens development costs and times for researchers who would prefer to focus on control, vision, and path planning, rather than toiling over hardware. It makes drones and robotics accessible to kids of a younger age and allow them to cultivate a passion for engineering and problem solving earlier in life. Finally, it allows hobbyists who would typically be constrained to quad copter development to branch out into mobile ground-based robotics.

1.3 Implementation

Though we made conscious efforts to keep the electrical system as cross-platform compatible as possible, some design decisions had to be made for the specific implementation, namely the HoTDeC hovercraft. The HoTDeC implements many key mechanical innovations that are rather unique to its specific mode, like custom bidirectionally-optimized propellers and a rubber air bearing levitation system. Almost all of the electrical components, however, should be able to be applied to a different robotic system with minimal additional development resources.

2 Theory

2.1 Basis Vector Thrust Decomposition

One key component of the theory of the system is a method to decompose intended accelerations vectors into individual thrust vectors. In other words, this takes the robot's intended acceleration and converts it into the necessary forces that each actuator needs to exert. In the HoTDeC configuration, there are four motors controlling the lateral motion and acceleration of the craft. In fact the HoTDeC's basis vector system is shared by numerous types of ground-based robots, including a very standard robotic drivetrain system often used in mobile robotics competitions such as FIRST robotics competition. This drivetrain is commonly referred to as the "Holonomic Drive," [2] though the holonomic qualifier actually classifies a much larger group of drivetrains. The free body diagram below shows the individual force decomposition used by the HoTDeC and the "Holonomic Drive":

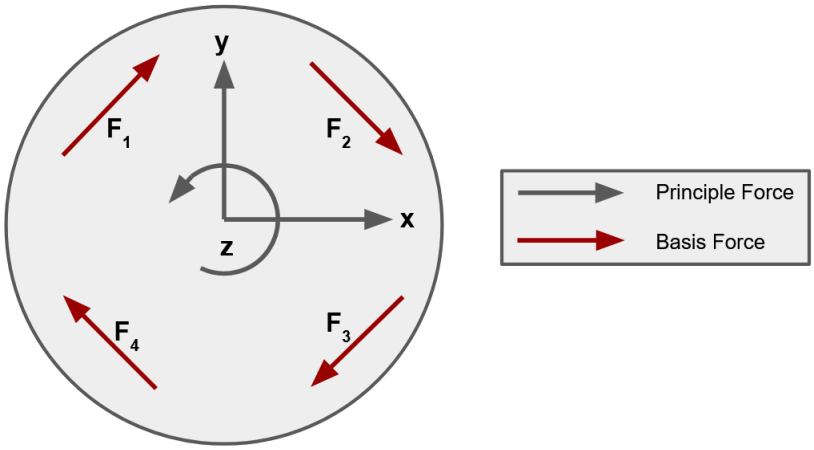


Figure 1: Free body Diagram

The solution to the system and the algorithm to compute the basis vectors are included in the appendix.

2.2 Kalman Filter

Kalman filters come in many variants, and selecting the appropriate type to design was largely influenced by availability of computing resources and differences in sensor timings. At the most basic level, Kalman filters can be divided into two principle categories: standard Kalman Filters, which assume system linearity, and Extended Kalman Filters which don't assume linearity and evaluate complex nonlinear relationships among characteristic matrices [3]. In the interest of computing power, the standard linear Kalman Filter was selected for the project, largely because the system should operate very linearly. Additionally, the ability to be able to run different correction steps for the different sensors was desireable. The magnetometer/gyroscope/accelerometer IMU unit, for example has a maximum possible data output rate of 800 Hz [4], but we decided to run at 100 Hz. The Decawave ultrawide-band system, however, was run at 10 Hz. Since sacrificing data is inefficient, the ability to process different types of sensor data at different times was crucial. Additionally, due to limited computing resources, the filter was constrained by the complexity of calculations necessary to execute each step. When considering the computing demands for individual steps in the Kalman filter, the algorithm flow consists exclusively of simple addition, transverse, and multiplication, with the exception of a single matrix pseudo-inverse operation. This occurs in the computation for Kalman gain [5]. As demonstrated in the appendix, this Kalman gain can be static and predetermined with the sacrifice of certain design freedoms. This would allow the gain to be precomputed and written into the firmware instead of running this tedious pseudo-inverse during each time step during filtering. This requires non-adaptive preset expected noise values, where dynamic noise estimation would otherwise be used. The

mathematics behind the Kalman filter are rather lengthy and are included in the appendix, for optional reference.

2.2.1 Air Bearing Principles

The HoTDeC uses an air bearing for levitation. Air bearings distribute a modest amount of pressure to the whole underside of an object. This allows an object to be lifted with minimal pressure. In fact, the fluid pressure necessary to lift an object using an air bearing scales linearly with the ratio of air bearing surface area to pump propeller arc surface area, relative to atmospheric pressure:

$$P_{bearing,min} - P_{atm} = (P_{prop} - P_{atm}) * \frac{A_{proparc}}{A_{bearing}} \tag{1}$$

For the HoTDeC's surface area ratio, Equation 1 translates into approximately 92% less lift force needed as compared to conventional vertical thrust lift devices, like quadcopters for example. Of course, without the additional consideration of airflow, an air bearing could theoretically be infinitely efficient. The air bearings on the hovercraft have an additional constraint that requires some minimum airflow and limits the efficiency cap of the bearing. Specifically, the HoTDeC air bearings rely on a small non-contact distance between the bearing surface and the ground which almost entirely eliminates physical solid-solid contact friction. This air layer is difficult to quantify explicitly though fluid mechanical properties, and was instead tested empirically.

3 Design

Original requirements and verification are included in the appendix. Unless otherwise stated, verification procedures were carried out as described to assess completion of requirements. Deviation from the original requirements or verification will be noted in the following sections.

3.1 Mechanical Design Fabrication

Key physical components include the chassis, the air bearing, and the custom propellers.

3.1.1 Bidirectional propeller design

Propellers are typically optimized for optimized performance in a single direction. Because the propellers on the HoTDeC craft need to perform with equal efficiency in either direction, conventional propeller blade profiles would not suffice. Consequently, we took on the challenge of designing custom bidirectional propellers. These propellers use a symmetrical

blade profile to guarantee symmetric performance. Propeller modelling was done in CREO Parametric, geometry was tested in Ansys Fluent computational fluid dynamics software, and then several prototypes were manufactured with Selective Laser Sintering (SLS). Each prototype was measured for linear thrust produced with a 3300 kv 30 mm brushless three-phase motor at 30% throttle. The top performer yielded approx 1.3 N of force, and was selected for use on the craft. Shown below in Figure 3.1.2 is one of the propellers printed for testing.

Figure 2: Propeller

3.1.2 Air Bearing Fabrication

Since the air bearing required carefully designed geometry, a flexible material, and a reasonable tensile strength, a fabrication process to create the bearing was developed. A mold incorporating the bearing geometry was created, 3D printed in PLA, and systematically sanded to minimize surface roughness. Next, several layers of teflon spray were applied to the mold to minimize sticking effects. Finally, Plast-Dip rubber spray was applied in 11 even coats and allowed to dry for approximately 1 hour between coats and for 24 hours after completion. The bearing was then peeled from the mold and holes cut in the underside for air escape pathways.

