Design and Testing of an Amphibious AUV

A Major Qualifying Project Report
Submitted to the Faculty of the
WORCESTER POLYTECHNIC INSTITUTE
in Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science
in Aerospace Engineering and Robotics Engineering

by

Michael Beskid

Ryan Brunelle

Calista Carrignan

Robert Devlin

Toshak Patel

Kofi Sarfo

MARCH 3, 2023

Approved by:

Michael A. Demetriou, Advisor
Professor, Aerospace Engineering Department
WPI

Abstract

The objective of this project is to design, construct, and test an autonomous quadrotor vehicle capable of combined at-will aerial flight and underwater locomotion. The system consists of a quadrotor structure with a waterproof enclosure containing the necessary electronics and microcontrollers for autonomous flight. A ballast system is utilized for air to water transitions and was designed to ensure the craft will sink and rotate to the correct position for underwater propulsion when filled and back to an upright position when emptied to allow for takeoff. All of this is designed to be completed autonomously with the designed flight controller which utilizes a Teensy 4.0 microcontroller and Raspberry Pi within a complete sensor array. To demonstrate these capabilities, the designed quadrotor is to complete a mission within an indoor swimming pool area. The craft will start in a hover above the surface at one end of the pool, land on the surface and dive underwater to traverse the length of the pool, then resurface at the opposite side to take off and return to base. Within this paper, the design process and iterations are presented along with the corresponding research used to improve each iteration.

Acknowledgements

All authors would like to thank Professor Michael Demetriou, Professor David Olinger, and Professor John Sullivan for their support and guidance during the duration of this project. We would also like to thank Nicholas Rehm for publishing his dRehmFlight flight computer publicly, which was invaluable to development of the flight software for this project.

Table of Authorship

Introduction	
Overview	Ryan Brunelle
Project Objectives	Kofi Sarfo
Requirements, Constrains, and Considerations Design	Robert Devlin
Project Management and Team Organization	Toshak Patel
Societal Impacts	Kofi Sarfo
Background	
Brief History	Calista Carrignan
Past MQP Work at WPI	Michael Beskid
Related Research on Amphibious AUVs	Ryan Brunelle
Quadrotor Flight Mechanics	Michael Beskid
Aerial Dynamics	Michael Beskid
Underwater Dynamics	Michael Beskid
Autonomy and Localization	Calista Carrignan

Technical Sections	
Quadrotor Flight Controllers	
Commercially Available Flight Controllers	Calista Carrignan
Custom Flight Controllers	Calista Carrignan
dRehmFlight VTOL	Calista Carrignan
Electronics	
Power Subsystem	Michael Beskid
Propulsion Subsystem	Michael Beskid
Sensing and Localization Subsystem	Michael Beskid
Ballast System Electronics	Michael Beskid
State Estimation	
Onboard Sensors	Calista Carrignan
Madgwick Filter	Calista Carrignan
Control Architecture	
Control System Overview	Michael Beskid
Electronic Speed Controllers	Michael Beskid
PID Control	Michael Beskid
Stabilization and Attitude Control	Michael Beskid
Altitude and Position Controllers	Michael Beskid
Flight Control Software	
Overview and Development Environment	Michael Beskid
Flight Computer Architecture	Michael Beskid
State Machine	Michael Beskid
Test Vehicle Construction	Michael Beskid
Test Quadrotor Vehicle	
Preliminary Calibration	Michael Beskid
Flight Testing	Michael Beskid
Quadrotor Body Design	
Quadrotor Body Design	Robert Devlin
Material Selection	Kofi Sarfo / Robert Devlin
Enclosure	Ryan Brunelle / Robert Devlin
Enclosure Version 1	Robert Devlin
Hydrostatic Analysis	
Force of Buoyancy and Gravity	Toshak Patel
State Space Modeling	Toshak Patel
Ballast System	
Ballast System	Kofi Sarfo / Robert Devlin
Ballast Test 1	Robert Devlin
Ballast Test 2	Robert Devlin
Ballast Version 3	Robert Devlin

Results	
Summary	Michael Beskid
Conclusions	Calista Carrignan
Recommendations for Future Work	Calista Carrignan
Broader Impact	Ryan Brunelle

Contents

Table of Figures

1 Introduction

1.1 Overview

Advancements in underwater and aerial travel show merit in both civilian and military applications. The ability to seamlessly transition between underwater and aerial movement opens the door for possibilities including reconnaissance, data collection, et cetera. Water surface vehicles such as sea planes are one of the most common vehicles that approach the idea of an amphibious aerial vehicle but, as their name implies, they are limited to the water's surface. An AUV (Autonomous Underwater Vehicle) Quadrotor is an excellent candidate for an adequate bridge between the two mediums as many restrictions placed on traditional fixed wing aircraft are lifted in the case of a quadrotor. As such, the need for further research regarding AUV Quadrotors is clear.

1.2 Project Objectives

The given mission for the amphibious quadrotor will be to initially take off from one side of a swimming pool (side A), translate over to hover above the water's surface in stage 1, then slowly lower to the surface of the pool at stage 2. The quadrotor will then perform its submerging sequence and rotate forward by 90 degrees, such that its vertical flying position will be horizontal. The vehicle is to then travel to stage 4 of the pool and rotate 90 degrees in the opposite direction, resurface and fly out of the pool (stage 5). The rotor is to loiter in stage 6 and lower back down to the surface of the water at stage 7, which will be a different part of the pool. The rotor should repeat the steps at stage 1 and 2, then maneuver its way back to its initial position (stage 3). Fig. 1.1 below illustrates the mission.

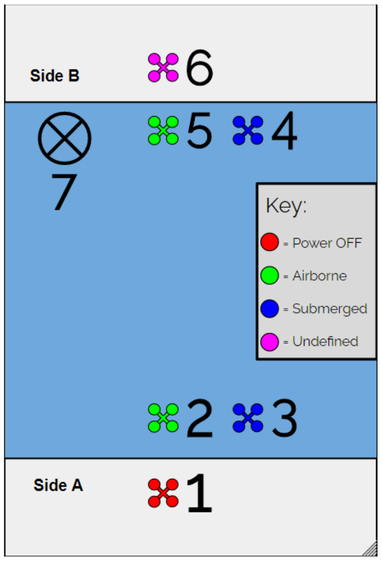


Fig. 1.1. Amphibious Quadrotor Mission Diagram.

To achieve the mission statement above, the quadrotor will have to meet certain criteria. Therefore, there are a few goals the team must reach to make this project a success. The main goals of this Amphibious AUV MQP begin with literature search. The search shows the team what has already been done and how it was accomplished. After this, the team will design and test an AUV, that is autonomous and is capable of transitioning from air to water. To ensure that the vehicle is capable of both aerial flight and maneuvering underwater, the team will perform aerodynamic and hydrodynamic analysis. Included in the aerodynamic analysis, the team will select and analyze an appropriate propulsion system for the aircraft to meet aerodynamic specs. After this, the team will perform stability analysis to ensure the aircraft is statically and dynamically stable. We will then design and optimize control algorithms to achieve stability in both mediums.

Stability is very important for autonomous operation of the aircraft. To avoid structural failure, the team will conduct structural analysis to ensure the quadrotor does not disassemble in flight or in water. Structural analysis will include waterproofing to prevent water from leaking into the quadrotor and destroying onboard electronics. A ballast system will be incorporated to achieve active buoyancy control and to aid in rotating the vehicle when submerged in water. When analysis is complete, the team will begin the construction of the amphibious AUV for flight and underwater testing. After testing, the team will construct a final working Amphibious AUV for the project showcase and demonstrate a successful mission at the end of the project.

1.3 Design Requirements, Constraints, and Considerations

For this MQP the group is to develop and manufacture an AUV Quadrotor that meets the following requirements:

- The vehicle shall be able to perform stable aerial flight.
- The vehicle shall autonomously navigate a pre-determined route in the air.
- The vehicle shall successfully transition between air and water as well as from water to air.
- The vehicle shall travel under the surface of the water while remaining stable.
- The vehicle shall autonomously navigate a pre-determined route underwater.

The primary constraints of this MQP are time and budget. The project should be completed within the three allocated terms from August 2022 to March 2023. The available budget is $250 dollars per person, totaling $1500 dollars. These budget and time constraints determine the scale and scope of the project, including size, potential sensors, and what components can be commercially sourced. Other constraints and considerations include:

- Adherence to FAA drone regulations.
- Adherence to WPI’s rules and policies.
- Understanding of potential safety risks. As the most critical components are commercial products, the risks of this project are like a standard quadrotor: highspeed propellers, Lithium-Ion batteries, low voltage electronics.

- Understanding environmental risks. Since the quadrotor will be operating within a standard chlorinated pool, caution must be given to ensure no external materials react poorly with the water. Other environmental risks are negligible and shared with standard quadrotors.

1.4 Project Management and Team Organization

The project was established to have six students and two faculty advisors from the Aerospace Engineering and Robotics Engineering departments. The group was given autonomy in its internal organization, allowing the students the opportunity to practice a degree of self-governance and interpersonal skills. The project team, comprised of Michael Beskid, Ryan Brunelle, Calista Carrignan, Robert Devlin, Toshak Patel, and Kofi Sarfo, was divided into two teams in order to best align each member's strengths with the needs of the project. Both teams operate under a central Chief Engineer, Calista, who oversees issue tracking, scheduling, and ensures both teams remain on schedule for all deadlines and successful completion of the project.

The Structures team – comprised of Ryan, Robert, Toshak, and Kofi – was tasked with the physical design and fabrication of the Autonomous AUV. This group was allocated four members to support designing the vehicle quickly such that it could be fabricated in time for the second team to advance their work.

The Navigation and Controls team – comprised of Michael and Calista – was tasked with the development of the equations of motion for the vehicle, selecting and implementing onboard electronics and sensing solutions, developing control algorithms, and writing the flight control software. The group decided that this team would have less resources at the beginning of the project in order to prioritize the physical fabrication effort. The entire project group would reevaluate resources in B term to ensure that all tasks were being performed at equal pacing.

The entire team has dedicated time to meet with the Aerospace Department advisor to present weekly progress on the project, and the written report, as well as to troubleshoot any issues that may have arisen during the development of the project. During these meetings, a presentation is given by 3-4 of the group members, in rotation with the other remaining members to ensure that everyone is given the opportunity to present the work being accomplished. After these presentations, the advisor would provide feedback, and further guide the group towards the objective. Outside of these times, each of the two teams have scheduled time to meet with their groups individually in order to continue to develop their respective portions of the project. The entire group would then convene weekly as a whole to provide updates and discuss larger design decisions for the project to ensure a seamless completion of project goals.

1.5 Societal Impacts

UAVs are a special type of vehicle that can run on electricity alone, this means that they produce no pollution through fuel consumption and emit no harmful particles or compounds into the atmosphere. Although these vehicles can be powered by liquid fuel and other forms of energy, this team will use a battery to power our amphibious UAV to limit pollution. Therefore, when this project is a success, they can be utilized in the real world and have minimal effect on global pollution. Due to our design constraints, this UAV will be able to observe and record data from dangerous places, especially underwater. A significant way this UAV can impact society is by going underwater to depths that may potentially be harmful to humans to search for things such as: sunken ships to observe marine life land, oil spills and many more. This vehicle can potentially save many lives by reducing the risk of sending people into unknown waters. Since the vehicle is autonomous it can easily operate in GPS-denied environments; for military purposes it could be used for intrusion detection in hazardous and threat spatial fields or to contain and eliminate adversaries. These are just a few of the ways in which the amphibious UAV could be utilized. Due to its capabilities, the amphibious UAV could be used in applications far beyond what is currently envisioned by the team.

2 Background

2.1 Brief History

The quadrotor was invented in 1907 in France by Acques and Louis Breguet, along with Professor Charles Richet [1]. While this model was virtually non-maneuverable and required multiple pilots, it was the start of the evolution of the quadrotor we know today. Over the next 70 years following this achievement, there would be virtually no progress with this technology until the growth of micro electromechanical systems in the 1990's, with which came the design of the microcontroller. This progress in microelectronics would allow for the technology to become more accessible, allowing for a larger application with a broader range of uses for the technology from research to hobbyists to military applications [2].

Alongside the fast-growing popularity of the quadrotor, the idea of modern UAVs and unmanned flight system was introduced in 1980, with most progress with this technology occurring over the last 20 years [3]. Each is defined as an unmanned vehicle with a UAV for aerial operation and AUV for underwater locomotion. UAVs and AUVs were designed to promote human safety, both pilot and pedestrian, as well as to allow for the creation of complex vehicles that are too small to carry a human pilot [4]. For the purpose of this project, the quadrotor was the best option of the various rotorcraft options to choose from, due to its high level of maneuverability, both above and below water, as well as its capacity for autonomous operation, as stated by Ghazbi [4], within their paper "Quadrotors Unmanned Aerial Vehicles: A Review".

2.2 Literature Review

2.2.1 Past MQP work at WPI

WPI has a significant history with regards to research into autonomous aerial vehicles, specifically quadrotors. Student MQP teams have carried out research in the design, control, and novel use of these vehicles for over a decade and have been reviewed in depth. The new research performed by our team in the development of an amphibious quadrotor vehicle builds upon the work of past students and the expertise of the faculty associated with the affiliated departments of Aerospace Engineering and Robotics Engineering.

The earliest quad-rotor project at WPI, titled *Design Optimization of a Quad-Rotor Capable of Autonomous Flight,* was completed in 2008 with the goal of achieving stable point to point autonomous flight [5]. The vehicle was constructed from an existing quadcopter design and incorporated some design modifications and custom electronics. The system used an MSP430 microcontroller, 5-DOF inertial sensor, and sonar sensor and made use of custom control algorithms developed by the team. Despite making progress on improving the vehicle's thrust capabilities and validating control methods in simulation, the team was unsuccessful in achieving the goal of an autonomously hovering craft. A new student team took over the project in 2009 and aimed to continue this work [6]. The team opted for a similar approach and performed several iterations of flight tests in attempts to achieve a stable hover. This team was able to achieve a significant weight reduction with a redesigned frame and overhauled the control scheme. These efforts, however, failed to result in a successful documented autonomous flight.

A 2011 project entitled *Design of an Autonomous Quadrotor for Urban Search and Rescue* had the objective of designing and testing a quadrotor capable of stable autonomous flight with additional computer vision capabilities [7]. This team, comprised of Electrical and Robotics Engineering students, aimed to produce a quadrotor which could autonomously navigate indoor environments and transmit data back to an operator for search and rescue purposes. The system utilized a low-level flight controller called the HoverflyPro, which was augmented with an Overo processor. The vehicle was outfitted with a suite of sensors including ultrasonic and IR rangefinders for altitude control, as well as an IMU for stabilization and LiDAR for eventual higher-level mapping and path planning. This team was successful in achieving stable, piloted flight and produced impressive flight time and payload capacity numbers compared to commercially available quadcopters of the time. While largely successful, the project fell short of the goal of achieving autonomous takeoff/landing and traversal through indoor spaces.

Vision-Based Obstacle Avoidance for Small UAVs was a 2015 project from the Aerospace Engineering department which leveraged computer vision on a quadrotor platform [8]. This team chose the commercially available IRIS quadcopter produced by 3DRobotics as a starting point for the system. The IRIS platform comes equipped with a Pixhawk flight controller which includes an integrated autopilot firmware. The project benefited from advances in quadcopter technology over the last few years, allowing the team to use the commercial flight controller and autopilot to achieve stable flight. This enabled more focused work to be done on the computer vision system. The team designed a custom mount to accommodate two Microsoft Lifecam HD 3000 cameras and a Raspberry Pi 2, which would act as a companion computer to run a Python implementation of OpenCV and communicate with the primary flight controller. Bench testing of this system was conducted of a stereo depth mapping algorithm which was found to have high precision but limited accuracy. The team performed a manually piloted test flight in an outdoor environment to validate the effectiveness of the obstacle avoidance system. The team was unable to collect good camera data due to overexposure and difficulty tuning camera parameters in addition to slow processing from the computationally intense algorithm. The conclusions indicated that achieving autonomy could be possible in future work by implementing a control loop.