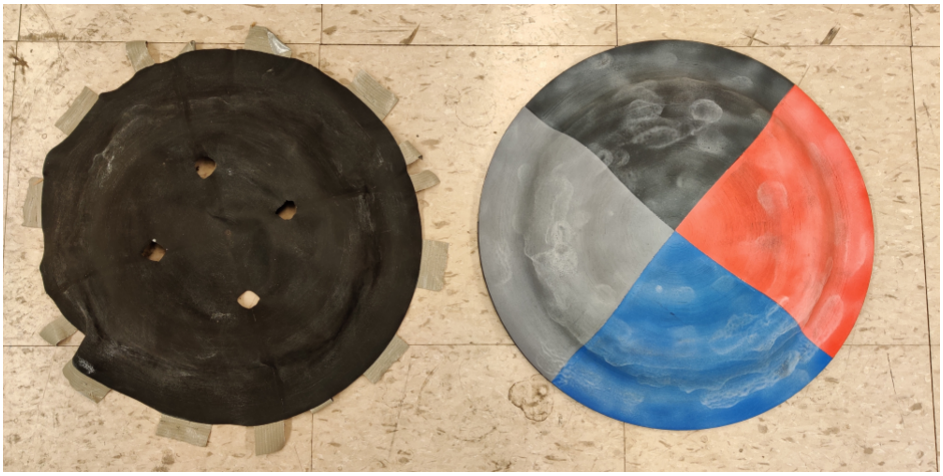


Figure 3: Air Bearing (left), Air Bearing Mold (right)

3.1.3 Electronic Speed Controllers Motors

Off the shelf BLHeli_32 1200 MultiShot electronic speed controllers were selected to convert PWM signals to synchronous 3 phase power to brushless 3-phase AC motors. Motors are conventional drone motors with relatively high Kv (rpm per volt). They were taken from an old project and the exact part number is unknown.

3.1.4 Requirements Verification

The requirements and verification for the physical components can be found in the appendix. Temperature was not explicitly measured in the verification process, but a verification that the PLA never reached glass transition temperature was carried out instead through qualitative observation and physical probing. Additionally, based on results of a heat transfer analysis, the propellers were designed to funnel airflow through the motor cavity for additional cooling, as the original propeller design didn't meet the requirement since it restricted airflow through the motor cavity. This is summarized in the appendix alongside the requirements.

3.2 Chassis

3.2.1 Functionality

The chassis is the frame of the HoTDeC encompassing the electronic components and mounting key mechanical features. The chassis was designed to be printed on Lulzbot TAZ-6 printers out of Polylactic Acid (PLA). It consists of 14 pieces and is designed to be highly modular and easily assembled. There were some moderate issues with surface roughness,

especially inside the propeller tubes, and a few issues with part warping due to nonuniform print bed heating, which were both out of our control.

3.2.2 Requirements and Verification

All verification for the chassis were met, save for two. One stated that no wires could stick out of the electronics housing. For this iteration, we did not construct a top plate to fully enclose the casing, so this requirement was not met. Additionally, one governing the resistivity of PLA was found using a slightly altered verification using a small thin strip of PLA and was confirmed with researched values. Regardless, we confirmed that the requirement was met.

3.3 Sensors

3.3.1 Inertial Measurement Unit

The Inertial Measurement Unit (IMU) is an off the shelf 9DOF Stick utizling LSM9DS1 chip from ST Microelectronics. It contains a 3 axis magnetometer, a 3 axis gyroscope, and a 3 axis accelerometer. Due to the limited degrees of freedom of the craft, only magnetic heading, z axis angular velocity and x and y accelerations are used. IMU data was read through I2C from the STM32 at 100 Hz.

3.3.2 Ultrawide-band

The ultrawide-band sensors implemented a Decawave DWM1000 module (including DW1000 chip and antenna unit). Two were set up in the environment at known locations and one was fixed on the craft. Efforts were made to maintain line of sight between the anchors and the sensor on the craft, to improve quality and accuracy of readings. The localization accuracy for any given reading was found to fall in a 3 cm radius. DWM1000 chips were implemented on a simple PCB with supporting filter capacitors and a voltage regulator, shown in Figure 3.3.2. Decawave data was sampled at 10 Hz. Shown below is an image of the Decawave board built for the localization system.

Figure 4: Decawave Board

3.3.3 Requirements & Verification

Sensors were one key area where the finalized system ended up deviating noticeably from requirement and verification procedures. Perhaps the most significant of these was the fact that the magnetometer wasn't able to return any valuable data. As we discovered after a period of extended confusion, MEL 2204, the room that the system was tested in, seems to have iron bars for structural integrity built into the floor of the building. These iron bars create spurious magnetic fields and certainly interfere with the earth's natural magnetic field. This completely invalidated magnetometer data. Another set of modifications was the removal IMU "drift" constraints. These requirements were designed before a detailed understanding of the Kalman filter's functionality was attained. Since "drift" in a sensor reading is encompassed by the overall state estimation of the filter, the actual "drift" is rather immaterial. Additionally, the whole terminology of "drift" as defined in the Design Document was rather vague, and generally irrelevant. As such, we made no effort to specifically evaluate these requirements.

4 Power

4.1 Subcomponents

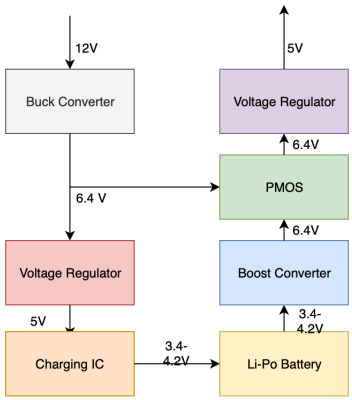


Figure 5: Power systems block diagram

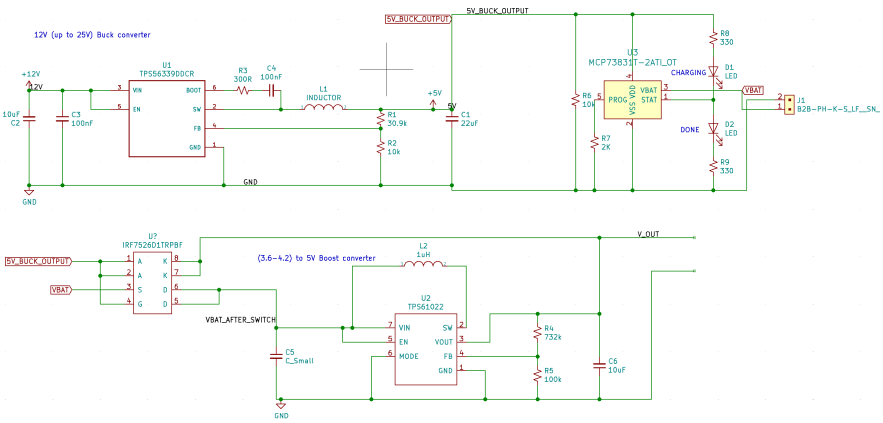


Figure 6: Power systems schematics

4.1.1 Batteries

The craft is powered by the main three-cell Lithium-Polymer (Li-Po) batteries, which supplies 12V power directly to the ESCs to power the motors. The main 12V line is passed through several buck converters and regulators before feeding into the electronics systems. A one cell Li-Po battery is utilized as a backup battery for the electronics to support hot swapping

4.1.2 Buck Converter

We utilize one buck converter in our system to step-down the 12V to 6.4V power for the electronics systems and charging IC for backup batteries. The converter is a switching regulator with around 86% efficiency.

4.1.3 Boost Converter

We utilize one boost converter in our system to step-up the 3.6-4.2 V from the backup battery to 6.4V for power electronics. The converter is a switching regulator with around 60% efficiency.

4.1.4 Charging IC

The charging IC for the backup battery uses TPS561022, a single-cell simple lithium-polymer battery charger, to charge the backup battery. The backup battery has voltage range from 3.6 to 4.2, which is the reason why we need to have a boost converter to step-up the voltage before feeding into the 5V electronic systems.

4.1.5 Switching PMOS

Since we want to allow battery hot-swapping, we need to have a mechanism to switch between the main 12V power and the backup battery. We utilize IRF7526, a small-packaged integrated PMOS and diode, to switch between the main and backup power. The gate of the PMOS is connected to the 6.4V power line with a 10k Ω pull-down resistor, therefore, when the main power is disconnected, the pull-down resistor will pull the gate pins of the PMOS low, consequently turning on the PMOS and connecting the battery to the boost converter to provide the power to the electronics systems. On the another hand, when the main power is connected, the PMOS will turn off, disconnecting the backup systems from the main 5V electronics power.

4.1.6 Regulator

A simple voltage regulator is used to output a consistent 5 volts from the potentially noisy 6.4 volt input.

4.1.7 Requirements and Verification

The majority of the power system was redesigned, so many of the requirements and verification of the specific components are no longer relevant. Though there are a number of the same components, their configuration in regards to one another is largely modified. since

many of the requirements relate to the specific voltages of the components for the old configuration, most are irrelevant. There are, however, some higher level requirements from the original set that were met and verified, and some that were not. Some of the key requirements that could have been met considering the design changes but weren't met includ the switches and E-stop button. We neglected to order an E-stop button and ended up using a piece of rope tied to the craft as a safety measure. We ended up determining that this was safer than the E-stop, since if the craft spins out of control, the E-stop will be difficult to press. In the future, an E-stop may be implemented as well, as an easy method to manually disconnect short circuits. Additionally, there is only a switch for the main thruster, one for the backup battery, and one for the mainboard, and no overall switch. This was done so that power could be supplied to components individually for separate testing.

4.2 Computation and Communication

This systems consists of one board, which we will hereon call the "mainboard". It hosts Raspberry Pi computing module and a STM32 development board and provides connections to the rest of systems (sensors and escs). Most of them utilize generic 2.54 mm pins to connect with these signal wires, which is the same size as breadboard wires and most connections in drone/mobile robot systems.

For connections, we have a SODIMM connector for the Raspberry Pi compute module, which has a 200 pins breakout into 200 SMD pins. Additionally, a Generic STM32F103C8 microcontroller board with 42 pins is mounted with through hole connections. The board takes in 5V via micro USB port. There are 2 switching power converters on the board, which convert the 5 volt input to 3.3 and 1.8 volts. These are required to power the Raspberry Pi compute module, while STM32 board contains on-board power regulator that takes in 3.3 V from the direct micro usb input.

There is no need for level shifters or photo-decoupler since all of the systems on this board communicate through 3.3V signal, even though there are some external 5V signals like those from ESCs, they are connected with the 5V tolerant pins on the STM32. The goal of this design is to keep things as simple as we could.

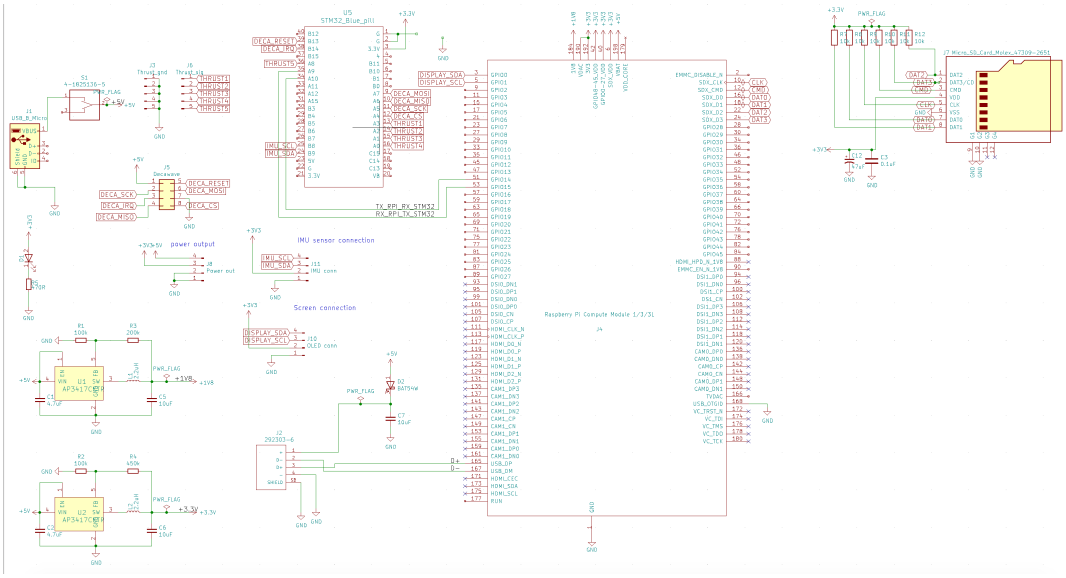


Figure 7: Computing systems schematics

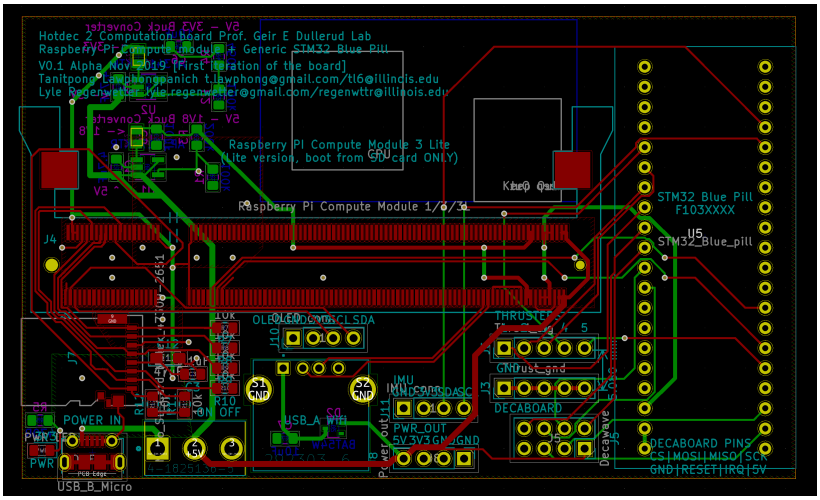


Figure 8: Computing systems board layout

4.3 Subcomponents

4.3.1 Raspberry Pi Compute module

This is our "brain" of the system. The raspberry pi contains ARM-Cortex M7 Quad core processors and runs linux operating systems. The raspberry pi is used for high-level and computation-intensive tasks such as path planning or communication over the network.

4.3.2 STM32 Microcontroller

The Microcontroller (MCU) is used for low-level real-time tasks such as sampling the sensors or outputting PWM wave forms/serial (depends on type of ESCs) to control the ESCs. We run bare metal C code here since there is currently no need for real-time operating systems (RTOS), but the chip could run real-time operating systems if needed.

4.3.3 Wifi module

For the sake of simplicity, we use an off-the-shelf usb wifi adapter plugged into the board's female full size usb A port to provide the raspberry pi a network connection. During development and testing, we connect to the board over the wifi through ssh to send commands and run codes.