The 2016 project *Autonomous Quadrotor Navigation and Guidance* carried on this work with the use of similar hardware and computer vision techniques [9]. This team utilized the same IRIS quadcopter platform with integrated Pixhawk flight controller, and further made use of OpenCV with a Raspberry Pi 2 companion computer to perform computer vision. The goal of this project was to develop an autonomous guidance system for a UAV with the use of a downward facing camera to track identifiable landmarks. This team was the first to demonstrate autonomous flight capability by making use of the Pixhawk's autopilot firmware and the associated ground station software called Mission Planner. A Raspberry Pi Cam was used for this project. The camera was used to implement a shape detection algorithm to determine the location of ground-based markers for path planning purposes. The team noted that the image processing and target detection algorithms were successful in bench testing but could benefit from further tuning. While communication between the Raspberry Pi 2 and Pixhawk flight controller was achieved, the team was unsuccessful in getting the quadcopter to fly with the computer vision system.

In 2020, WPI’s Aerospace Engineering Department hosted a UAV Competition in which multiple MQP teams competed. Two MQP reports produced from this competition include *Quad-plane Design for Autonomous Cargo Delivery* and *A High-Performance Aircraft for the 2020 WPI UAV Competition* [10], [11]. These projects were required to use electric propulsion systems and integrate vertical takeoff and landing capabilities. The competition involved the development of a VTOL micro-aerial vehicle that could carry a payload and release it to hit a specified target. Both teams once again made use of the Pixhawk flight controller, ArduPilot firmware, and Mission Planner ground station software to control their vehicles. The first team upgraded to the NVIDIA Jetson Nano for a more powerful companion computer than a Raspberry Pi and included a Raspberry Pi Cam and Garmin LiDAR Vision L3 Lite for target identification and path planning. These projects are especially relevant to our work due to their multiple flight modes including vertical takeoff/landing and standard airplane flight. The designs both showed some promise in initial flight testing but suffered some challenges and did not fully demonstrate the goals of the competition. The sudden outbreak of the Covid-19 pandemic prevented the continuation of hands-on testing, and the competition was never held.

The most recent quadrotor MQP project at WPI was 2022’s *Low-Cost Quadrotor Micro-Aerial Vehicle* project [12]. The goal of the project was to develop a micro-aerial vehicle (MAV) which would carry a multi-sensor payload and navigate through an indoor environment with a Vicon Motion Capture localization system in place of GPS positioning. The team developed a traditional X-frame quadcopter with a Pixhawk 4 mini flight controller for stabilization and attitude control. A HardKernel ODROID XU4 companion computer was added for high-level control and processing. This project uncovered several interesting findings and recommendations for quadrotor construction and stability given its unique size and power constraints. While testing of the final MAV was unsuccessful in demonstrating stable flight, extensive research into motion capture technology and its interface with commercial flight computers provided valuable data that will be of use to future UAV research at WPI.

2.2.2 Related Research on Amphibious AUVs

Research into other amphibious AUVs was conducted by the team to gain a greater understanding of the current research into AUV Quadrotors. The primary source of research was the LoonCopter, an amphibious quadrotor developed by Hamzeh Alzubi, Iyad Mansour, and Osamah Rawashdeh from Oakland University in 2014. The LoonCopter is a remote-controlled quadrotor that operates as a typical quadrotor when airborne [13]. Their RC (radio controlled) method is the primary difference between the scope of this project and the work covered in their research, as this project's intent is to create a fully autonomous quadrotor that relies solely on sensor information instead of user input. In terms of mechanism, the LoonCopter is outfitted with a ballast system which is used to control the buoyancy of the vehicle.

The most interesting design choice the LoonCopter team made lies in how they control their center of buoyancy with the ballast system. While operating, the center of buoyancy and mass is displaced in such a way that the vehicle is stable underwater when rotated 90 degrees from its original orientation. This allows for the control of the vehicle to be simplified as the righting force of buoyancy now keeps the craft in an optimal position to travel while underwater. The research paper associated with the LoonCopter included diagrams describing the forces experienced by the vehicle whilst underwater as well as the shifting of the center of mass and buoyance. These concepts inspired this project to utilize similar methods of buoyancy control found in the LoonCopter. The LoonCopter is shown in Fig. 2.1.

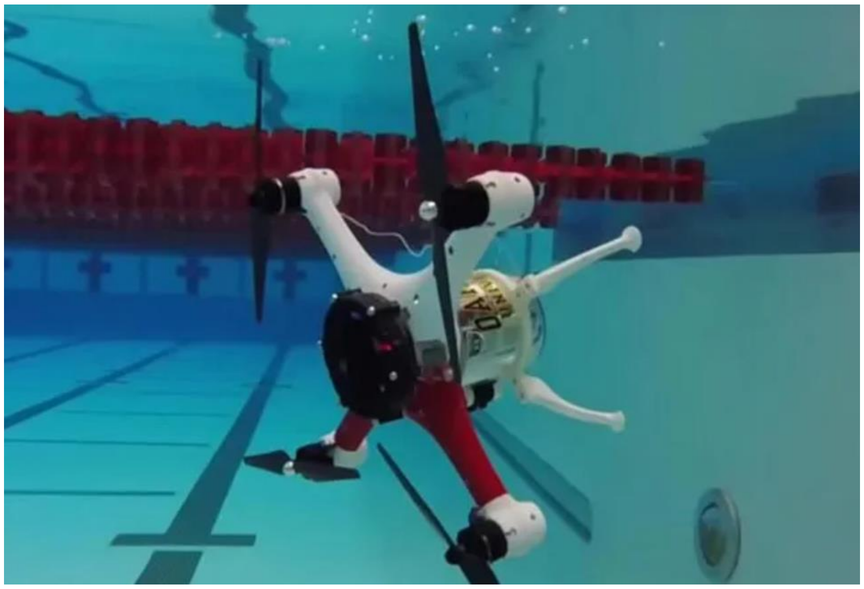


Fig. 2.1. LoonCopter from Oakland University. [13]

Other sources of research regarding Amphibious AUVs extend to a craft developed by Marco Moreno Maia, Parth Soni, and F. Javier Diez-Garias from Rutgers University. Their paper, titled *Demonstration of an Aerial and Submersible Vehicle Capable of Flight and Underwater Navigation with Seamless Air-Water Transition* provides great insight into the challenges faced when trying to design a craft that can transition between air and water and approaches the problem with a little more inspiration from natural sources ￼￼￼[14]￼￼animals of nature conquer their respective mediums of travel. Flight, for a bird, is a combination of lift and thrust counteracting weight while, for a fish, swimming buoyancy. They make a comparison for aquatic birds as well as flying fish and exemplify how the forces at place for an amphibious creature need to be balanced for the combination of the two modes of travel to be properly integrated.

To tackle the problem of air to water transition, Rutger's team utilized a dual propeller system at the end of each of the vehicle's arms separated by a significant distance, which allows for the lower propellers to become submerged while the upper propellers maintain the primary upwards thrust of the vehicle. Their vehicle, dubbed "The Naviator," is depicted in Fig. 2.2.

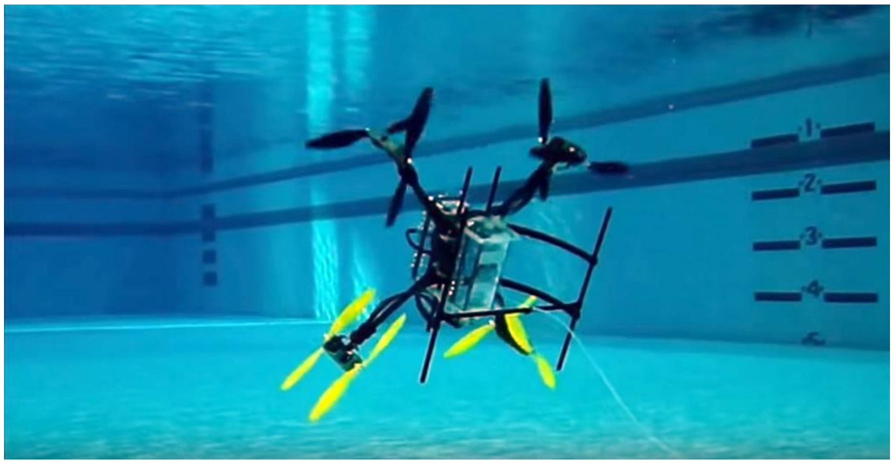


Fig. 2.2. The Naviator from Rutgers University. [14]

This allows for an extremely smooth transition between air and water. They also address the issues of using standard propellers meant for air whilst in water. The quadrotor is positively buoyant and forces itself underwater with the downward facing propellers in the same way that the upward facing propellers lift the craft above the ground. While allowing for each set of propellers to be better optimized for the two unique environments, this method is much more power consuming than the LoonCopter's active ballast system. They concluded that the effects encountered within water do not require any special propeller modifications, which reflects the findings of the LoonCopter team on the same issue. The LoonCopter team goes into more depth about vapor pockets forming underwater at high propeller speeds but found that the propeller speeds needed for underwater travel did not reach the threshold of encountering that issue. Overall, Rutger's team's research provided some inspiration and information regarding the necessary balance between the two modes of transportation.