4.3.4 Requirements and Verification

The high level requirements of this system are mostly non-quantitative since we put most of the quantitative requirements such as sensor sampling rate or voltage stability under the power and sensors section already. The main requirement of the systems is to work together as a whole to control the whole system. The first requirement that we checked is the board can boot up and accessible over wifi through ssh. All requirements were eventually tested either through direct verification as proposed or though logical deduction of functionality through more advanced comprehensive tests. One issue encountered is that the STM32 boards struggled to send all sensor data without corrucpiing PWM timers. We suspect that genuine boards will correct this issue, as we likely received knockoffs. If it is not resolved, we will program the STM32s with the ST toolchain, which will resolve the issue.

4.4 Software

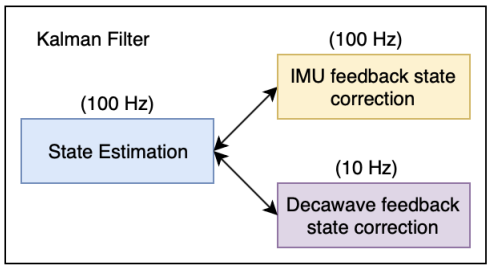


Figure 9: Software Kalman filter system

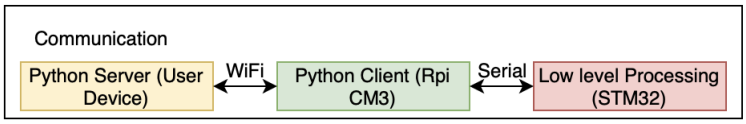


Figure 10: Software communication/command system

The Raspberry Pi runs Linux and the microcontroller runs C code. In order to create a systems that is easily configurable while still able to operate in real time. We utilize python as our main programming language and low level C code in the STM32. The reason why python was chosen was its ease of programming, especially for mathematical calculations and its relatively good efficiency compared to other math scripting languages. Real time tasks such as sampling data and outputting PWM waves are computed in the microcontroller.

4.4.1 Kalman Filter

As mentioned earlier, the system needs to reliably fuse data from each sensor together. Details of the precise design and operation of the Kalman Filter are rather lengthy. The mathematics of the Kalman filter is included in the appendix. Currently the Kalman filter is written in Python, for the ease of development and simulation, which is not optimal since Python is not designed for real-time tasks. In the future, we planned to move the Kalman filter into the microcontroller since the MCU is better for running real-time tasks than normal Linux systems

4.4.2 Communication/command system

The flow of the command systems is as follows: centralized server on a user's laptop running python takes in user input from joysticks, then sends the command over the wifi to the Raspberry Pi on the craft. Next the craft will send commands through serial port to the microcontroller. For example, a user can control the craft using a joystick which is connected to his/her laptop, and the code on his laptop will send a command packet over the network using Python's pickle library to the Python code on the raspberry pi. The Raspberry Pi then sends a command through JSON via serial wire to the microcontroller
For autonomous operations, the user can directly program a path planning algorithm or any control algorithm using python on the hovercraft, in which we already provide the python API to send/receive command and sensors data from the craft.

4.4.3 Requirements and Verification

Unfortunately, due to issues with sensor data from our STM32 boards, we were unable to run the Kalman Filter on real data. We were, however, able to test its performance on simulated

data. The Kalman Filter and control algorithm met all requirements based on simulated data. A description of the simulation effort is included in the following section

5 Software Simulation

To evaluate the performance of the Kalman filter and control algorithm, a testing framework in Python was developed to feed in simulated sensor value, simulate physical response of the system, and track sensor and control performance. This framework also allowed us to test interesting performance cases both regarding noise and regarding possible system failures. Among noise cases that we tested are: extreme random sensor noise, extreme consistent sensor noise, extreme random control noise, extreme consistent control noise, and large linear disturbances. Among system failure cases were tests to evaluate performance without a sensor or even without groups of sensors as well as cases in which individual motors weren't performing correctly and provide no thrust. In all, the system was found to be extremely resilient, except in the case that a motor is disconnected or gets stuck. In this case, the system does not react well, since there is no functionality to detect the loss of a motor and intentionally accommodate. The control input ceases to correspond to any comparable physical motion. Several sample simulation plots are included in the appendix.

6 Costs

Labor costs for the project are estimated using salaries of $40 per hour, 10 working hours per week per person, and 15 weeks for the semester.

$$2 * \frac{\$40}{hr} * \frac{10\,hr}{wk} * 15 * 2.5 = \$30,000 \tag{2}$$

Component costs are tabulated in the table below:

Part	Cost
Raspberry Pi + 8GB SD Card	$42.99
Teensy 4.0	$26.95
Assorted Wires & Connectors	$12.85
$15\,in^2$ of PCB	$45.00
2x 4000 mAh Li-Po battery + 2 spares	$123.96
Gyrosensor and Accelerometer unit	$14.94
5x ESCs	$50.64
5x Decawave	$149.40

4x USB Charger (for decawave)	$15.96
Mounting hardware (varied)	$2.99
Total	$458.64

Table 1: Parts List & Cost

All in all, development costs amounted to approximately $30,000 and each unit will cost around $459. We did spend just over $1000 on components, though, some of which ended up unused, or broken. If someone, perhaps for coordinated drone projects, wishes to build several units at once, the cost per unit could probably be drastically cut to perhaps as low as $250 or so, primarily because only one set of decawave receivers and their USB chargers would need to be purchased, and fewer batteries would probably be needed.

7 Schedule

Below is a weekly schedule with the deliverables accomplished each week.

Week	Lyle	Tae
October 7	Began research on Kalman Filter	Prototyping mainboard
October 14	Further work on Kalman Filter, assisted prototyping mainboard,	Submitted Decawave PCB, More prototyping
October 21	Finished prototyping mainboard	Debugged Decawave, set up pygame server, STM32 arduino firmware
October 28	Individual progress report, Built simulation for Kalman Filter	Designed mainboard PCB, Individual progress reports
November 4	Checked & ordered mainboard PCB	Checked & ordered mainboard, ordered power system components
November 11	Built and tested buck converter, Assembly of mainboard	Assembly & debugging of mainboard
November 18	Built new mainboard, built boost converter, built charging circuit	Built PMOS circuit

November 25	Integrated power system, debugged, redesigned & rebuilt power system, demo prep	Travelling
December 2	Final Demo, expanded simulation of Kalman filter to incorporate PID control	Final Demo, debugging server & serial, connection issues
December 9	Final presentation, final report, notebook check	Final presentation, final report, notebook check

Table 2: Schedule

8 Future Direction

There are a number of potential improvements to make to the project and several potential applications to explore. Among the improvements we hope to add are:

- Put power system on PCB
- Include optic sensors, such as the kind found on the understand of a standard computer mouse
- Add a camera and vision systems including perhaps a NVIDIA Jetson for image processing
- Add an automatic charging dock that the craft can drive up to for a recharge

Additionally, we hope to explore the following possibilities for expanding and applying the project:

- Manufacture multiple and test swarm coordination alrorithms
- Apply the electronics and control system to other types of crafts, such as a hybrid drone/hovercraft vehicle, or a classic 4 wheel holonomic drivetrain or a three wheel holonomic such as a kiwi drive.
- Assist in the testing of lunar robots by simulating low gravity and friction conditions. Currently, testing facilities that are essentially huge air hockey tables, with many air holes in the floor and massive compressors are used for this testing. We might be able to cut cost.