2.3 Quadrotor Flight Mechanics

By the simplest definition, a quadrotor is a helicopter that has four rotors. Quadrotors are a subset of multicopters, which can be defined more generally as any aerial vehicle which uses more than one rotor to achieve lift. Multicopters are most often small, unmanned vehicles and typically feature three, four, six, or eight rotors. The quadrotor is the most common and may be constructed in several different configurations. Fundamentally, the frame of a quadcopter consists of a central hub, which houses onboard computers and electronics, and four arms which extend outward from the center. The propellers are installed onto motors mounted at the ends of the arms. Three common frame configurations include the X frame, plus-frame, and H frame, each named to describe the structure's geometry. A depiction of these different frame types is shown in Fig. 2.3. Some other frame types exist, including those with stretched or asymmetric configurations, however these are less common due to reduced stability and difficulty of control. The amphibious quadrotor developed by our team features an X frame, which will be the basis for the dynamics and controls discussed in this report.

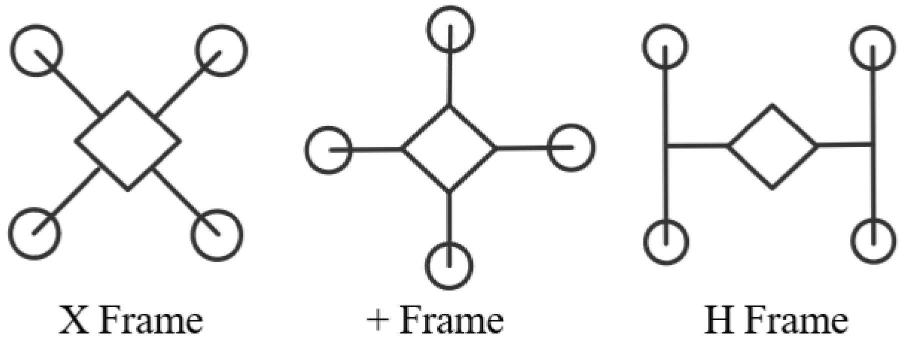


Fig. 2.3. Common Quadrotor Frame Configurations.

The coordinate frame used to describe a quadrotor's motion is shown in Fig. 2.4. Note that this is a body-fixed frame which translates and rotates with the vehicle with its origin at the center of gravity. Translation motion in the X direction is defined as longitudinal movement. Translation motion in the Y direction is defined as lateral movement. Finally, translational motion in the Z direction is defined as vertical movement. Rotations about the X, Y, and Z axes are defined as roll, pitch, and yaw respectively. These definitions are illustrated below.

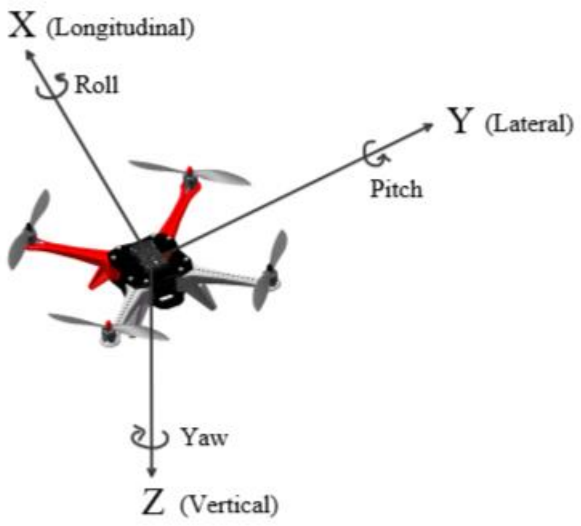


Fig. 2.4. Quadrotor Coordinate Frame Definitions.

A quadrotor achieves different motions by varying the speed of its four propellers. The vehicle features two propellers which spin in the clockwise direction and two which spin in the counterclockwise direction. Pairs of propellers which spin in the same direction are located opposite one another on the frame. This arrangement allows for the torques produced about the vehicle's center of gravity by one pair of motors to be cancelled out by the torques produced by the other pair of motors. The quadrotor is non-holonomic given that it has six degrees of freedom and only four actuators. Due to this constraint, the vehicle is capable of changing its position and attitude but can only maintain its position when level with the ground.

Vertical movement is achieved by increasing or decreasing the speed of all four motors to affect the total thrust produced. The vehicle achieves a hover when its thrust exactly equals its weight. Lateral movements are achieved by varying the angular speeds of the four propellers independently to induce angular rotations. This changes the attitude of the vehicle, misaligning the thrust vector with the downward gravitational force, causing a lateral motion. The following figures explain this principle in more detail by illustrating how different motions are achieved. The blue arrows represent clockwise spinning propellers, while the green arrows represent counterclockwise spinning propellers. The arrows are also weighted to show relative angular speeds. Thicker arrows represent higher propeller speeds while thinner arrows represent lower propeller speeds.

As discussed above, vertical motion is achieved by increasing the speed of all propellers to move upwards, or conversely decreasing the speed of all motors to move downwards. This is shown in Fig. 2.5.

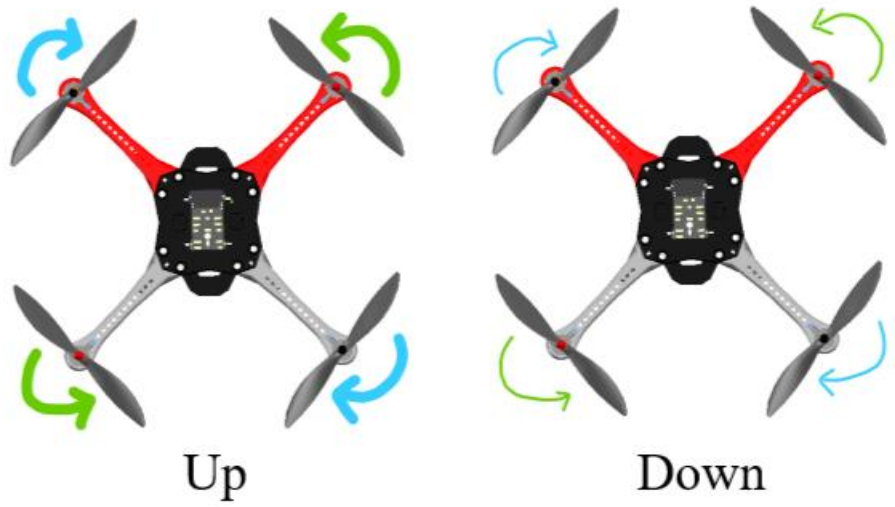


Fig. 2.5. Motor Speeds for Vertical Movements.

Turning the quadrotor to change its heading is achieved by increasing the speed of the propellers spinning in one direction while decreasing the speed of the propellers spinning in the other direction. This acts to unbalance the torques produced by each pair of propellers, causing the vehicle to yaw. This is demonstrated in Fig. 2.6.

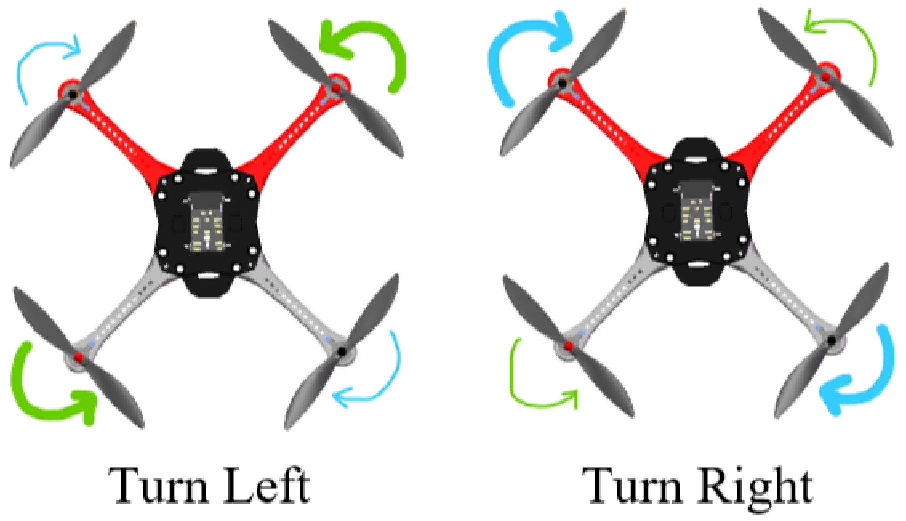


Fig. 2.6. Motor Speeds for Turning Movements.

Longitudinal translations are achieved by varying the relative speeds of the pairs of adjacent propellers at the front and the rear of the quadrotor. This induces a pitch angle which causes forward or backwards motions as shown in Fig 2.7.

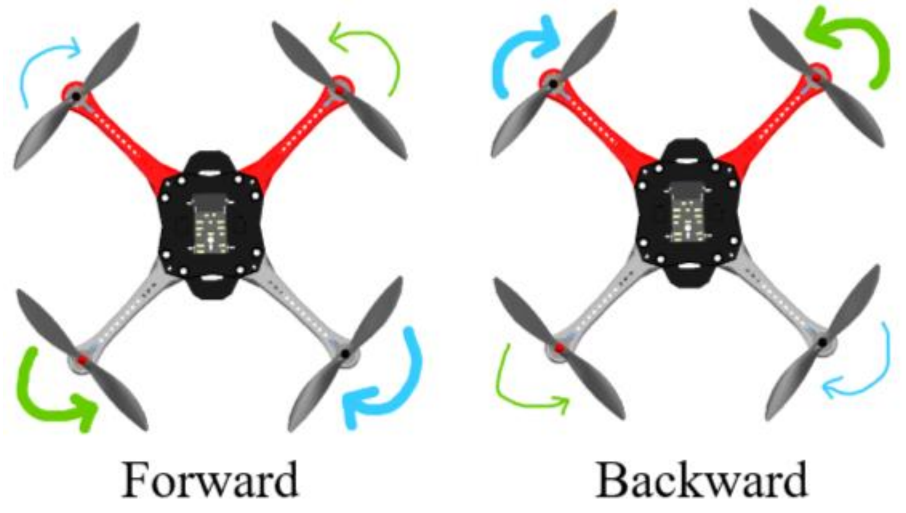


Fig 2.7. Motor Speeds for Longitudinal Movements.

Similarly, lateral translations are achieved by varying the relative speeds of the pairs of adjacent propellers on either side of the quadrotor. This induces a roll angle which causes movements to the left or right as shown in Fig. 2.8.

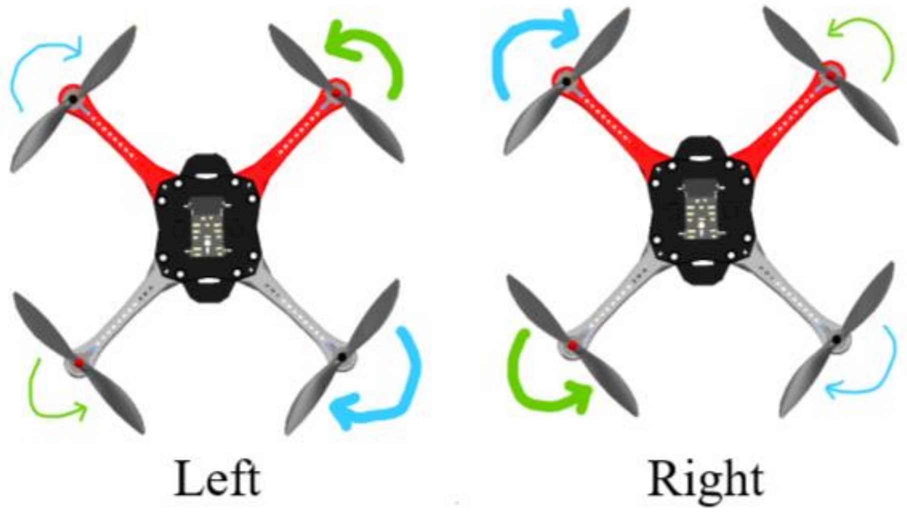


Fig. 2.8. Motor Speeds for Lateral Movements.

2.4 Equations of Motion

Comprehending the dynamic ability of an amphibious AUV to operate in both aerial and underwater environments requires an understanding of two unique sets of operating principles. The equations of motion for a quadrotor operating in-air are well understood and documented due to extensive research in the area. The equations of motion for traditional underwater autonomous vehicles have also been developed and documented in relevant literature. The novel nature of our vehicle presents a unique case which was not found to be documented in the few related publications our team encountered. For this reason, the team performed a derivation of the equations of motion for this type of vehicle operating in an underwater environment. This derivation follows the same procedure as the derivation of the aerial dynamics, while making several key assumptions and defining a different coordinate system. The derivations for the vehicle dynamics follow for both the air and water environments, respectively.

2.4.1 Aerial Dynamics

<u>Coordinate System</u>

Two coordinate systems are used to describe the quadrotor's motion. The Navigation Frame is an inertial frame that is fixed in space at water level. For convenience, the X and Y axes are defined in the longitudinal and lateral directions with respect to the swimming pool. The H axis is directed upward and indicates height above the water's surface. The body frame is fixed to the quadrotor's center of mass such that it translates and rotates with the vehicle. These coordinate frames are defined below in Fig. 2.9.

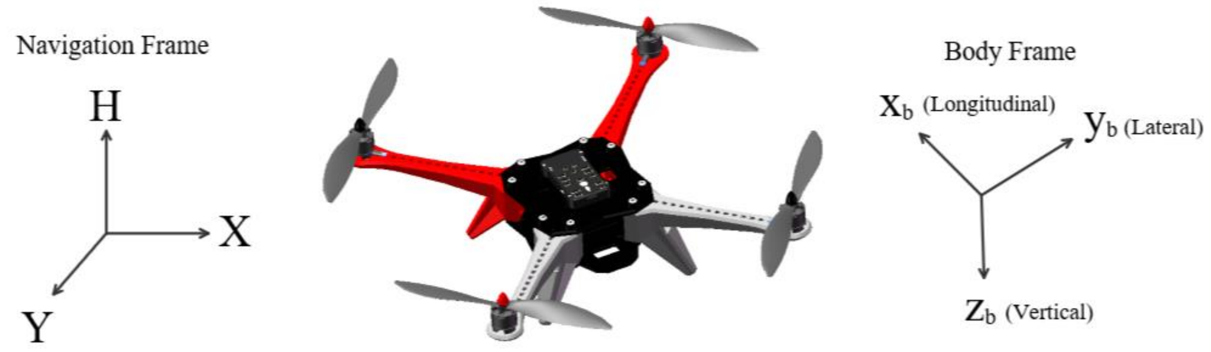


Fig. 2.9. Coordinate Frames for Aerial Dynamics.

<u>Definitions</u>

Linear Velocities in vehicle body frame:

$$\boldsymbol{v^b} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \text{longitudinal velocity} \\ \text{lateral velocity} \\ \text{vertical velocity} \end{bmatrix} \tag{1}$$

Angular Velocities in vehicle body frame:

$$\boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \text{roll rate} \\ \text{pitch rate} \\ \text{yaw rate} \end{bmatrix} \tag{2}$$

Forces acting on quadrotor:

$$\boldsymbol{F} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} \text{force in X direction} \\ \text{force in Y direction} \\ \text{force in Z direction} \end{bmatrix} \tag{3}$$

Moments acting on quad rotor:

$$\boldsymbol{M} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} \text{moment about X axis} \\ \text{moment about Y axis} \\ \text{moment about Z axis} \end{bmatrix} \tag{4}$$

Euler Angles:

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{bank angle} \\ \text{pitch angle} \\ \text{heading} \end{bmatrix} \tag{5}$$

Absolute Position in inertial navigation frame:

$$\boldsymbol{S} = \begin{bmatrix} X \\ Y \\ H \end{bmatrix} = \begin{bmatrix} X \\ Y \\ -Z \end{bmatrix} = \begin{bmatrix} \text{longitudinal position} \\ \text{lateral position} \\ \text{height} \end{bmatrix} \tag{6}$$

Assumption: The vehicle is symmetric about x_b and y_b axes.