9 Ethics

First of all, the project itself does have a power system that poses a safety concern over the use of lithium-polymer batteries. Lithium polymer batteries are highly flammable if used incorrectly and pose a safety hazard if not handled with care [6]. In order to hold "paramount the safety, health, and welfare of the public,"[7] as the IEEE directs, making sure that this battery system is safe was important. In the design and assembly of the system, we made sure to have fire suppressing equipment easily on hand and made sure never to run the craft in spaces where such equipment was unavailable. Aside from fire safety, we made sure to take other precautions when building and testing the system. We made sure the arc of each propeller encompassed in a solid housing, and made sure that propellers were kept out of reach of wires and batteries. Finally, we added a "leash" to the craft to ensure that it couldn't drive off someone out of our reach.

Regardless of our efforts, certain safety precautions must be taken by the end user. For safe operation, the craft should only be operated in the safe temperature operating range of the battery, or between $0^\circ C$ and $50^\circ C$ [8]. Additionally, batteries should not be discharged to lower than 9 V to avoid battery damage. Key considerations like this will be detailed in any finalized product as prominent warnings in compliance with IEEE's guideline to "disclose promptly factors that might endanger the public or the environment" [7].

Though in many cases we were limited in our material selection, the bulk of the material on the craft is Polylactic Acid, which is biodegradable. Other nonbiodegradable components were made sure to be disposed of or, when applicable, recycled, to minimize the environmental development cost of the project.

10 Conclusion

In all, the project was an astounding success. Though some of the final functionality proved unattainable for the demo, we achieve most of the goals we set out to achieve. Several key areas of the project were redesigned since the design document, but we are confident that the redesigns have made the system more robust. We are happy at the results of a semester's worth of hard work, and look forward to continuing the project and exploring the potential applications of our work.

11 Appendix

11.1 Requirements and Verification

11.1.1 Electromechanical

11.1.1.1 Motors Actuators A heat transfer analysis estimating the convection and conduction resistance of heat from the motor through surrounding Nylon and to the air was performed using published values. Heat should cap at $62°C$, however a redesign of propellers to funnel air through the internals of the motor drop this equilibrium point far below the limit. Shown below are the original requirements and verifications.

Requirements	Verification
Combined thermal dissipation of over 320 W at $55°\,C$	• Run motor on 100% throttle for 5 minutes • Measure temperature of motor surface, confirming that temperature is under $55°\,C$.
All moving component's motion envelopes need to be shielded from obstacles and holding surfaces are 5 cm minimum away from motion arcs	• Ensure that no static obstacles can enter a motion envelope by horizontal translation of the craft. • Ensure that all gripping areas of the craft are at least 5 cm from any moving components at any configuration of the craft

Table 3: Motor and Actuator Requirements & Verification

11.1.1.2 ESCs

Requirements	Verification

Capable of running forward and backward at 250 RPM	• Set ESC to drive motor at lowest speed possible such that it is still spinning • Monitor 3-phase motor voltage using oscilloscope prove • Verify that the frequency is less than 4.167 Hz. • Repeat for the opposite direction.
Capable of running forward and backward at 3000 RPM	• Set ESC to drive motor at maximum throttle supported by the ESC • Monitor 3-phase motor voltage using oscilloscope prove • Verify that the frequency is greater than 50 Hz. • Repeat for the opposite direction.
Transmits signals in less than 10 ms	• Initially send a neutral command to the ESC • Monitor input signal and output signal on oscilloscope • Set oscilloscope to trigger when input signal is received • Determine the signal delay and ensure it is less than 10 ms.

Table 4: ESC Requirements & Verification

11.1.2 Chassis

Requirements	Verification

Contains all wires and electronic components to interior of box	• Attempt to feed loose cables through any holes in box • Verify that cables were unable to be pulled through any holes.
All components must be fastened and withstand 2 g's of acceleration	• Manually subject system to at least 3.46 g's of acceleration in each principle direction, which ensures that a minimum of 2 g's can be withstood in any direction • This can be accomplished by weighing the craft, selecting an appropriate counterweight, and setting up a simple balance lever to accelerate the craft upwards at a precisely measured rate • Verify acceleration targets were reached through timing and distance travelled • Verify that all components are still properly situated.
Case must withstand 10 lbs distributed compressive or tensile load in each principle axis with less than 5% total strain and no fracture of any sort	• Subject case to compressive force in each primary axis • Determine strain and verify it is less than 5% • Perform fracture and fatigue inspection • Repeat with tensile load
Resistivity of case must be greater than $\frac{1G\Omega}{m}$	• Measure resistance over 10 cm distance on case • Verify that resistance is greater than $10^8\Omega$

Table 5: Case Requirements & Verification

11.1.3 Sensors

Included below are the requirements and verification for the sensors

11.1.3.1 Magnetometer

Requirements	Verification
Magnetometer accurate to 5° while craft is stationary	• Align craft to polar north • Find error in magnetometer orientation reading and confirm that it is under 5° • Repeat until a 99% confidence interval is achieved
Magnetometer accurate to 10° while craft is in motion	• Repeatedly rotate craft back and forth between two fixed known angle setpoints • Find error in magnetometer orientation reading between extreme readings and setpoints • Confirm that each error is under 10° • Repeat until a 99% confidence interval is achieved
Magnetometer can report values every 50 ms	• Sample values every 50ms for 2 minutes • Confirm that no two read values are exactly the same unless craft is absolutely stationary

Table 6: Magnetometer Requirements & Verification

11.1.3.2 Accelerometer/Gyroscope

Requirements	Verification
Drift of less than $\frac{5°}{min}$ while stationary	• Record 1000 measurements of gyroscope data for the stationary craft • Calibrate gyroscope by applying a software shift equal to the average of the 1000 measurements • Leave craft stationary for 1 minute • Measure reported change in angle and ensure it is less than 5° • Repeat until a 99% confidence interval is achieved
Drift of less than $\frac{20°}{min}$ while experiencing angular accelerations up to $\frac{1\,rad}{s^2}$	• Record 1000 measurements of accelerometer data for the stationary craft • Calibrate accelerometer by applying a software shift equal to the average of the 1000 measurements • Apply a periodic angular acceleration control sequence to the craft for 1 minute • Measure reported change in angle and ensure it is less than 20° • Repeat until a 99% confidence interval is achieved

Drift of less than $\frac{5cm}{min}$ while stationary	• Record 1000 measurements of accelerometer data for the stationary craft • Calibrate accelerometer by applying a software shift equal to the average of the 1000 measurements • Leave craft stationary for 1 minute • Measure reported change in absolute position and ensure it is less than 5 cm • Repeat until a 99% confidence interval is achieved
Drift of less than $\frac{1m}{min}$ while experiencing angular accelerations up to $\frac{5\,m}{s^2}$	• Record 1000 measurements of gyroscope data for the stationary craft • Calibrate gyroscope by applying a software shift equal to the average of the 1000 measurements • Apply a periodic random acceleration control sequence to the craft for 1 minute • Measure reported change in angle and ensure it is less than 1 m • Repeat until a 99% confidence interval is achieved
Gyroscope and accelerometer can report values every 10 ms	• Sample values every 10 ms • Confirm that no two read values are exactly the same unless craft is absolutely stationary

Table 7: Gyroscope/Accelerometer Requirements & Verification

11.1.4 Power

11.1.4.1 Switches

Requirements	Verification
On/off switch cuts power to at most 0.5V in under 10 ms	• Connect power lines to oscilloscope • Set trigger to a threshold under normal power • Shut off power using On/off switch • Verify power coming out of the power management unit reached under 0.5V in 10ms.
On/off switch is easily accessible	• Press button • Verify that it was pressed easily and without strain or excessive force
On/off switch is at least 5 cm absolute distance from unshielded actuated mechanical components or power transmitting areas	• Measure distance to nearby hazardous components • Verify that each distance is greater than 5 cm

Table 8: Power Button Requirements & Verification

11.1.4.2 E-stop

Requirements	Verification

E-stop switch cuts power to at most 0.5V in under 10 ms	• connect power lines to oscilloscope • Set trigger to a threshold under normal power • Shut off power using E-stop • Verify power coming out of the power management unit reached under 0.5V in 10ms.
E-stop switch is easily accessible	• Press button • Verify that it was pressed easily and without strain or excessive force • Verify that it was pressed in under 1 second • Close eyes and attempt to press button again • Repeat 10 times • Verify that E-stop could be pressed without direct vision at least 9 out of 10 times
E-stop button is at least 5cm absolute distance from unshielded actuated mechanical components or power transmitting areas	• Measure distance to nearby hazardous components • Verify that each distance is greater than 5cm

Table 9: E-stop Requirements & Verification

11.1.4.3 Recharging

Requirements	Verification

System disconnected voltage threshold falls between 9.25 V and 9.5 V	• Run system until threshold is triggered • Measure battery voltage, confirm voltage falls between 9.25 V and 9.5 V
Backup battery supplies $5 \pm 0.5V$ at full charge	• Discharge backup battery to between 9 V and 9.5 V • Charge backup battery to maximum capacity • Verify on voltmeter that backup battery voltage falls into this range
Backup battery charges in under 10 minutes	• Unplug main battery • Completely discharge backup battery (powering any 5VDC device that dissipates heat appropriately will work) • Plug in main battery that is fully charged • Measure amperage into backup battery • Measure time taken until charging shuts off

Backup battery stores 0.5 Ah before dropping below 4.5 Volts	• Charge backup battery to capacity • Discharge backup battery through amp meter without exceeding 1A. • Measure amperage at 30 second time intervals until battery voltage reaches 4.5 V • Verify that the average amperage supplied times the total time amounted to at least 0.5 Ah
Backup battery continues to supply $5 \pm 0.5V$ when main battery is disconnected	• Plug in main battery for 10 minutes • Remove main battery • Verify on voltmeter that backup battery voltage still falls within $5 \pm 0.5V$

Table 10: Battery Management Requirements & Verification

11.1.4.4 DC-DC Converter

Requirements	Verification
Voltage of main electronics system is 5 ±0.25 V output	• Hook up voltmeter to output of converter • Connect a power supply • Hook up voltmeter to battery leads • Sweep voltage from 9.5 to 12.4 V • Verify that DC-DC converter output voltage remains between 4.75 and 5.25 V throughout

Main electronics system can supply 5A current	• Create a 0.8 ± 0.1Ω equivalent resistance by combining resistors • Verify equivalent resistance • Short equivalent resistance over DC-DC converter output leads • Verify that current draw falls above 5V and that voltage still falls within • Verify that DC-DC converter output voltage remains constrained to 5 ±0.25 V throughout.

Table 11: DC-DC Converter Requirements & Verification

Power_isolator.png

Figure 11: Power isolator schematics

11.1.4.5 Power Isolator

Requirements	Verification
Verify voltage spikes do not exceed 3.6 V	• Connect oscilloscope to output of isolator • Set to trigger at 3.6 V • Run hovercraft for 10 minutes, performing various motions, verify that 3.6 V was not exceeded at any point • Swap batteries, verify that 3.6 V was not exceeded.

Table 12: Power Isolator Requirements & Verification

11.1.4.6 Battery

Requirements	Verification
Battery pack leads cannot be shorted against any flat surface	• Press non-conductive surface against battery connector • Verify that neither lead is able to make contact with surface
Battery pack supplies $11.8 \pm 0.6V$ at full charge	• Discharge battery to between 9 V and 9.5 V. Use a motor or fan some device that will not overheat. • Charge battery to maximum capacity using consumer cell-balancing Li-Po battery charger • Verify on voltmeter that battery voltage falls into this range
Battery pack stores provides 6 Ah before dropping below 9.5 Volts	• Charge battery to capacity using consumer cell-balancing Li-Po battery charger • Discharge battery through an amp meter. Use a motor or fan some device that will not overheat. • Measure amperage at 30 second time intervals until battery voltage reaches 9.5 V • Verify that the average amperage supplied times the total time amounted to at least 6 Ah

Battery pack can discharge 90 A for 5 minutes continuously	• Charge battery to capacity • MEasure thickness of battery • Discharge battery at 100 A for 5 minutes • Inspect for 'burning' smell of any sort, measure thickness of battery • Make sure battery hasn't physically swollen

Table 13: Battery Requirements & Verification

11.1.5 Communication and Computation

11.1.5.1 Wi-Fi Module

Requirements	Verification
Pinging back and forth from Raspberry Pi takes under 20 ms at 15 meters range	• Place robot and router 15 meters apart with direct line of sight • Write software on RPi to send back data as soon as ping is received from user interface • Send a ping to and from RPi and measure transmission time. • Ensure the time from sending to receiving of the signal is under 20 ms.

Packet loss is under 20% at 15 m of range with line of sight	• Place robot and router 15 meters apart with direct line of sight. • Write code to count corrupted packets and correct packets • Send packets of data back and forth • Count each packet that was successfully delivered and each that wasn't • Ensure percentage of packets dropped is under 5%

Table 14: Wifi Module Requirements & Verification

11.1.5.2 microprocessor

Requirements	Verification
Combined processing and transmission time of movement command takes under 10 ms from RPi receipt to motor signal output	• Allow Craft to come to rest, dont send any control signals • Program RPI to ouput a digital signal to a free GPIO pin, hook pin to oscilloscope • Set oscilloscope trigger to 3V. • Connect another oscilloscope probe to one of the motor leads • Send motion command, measure total motion command latency on scope and verify it is under 10ms

Combined processing and relay time of state data from sensor input to microcontroller processing takes under 10 ms	• Allow craft to come to rest, dont send any control signals • Program microcontroller to output a digital signal to a pin when a the acceleration signal exceeds a $0.1\frac{m}{s^2}$ • Program RPI to ouput a digital signal to a free GPIO pin, when state registers acceleration greater than $0.1\frac{m}{s^2}$ • Connect oscilloscope to both pin, set trigger to 3V • Give the craft a sudden jolt • Measure delay time on oscilloscope, verify it is under 10 ms

Table 15: Microprocessor Requirements & Verification

11.1.5.3 Microcontroller

Requirements	Verification
Have no more than 1% corrupted packets though serial communications	• Write code to count corrupted packets and correct packets • Send packets of data until at least 1000 have been sent • Verify that fewer than 1% of packets were corrupted

Table 16: Microcontroller Requirements & Verification

11.1.6 Software

Requirements	Verification
Microprocessor runs Kalman filter state predictions in under 1 ms	• Time each prediction step in software • Repeatedly Feed Kalman filter an array of random data • Collect times of Kalman filter operations until 1000 cycles are reached • Verify that each prediction step takes under 1 ms
Each Kalman filter correction step takes under 1 ms	• Time each correction step iteration in software • Repeatedly feed Kalman filter random sensor inputs • Collect times of Kalman filter operations until 100 cycles are reached • Verify that each prediction step takes under 1 ms • Repeat for each sensor

Control system calculates output in under 5 ms	• Time each operation of the control algorithm in software • Repeatedly feed control algorithm random current state and random desired state • Collect times of control algorithm operations until 1000 cycles are reached • Verify that each prediction step takes under 5 ms