Assuming symmetry about the y and z axes is useful to diagonalize the moment of inertia matrix and simplify the resulting calculations.

Representing Inertial Motion in the Vehicle's Body Frame

$$\boldsymbol{v^b} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}^b = u\hat{x}_b + v\hat{y}_b + w\hat{z}_b \tag{7}$$

$$\left(\frac{d\boldsymbol{v^b}}{dt}\right)_{\text{inertial}} = \left(\frac{du}{dt}\hat{x}_b + \frac{dv}{dt}\hat{y}_b + \frac{dw}{dt}\hat{z}_b\right) + \left(u\frac{d\hat{x}_b}{dt} + v\frac{d\hat{y}_b}{dt} + w\frac{d\hat{z}_b}{dt}\right) \tag{8}$$

$$\dot{\boldsymbol{v}}_{\text{inertial}} = \dot{\boldsymbol{v}}^b + \boldsymbol{\omega}_n^b \times \boldsymbol{v}^b \tag{9}$$

Newton's Second Law

$$\boldsymbol{F} = \frac{d\boldsymbol{p}}{dt} = m\frac{d\boldsymbol{v}}{dt} = m\boldsymbol{a} \tag{10}$$

Taking the cross product of $\boldsymbol{r}$ by each side yields the rotational analog:

$$\boldsymbol{r} \times \boldsymbol{F} = \boldsymbol{r} \times m\frac{d\boldsymbol{v}}{dt} \tag{11}$$

$$\boldsymbol{M} = \frac{d\boldsymbol{H}}{dt} = I\frac{d\boldsymbol{\omega}}{dt} = I\boldsymbol{\Omega} \tag{12}$$

Linear accelerations:

Derived from the chain rule:

$$\boldsymbol{F} = m\frac{d\boldsymbol{v}}{dt} \;\rightarrow\; \boldsymbol{F} = m(\dot{\boldsymbol{v}}^b + \boldsymbol{\omega}_n^b \times \boldsymbol{v}^b) \tag{13}$$

Angular accelerations:

Derived from the chain rule:

$$\boldsymbol{M} = I\frac{d\omega}{dt} \;\rightarrow\; \boldsymbol{M} = I\dot{\boldsymbol{\omega}}_n^b + \boldsymbol{\omega}_n^b \times I\boldsymbol{\omega}_n^b \tag{14}$$

External Forces and Moments

Each propeller produces a thrust force perpendicular to the propeller (in the x_b direction) and a torque about the quadrotor's center of gravity. The thrust produced is a function of the propeller geometry, fluid density, and motor speed. The other major forces acting on the vehicle are the buoyant force and gravitational force, which can be assumed to act in opposite directions with equal magnitude. Fig. 2.10 illustrates the forces acting on the quadrotor while operating underwater.

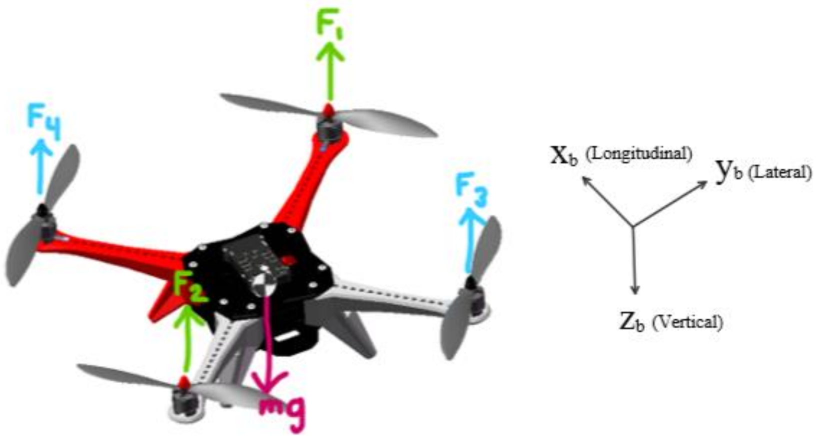


Fig. 2.10. Free Body Diagram of Quadrotor Operating in Air.

Force exerted by propeller thrust:

$$\boldsymbol{F}_{\text{prop}} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -F_1 - F_2 - F_3 - F_4 \end{bmatrix} \tag{15}$$

Gravitational force in body frame:

$$\boldsymbol{F}_{\text{grav}}^n = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \;\rightarrow\; \boldsymbol{F}_{\text{grav}}^b = \begin{bmatrix} -mg\sin(\theta) \\ mg\sin(\phi)\cos(\theta) \\ mg\cos(\phi)\cos(\theta) \end{bmatrix} \tag{16}$$

The torque T which each propeller generates about the quadrotor's center of gravity is a function of the thrust and y, z distances of the motor shaft from the center of gravity. These distances are clearly labeled in Fig. 2.11.

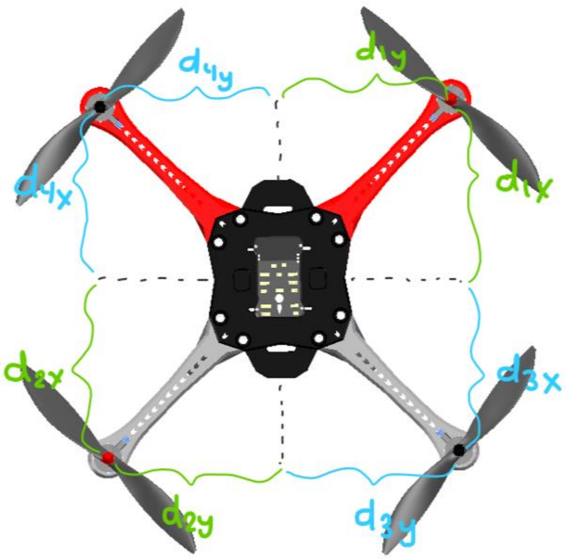


Fig. 2.11. Propeller Locations in Aerial Body Frame.

$$L = F_1 d_{1y} - F_2 d_{2y} - F_3 d_{3y} + F_4 d_{4y} \tag{17}$$

$$M = -F_1 d_{1x} + F_2 d_{2x} - F_3 d_{3x} + F_4 d_{4x} \tag{18}$$

$$N = T\left(F_1, d_{1y,} d_{1z}\right) + T\left(F_2, d_{2y,} d_{2z}\right) - T\left(F_3, d_{3y,} d_{3z}\right) - T(F_4, d_{4y,} d_{4z}) \tag{19}$$

Moment of Inertia Matrix

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \tag{20}$$

Assuming symmetry about the y and z axes yields the diagonal matrix:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{21}$$

<u>Acceleration Equations</u>

Translational Motion:

$$\dot{\boldsymbol{v}}_{\text{inertial}} = \dot{\boldsymbol{v}}^b + \boldsymbol{\omega}_n^b \times \boldsymbol{v}^b \tag{22}$$

$$\dot{\boldsymbol{v}}_{\text{inertial}} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \tag{23}$$

$$\boldsymbol{F} = m\dot{\boldsymbol{v}}^b \tag{24}$$

$$\boldsymbol{F}_{\text{prop}} + \boldsymbol{F}_{\text{grav}} = m\dot{\boldsymbol{v}}_n^b \tag{25}$$

$$\begin{bmatrix} -mg\sin(\theta) \\ mg\sin(\phi)\cos(\theta) \\ -F_1 - F_2 - F_3 - F_4 + mg\cos(\phi)\cos(\theta) \end{bmatrix} = m \begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \tag{26}$$

Rotational Motion:

$$\boldsymbol{M} = I\dot{\boldsymbol{\omega}}_n^b + \boldsymbol{\omega}_n^b \times I\boldsymbol{\omega}_n^b \tag{27}$$

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{28}$$

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} I_{xx}\dot{p} \\ I_{yy}\dot{q} \\ I_{zz}\dot{r} \end{bmatrix} + \begin{bmatrix} -I_{yy}qr + I_{zz}qr \\ I_{xx}pr - I_{zz}pr \\ -I_{xx}pq + I_{yy}pq \end{bmatrix} = \begin{bmatrix} I_{xx}\dot{p} - I_{yy}qr + I_{zz}qr \\ I_{yy}\dot{q} + I_{xx}pr - I_{zz}pr \\ I_{zz}\dot{r} - I_{xx}pq + I_{yy}pq \end{bmatrix} \tag{29}$$