Table 17: Microcontroller Requirements & Verification

11.2 Basis Vector Thrust Decomposition

The vector decomposition can be solved with a system of equations and some linear algebra. The system of equations is included below, but the full linear algebra decomposition is not included. Note, F_1 through F_4 represent scalars from -1 to 1 of maximum linear thrust able to be supplied by any motor. Likewise x, y, and z represent scalars of the maximum thrusts able to be applied singularly in their respective directions.

$$x = \frac{F_1 + F_2 - F_3 - F_4}{4}, \quad y = \frac{F_1 - F_2 - F_3 + F_4}{4}, \quad z = \frac{F_1 + F_2 + F_3 + F_4}{4} \tag{3}$$

Since there is an extra degree of freedom in the decompoisition, the solution to the decomposition is a line in 4D space. This line is best described by the following system, derived from the original system:

$$F_1 = -F_4 + 2(y + z), \quad F_2 = F_4 + 2(x - y), \quad F_3 = -F_4 + 2(z - x) \tag{4}$$

An additional constraint on solvability is the fact that F_1 through F_4 fall between -1 and 1. This limites the geometric solution space of the system and infact constitutes a rhombic dodecahedron in cartesian representation of x, y, and z. A sample rhombic dodecahedron is shown below for reference.

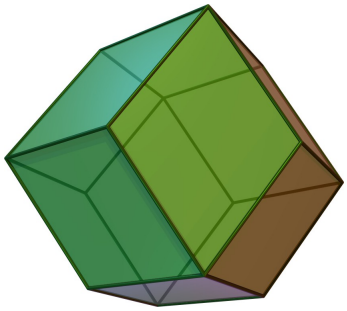


Figure 12: Rhombic Dodecahedron [9]

We seek to determine the most "scalable" solution, i.e. the vector decomposition that will be valid for the largest scalar multiple of the given basis vector. The optimization problem simplifies to the minimization of the greatest magnitude basis vector scalar coefficient. Based on the rearranged system definition in Equation 4, we easily observe that this "optimum" point occurs when two of the basis forces have equal magnitude. Partly due to the symmetry of the system, this simplifies to a very simple basis decomposition algorithm. This is extremely convenient for the purposes of computation time. We first set F_4 to zero and singly determine each other force. Then we calculate the maximum absolute difference between any two individual forces, scale F_4 by the average of these forces, and adjust F_1 through F_3 appropriately. The algorithm for the force decomposition is shown below in Figure 13 in Python.

```
def thrustAssign(x,y,z):
    c=[-2*(y+z), 2*(x-y), -2*(z-x), 0]
    max=0
    for i in range(3):
        for j in range(i+1, 4):
            if abs(c[i]-c[j])>max:
                max=abs(c[i]-c[j])
                s=-(c[i]+c[j])/2
    return -(c[0]+s),c[1]+s,-(c[2]+s),c[3]+s
```

Figure 13: Basis decomposition algorithm

11.3 Kalman Filter Mathematics

This section will detail the mathematical flow of the Kalman filter and explain the tradeoffs decided upon to improve the runtime of the filtering steps. I will explain each of the Kalman filtering steps and document each of the key matrices in the algorithm and their purposes. The first step in the filter is the prediciton step. In this step, the state of the next time step is predicted based on current time step and current system control input. Here, n is the time step, X is the State Matrix, A is the Dynamics Matrix, B is the Control Matrix, and U is the Control Input:

$$X_{n+1} = AX_n + Bu \tag{5}$$

For a ground based robotics system, certain system parameters such as vertical position, tilt, and corresponding velocities and accelerations aren't necessary parameters to keep track of in the system state since we typically assume these to be static. Additionally, acceleration can either be included in the system state or excluded in the system state. If acceleration is excluded, accelerometer data is used directly in the system prediction step. Additionally, the acceleration and system control input would need to be fused in some capacity prior to integration in the system, which would result in additional computation time. Since one of the key benefits of omission of acceleration state variables is a reduction in system size and subsequent ease of computing, this defeats some of the purpose of the smaller system. Because the accelerometer will ideally run at a faster frequency than the system prediction step, including accelerations in the system states was deemed to be preferable. As such, the system state matrix X is expressed below in Equation 6. Worth noting is that z, $\dot{z}$ and $\ddot{z}$ refer to angle, angular velocity, and angular acceleration, respectively. x and y are standard Cartesian coordinates.

$$X^T = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \ddot{x} & \ddot{y} & \ddot{z} \end{bmatrix} \tag{6}$$

Next, I introduce the Dynamics Matrix, A, which controls how the prediction of system state given no additional input is performed. dt here is the time elapsed since the past state prediction step. This matrix is designed to update the positions and velocities based on the velocities and accelerations respectively multiplied by the time elapsed. Note that the last three rows lack the 1s on the diagonal like the velocity and the position scaling terms because, while inertia applies to position and velocity, it does not apply for acceleration and without a force input u, a nonzero acceleration is not expected.

$$A = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

The input matrix u contains the various forces that modify the state in some way, and the matrix B governs which system states these forces affect. For this system, the forces will correspond to the expected linear accelerations and angular acceleration caused by the current signal to the motors. B is determined from simple Newtonian kinematics and some basic calculus. An example derivation for one of the position factors is shown in Equation 10, where a_{fx} is the expected applied acceleration and a_x is the previous time step's acceleration. Recall that the previous state's acceleration terms are zeroed out in the A matrix, justifying the 1:1 scaling of acceleration due to system control input. B and u are shown below:

$$B = \begin{bmatrix} \frac{dt^2}{6} & 0 & 0 \\ 0 & \frac{dt^2}{6} & 0 \\ 0 & 0 & \frac{dt^2}{6} \\ \frac{dt}{2} & 0 & 0 \\ 0 & \frac{dt}{2} & 0 \\ 0 & 0 & \frac{dt}{2} \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$u = \begin{bmatrix} a_x \\ a_y \\ \alpha_z \end{bmatrix} \tag{9}$$

$$\frac{\int_0^{\delta t} \int_0^{\delta t} (a_x + a_{fx} t) dt dt - v * \delta t}{\delta_t} = \frac{a_{fx} * \delta t^2}{6} \tag{10}$$

The next step of the Kalman filtering algorithm is the update step of the covariance matrix, P, which depends on P, A and the Noise Covariance Matrix, Q. In designing Q, I assumed standard linear acceleration noise. In this case, Q is expressed through a an operation using a vector, G derived similarly to B, where the components of G scale the standard deviations of acceleration according to their respective effects on the system, just as in B:

$$P = APA^T + Q \tag{11}$$

$$Q = GG^T * \sigma_a^2 \tag{12}$$

$$Q = GG^T * \sigma_a^2 \tag{13}$$

$$G^T = \begin{bmatrix} \frac{\delta t^3}{6} & \frac{\delta t^3}{6} & \frac{\delta t^3}{6} & \frac{\delta t^2}{2} & \frac{\delta t^2}{2} & \frac{\delta t^2}{2} & \delta t & \delta t & \delta t \end{bmatrix} \tag{14}$$

The next step in the filter is the evaluation of y, the innovation factor. The components of Z constitute the 6 pieces of useful sensor input. The components of H scale the expected system state and arrange them to compare against sensor values. As such, these values are selected to match sensor output with their measured system parameters. Additionally, the constants chosen need to scale the state appropriately to match the measurement scale from the sensors.

$$y = Z - (HX) \tag{15}$$

$$Z^T = \begin{bmatrix} Deca_x & Deca_y & Mag_\theta & Accel_x & Accel_y & Gyro \end{bmatrix} \tag{16}$$

$$H = \begin{bmatrix} c_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_6 & 0 & 0 & 0 \end{bmatrix} \tag{17}$$