<u>Equations of Motion in Vehicle's Body Frame</u>

Linear accelerations:

$$\dot{u} = -g\sin(\theta) + rr - qw \tag{30}$$

$$\dot{v} = g\sin(\phi)\cos(\theta) - ru + pw \tag{31}$$

$$\dot{w} = \frac{1}{m}(-F_1 - F_2 - F_3 - F_4) + g\cos(\phi)\cos(\theta) + qu - pr \tag{32}$$

Angular accelerations:

$$\dot{p} = \frac{1}{I_{xx}}\left(L + \left(I_{yy} - I_{zz}\right)qr\right) \tag{33}$$

$$\dot{q} = \frac{1}{I_{yy}}\left(M + (I_{zz} - I_{xx})pr\right) \tag{34}$$

$$\dot{r} = \frac{1}{I_{zz}}\left(N + \left(I_{xx} - I_{yy}\right)pq\right) \tag{35}$$

Where:

$$L = F_1 d_{1y} - F_2 d_{2y} - F_3 d_{3y} + F_4 d_{4y} \tag{36}$$

$$M = -F_1 d_{1x} + F_2 d_{2x} - F_3 d_{3x} + F_4 d_{4x} \tag{37}$$

$$N = T\left(F_1, d_{1y,} d_{1z}\right) + T\left(F_2, d_{2y,} d_{2z}\right) - T\left(F_3, d_{3y,} d_{3z}\right) - T(F_4, d_{4y,} d_{4z}) \tag{38}$$

<u>Equations of Motion in Inertial Navigation Frame</u>

Linear accelerations:

$$\dot{X} = (\cos\theta\cos\phi)u + (-\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi)v + (\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi)w \tag{39}$$

$$\dot{Y} = (\cos\theta\sin\psi)u + (\cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi)v + (-\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi)w \tag{40}$$

$$\dot{H} = (\sin\theta)u - (\sin\phi\cos\theta)v + (\cos\phi\cos\theta)w \tag{41}$$

Euler angle rates of change:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & -\cos(\phi)\sec(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{42}$$

2.4.2 Underwater Dynamics

Coordinate System

Two coordinate systems are used to describe the quadrotor's motion. The Navigation Frame is an inertial frame that is fixed in space at water level. For convenience, the X X and YY axes are defined in the longitudinal and lateral directions with respect to the swimming pool. The D H axis is directed downward and indicates depth below the water's surface. The body frame is fixed to the quadrotor's center of mass such that it translates and rotates with the vehicle. These coordinate frames are defined below in Fig. 2.12.

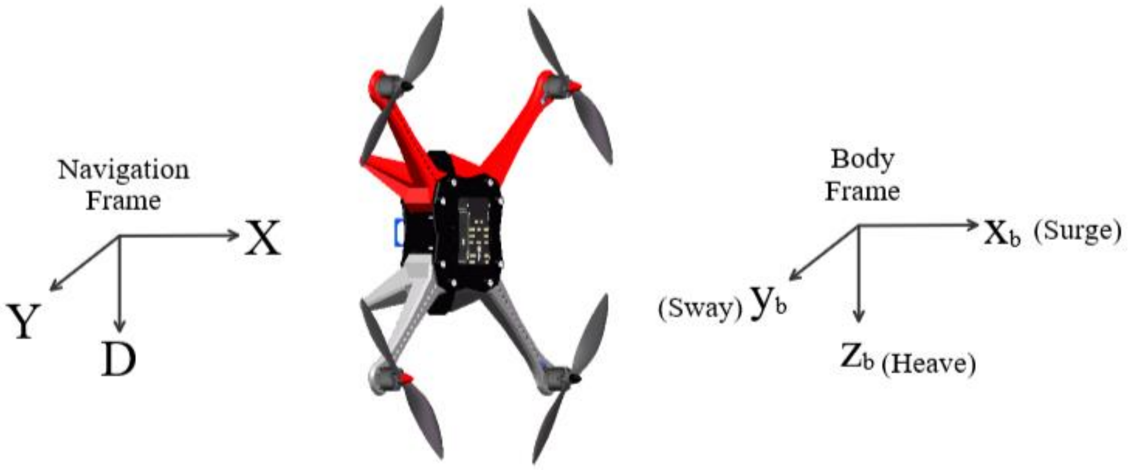


Fig. 2.12. Coordinate Frames for Underwater Dynamics.

Definitions

Linear velocities in vehicle body frame:

$$\boldsymbol{v^b} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \text{surge velocity} \\ \text{sway velocity} \\ \text{heave velocity} \end{bmatrix} \tag{43}$$

Angular velocities in vehicle body frame:

$$\boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \text{roll rate} \\ \text{pitch rate} \\ \text{yaw rate} \end{bmatrix} \tag{44}$$

Forces acting on quadrotor:

$$\boldsymbol{F} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} \text{force in X direction} \\ \text{force in Y direction} \\ \text{force in Z direction} \end{bmatrix} \tag{45}$$

Moments acting on quadrotor:

$$\boldsymbol{M} = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} \text{moment about X axis} \\ \text{moment about Y axis} \\ \text{moment about Z axis} \end{bmatrix} \tag{46}$$

Euler angles:

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{bank angle} \\ \text{pitch angle} \\ \text{heading} \end{bmatrix} \tag{47}$$

Absolute position in inertial navigation frame:

$$\boldsymbol{S} = \begin{bmatrix} X \\ Y \\ D \end{bmatrix} = \begin{bmatrix} \text{longitudinal position} \\ \text{lateral position} \\ \text{depth} \end{bmatrix} \tag{48}$$

Assumption: The vehicle is neutrally buoyant.
Assumption: The center of buoyancy lies directly above the center of gravity.
Assumption: The vehicle is symmetric about the y_b and z_b axes.

These assumptions are important as they allow the vehicle passive stability underwater. A tilt angle in any direction results in a restoring couple due to the misalignment of the buoyant force and gravitational force, giving the vehicle the tendency to remain upright by passively stabilizing pitch and roll. The neutral buoyancy assumption further allows the vehicle to passively maintain its depth beneath the water's surface without floating or sinking. Assuming symmetry about the y and z axes is useful to diagonalize the moment of inertia matrix to simplify the calculations.

Having established the above assumptions, the buoyant force and gravitational force can be neglected in our static force analysis because they will always act in opposite directions with equal magnitude and effectively cancel out. This is a reasonable assumption for small pitch and roll angles which are expected during nominal operation of the vehicle underwater.

<u>Representing Inertial Motion in the Vehicle's Body Frame</u>

$$\boldsymbol{v^b} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}^b = u\hat{x}_b + v\hat{y}_b + w\hat{z}_b \tag{49}$$

$$\left(\frac{d\boldsymbol{v^b}}{dt}\right)_{\text{inertial}} = \left(\frac{du}{dt}\hat{x}_b + \frac{dv}{dt}\hat{y}_b + \frac{dw}{dt}\hat{z}_b\right) + \left(u\frac{d\hat{x}_b}{dt} + v\frac{d\hat{y}_b}{dt} + w\frac{d\hat{z}_b}{dt}\right) \tag{50}$$

$$\dot{\boldsymbol{v}}_{\text{inertial}} = \dot{\boldsymbol{v}}^b + \boldsymbol{\omega}_n^b \times \boldsymbol{v}^b \tag{51}$$

<u>Newton's Second Law</u>

$$\boldsymbol{F} = \frac{d\boldsymbol{p}}{dt} = m\frac{d\boldsymbol{v}}{dt} = m\boldsymbol{a} \tag{52}$$

Taking the cross product of $\boldsymbol{r}$ by each side yields the rotational analog:

$$\boldsymbol{r} \times \boldsymbol{F} = \boldsymbol{r} \times m\frac{d\boldsymbol{v}}{dt} \tag{53}$$

$$\boldsymbol{M} = \frac{d\boldsymbol{H}}{dt} = I\frac{d\boldsymbol{\omega}}{dt} = I\boldsymbol{\Omega} \tag{54}$$

<u>Linear accelerations</u>:

Derived from the chain rule:

$$\boldsymbol{F} = m\frac{d\boldsymbol{v}}{dt} \;\rightarrow\; \boldsymbol{F} = m(\dot{\boldsymbol{v}}^b + \boldsymbol{\omega}_n^b \times \boldsymbol{v}^b) \tag{55}$$

<u>Angular accelerations</u>:

Derived from the chain rule:

$$\boldsymbol{M} = I\frac{d\omega}{dt} \;\rightarrow\; \boldsymbol{M} = I\dot{\boldsymbol{\omega}}_n^b + \boldsymbol{\omega}_n^b \times I\boldsymbol{\omega}_n^b \tag{56}$$

<u>External Forces and Moments</u>

Each propeller produces a thrust force perpendicular to the propeller (in the x_b direction) and a torque about the quadrotor's center of gravity. The thrust produced is a function of the propeller geometry, fluid density, and motor speed. The other major forces acting on the vehicle are the buoyant force and gravitational force, which can be assumed to act in opposite directions with equal magnitude. Fig. 2.13 illustrates the forces acting on the quadrotor while operating underwater.