Next, an intermediate term, S, based on matrices already defined as well as one new matrix, the measuring noise conversion matrix, R, which defines the variance of each sensor input along the main diagonal and the covariances of each pair of sensor inputs at every $R_{i,j}$. Though this will be confirmed through testing, I currently suspect that all covariances will be 0, implying that each sensor input is independent of every other A demonstration of method to determine R will be presented later in the report.

$$S = HPH^T + R \tag{18}$$

$$R \begin{bmatrix} R_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & R_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & R_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & R_6 \end{bmatrix} \tag{19}$$

Next, the Kalman Gain, K is computed. This is the step in the algorithm that requires the inverse operation. Following the computation of the Kalman gain, the predicted state is corrected using the gain and the sensor input. Additionally, the covariance matrix is updated. This concludes one cycle of the Kalman filter, featuring one state prediction and one state correction.

$$K = PHT^T S^{-1} \tag{20}$$

$$X_{new} = X_{old} + Ky \tag{21}$$

$$P_{new} = P_{old} - KHP_{old} \tag{22}$$

In order to avoid calculating the inverse every sensor correction cycle, We examine what exactly factors into the Kalman gain. K depends on P, H and S, which subsequently depends on R. P depends on A and Q. We observe that none of these matrices depend at any point on the input, u, the sensor measurements, Z, or the current state, X. As such, for systems with static expected error, K, P, H, S, R, P and A actually constitute their own system independent of system state or input. Consequently, the convergence state of this subsystem can in fact be precomputed independent of any real time data or measured parameters. This yields static Kalman gain, K, which can be hard coded into the algorithm. Simply, put the tradeoff for this computation efficiency is the inability for the craft to change its behavior depending on its situational surroundings.

11.4 Sample Simulation Results

Shown below is a sample plot of x, y and angle when the craft doesnt have access to IMU data, starts off position, and is subject to both measurement and control noise throughout its path.

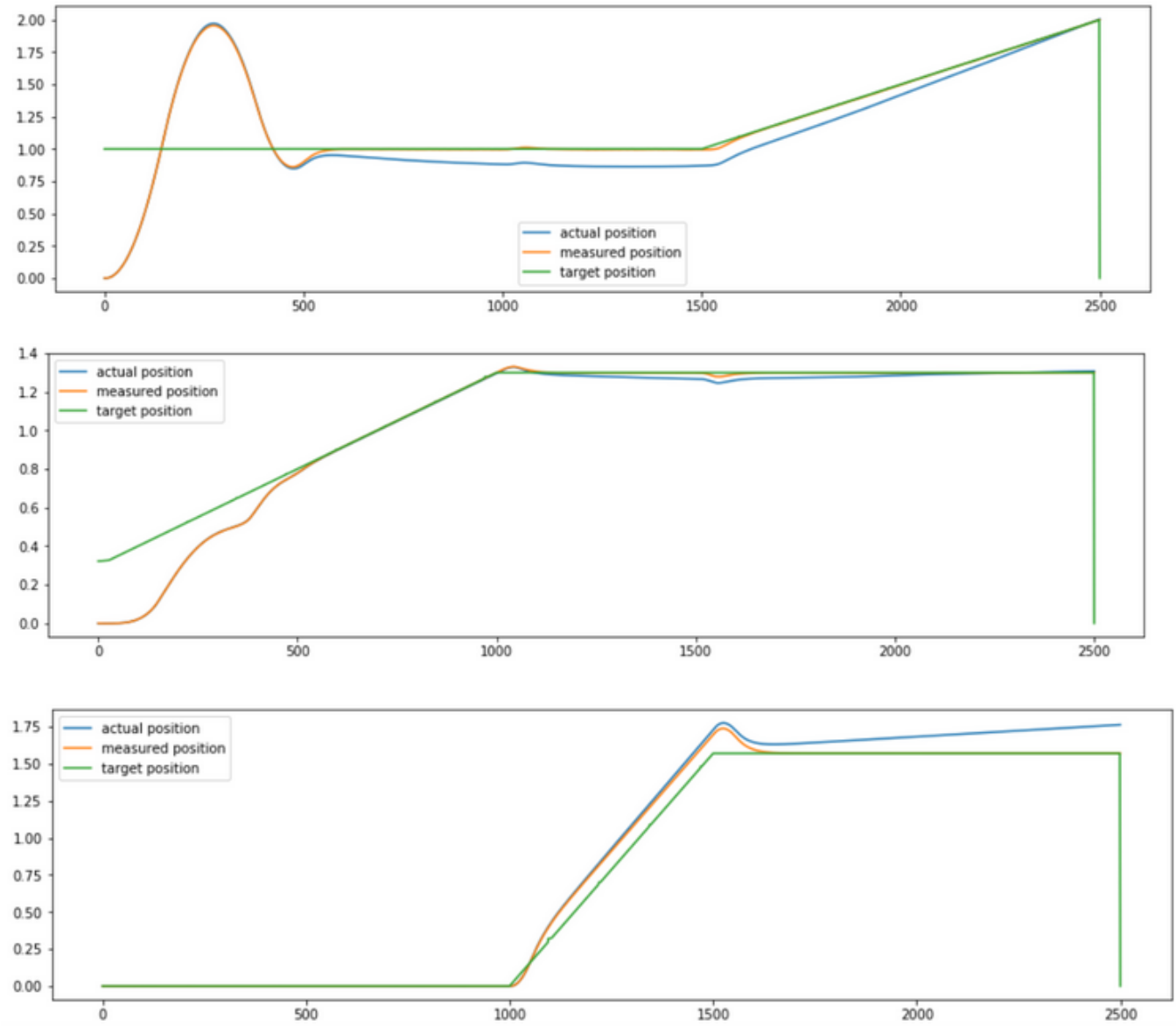


Figure 14: Sample Simulation Plots

12 References

References

[1] V. Vladimerou, A. Stubbs, J. Rubel, A. Fulford, J. Strick, and G. Dullerud, "A hovercraft testbed for decentralized and cooperative control," in *Proceedings of the 2004 American Control Conference*, vol. 6. IEEE, 2004, pp. 5332–5337.

[2] "Resources." [Online]. Available: http://www.simbotics.org/resources/mobility/drivetrain-selection

[3] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, vol. 3068. International Society for Optics and Photonics, 1997, pp. 182–193.

[4] K. Townsend, "Nxp precision 9dof breakout," Apr 2017. [Online]. Available: https://learn.adafruit.com/nxp-precision-9dof-breakout/overview

[5] S. Srini, "The kalman filter: An algorithm for making sense of fused sensor insight," Apr 2018. [Online]. Available: https://towardsdatascience.com/kalman-filter-an-algorithm-for-making-sense-from-the-insights-of-various-sensors-fused-together-ddf67597f35e

[6] R. H. Reif, M. Liffers, N. Forrester, K. Peal *et al.*, "Lithium battery safety: A look at woods hole oceanographic institution's program," *Professional Safety*, vol. 55, no. 02, pp. 32–37, 2010.

[7] "Ieee code of ethics," 2016. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[8] "Safety rules for lipo batteries - onyxstar - leading edge drones." [Online]. Available: https://www.onyxstar.net/safety-rules-for-lipos/

[9] "Rhombic dodecahedron," Oct 2019. [Online]. Available: https://en.wikipedia.org/wiki/Rhombic$_{d}odecahedron/media/File$: $Rhombicdodecahedron.jpg$