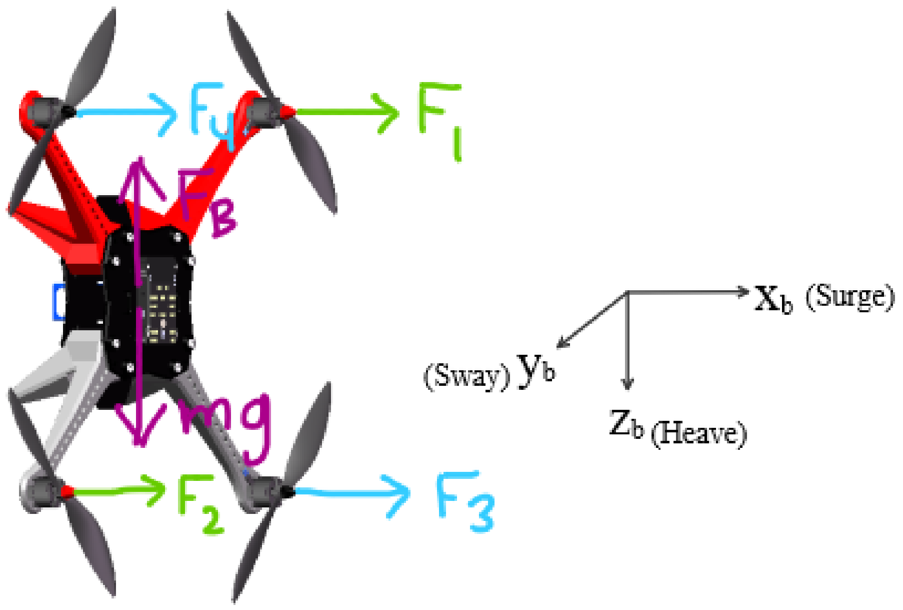


Fig. 2.13. Free Body Diagram of Quadrotor Operating Underwater.

Force exerted by propeller thrust:

$$\boldsymbol{F}_{\text{prop}} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} F_1 + F_2 + F_3 + F_4 \\ 0 \\ 0 \end{bmatrix} \tag{57}$$

The torque T which each propeller generates about the quadrotor's center of gravity is a function of the thrust and y, z distances of the motor shaft from the center of gravity. These distances are clearly labeled in Fig. 2.14.

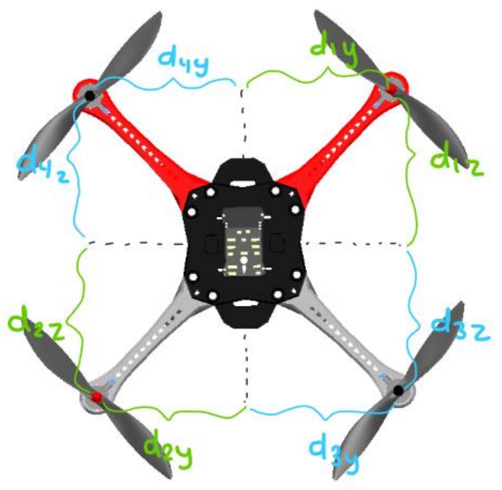


Fig. 2.14. Propeller Locations in Underwater Body Frame.

$$L = T\left(F_1, d_{1y,} d_{1z}\right) + T\left(F_2, d_{2y,} d_{2z}\right) - T\left(F_3, d_{3y,} d_{3z}\right) - T(F_4, d_{4y,} d_{4z}) \tag{58}$$

$$M = -F_1 d_{1z} + F_2 d_{2z} + F_3 d_{3z} - F_4 d_{4z} \tag{59}$$

$$N = F_1 d_{1y} - F_2 d_{2y} + F_3 d_{3y} - F_4 d_{4y} \tag{60}$$

<u>**Moment of Inertia Matrix**</u>

$$I = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \tag{61}$$

Assuming symmetry about the y and z axes yields the diagonal matrix:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{62}$$

<u>**Acceleration Equations**</u>

Translational Motion:

$$\dot{\boldsymbol{v}}_{\text{inertial}} = \dot{\boldsymbol{v}}^b + \boldsymbol{\omega}_n^b \times \boldsymbol{v}^b \tag{63}$$

$$\dot{\boldsymbol{v}}_{\text{inertial}} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \tag{64}$$

$$\boldsymbol{F} = m\dot{\boldsymbol{v}}^b \tag{65}$$

$$\boldsymbol{F}_{\text{prop}} = m\dot{\boldsymbol{v}}^b \tag{66}$$

$$\begin{bmatrix} F_1 + F_2 + F_3 + F_4 \\ 0 \\ 0 \end{bmatrix} = m \begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \tag{67}$$

Rotational Motion:

$$\boldsymbol{M} = I\dot{\boldsymbol{\omega}}_n^b + \boldsymbol{\omega}_n^b \times I\boldsymbol{\omega}_n^b \tag{68}$$

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{69}$$

$$\begin{bmatrix} L \\ M \\ N \end{bmatrix} = \begin{bmatrix} I_{xx}\dot{p} \\ I_{yy}\dot{q} \\ I_{zz}\dot{r} \end{bmatrix} + \begin{bmatrix} -I_{yy}qr + I_{zz}qr \\ I_{xx}pr - I_{zz}pr \\ -I_{xx}pq + I_{yy}pq \end{bmatrix} = \begin{bmatrix} I_{xx}\dot{p} - I_{yy}qr + I_{zz}qr \\ I_{yy}\dot{q} + I_{xx}pr - I_{zz}pr \\ I_{zz}\dot{r} - I_{xx}pq + I_{yy}pq \end{bmatrix} \tag{70}$$

Equations of Motion in Vehicle's Body Frame

Linear accelerations:

$$\dot{u} = \frac{1}{m}(F_1 + F_2 + F_3 + F_4) + qw - rv \tag{71}$$

$$\dot{v} = -ru + pw \tag{72}$$

$$\dot{w} = -pv + qu \tag{73}$$

Angular accelerations:

$$\dot{p} = \frac{1}{I_{xx}}\left(L + \left(I_{yy} - I_{zz}\right)qr\right) \tag{74}$$

$$\dot{q} = \frac{1}{I_{yy}}\left(M + (I_{zz} - I_{xx})pr\right) \tag{75}$$

$$\dot{r} = \frac{1}{I_{zz}}(N + (I_{xx} - I_{yy})pq) \tag{76}$$

Where:

$$L = T(F_1, d_{1y,}d_{1z}) + T(F_2, d_{2y,}d_{2z}) - T(F_3, d_{3y,}d_{3z}) - T(F_4, d_{4y,}d_{4z}) \tag{77}$$

$$M = -F_1 d_{1z} + F_2 d_{2z} + F_3 d_{3z} - F_4 d_{4z} \tag{78}$$

$$N = F_1 d_{1y} - F_2 d_{2y} + F_3 d_{3y} - F_4 d_{4y} \tag{79}$$

<u>Equations of Motion in Inertial Navigation Frame</u>

Linear accelerations:

$$\begin{aligned}\dot{X} = {} & (\cos\theta\cos\psi)u + (-\cos\theta\sin\psi + \sin\phi\sin\theta\cos\psi)v \\ & + (\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi)w\end{aligned} \tag{80}$$

$$\begin{aligned}\dot{Y} = {} & (\cos\theta\sin\psi)u + (\cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi)v \\ & + (-\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi)w\end{aligned} \tag{81}$$

$$\dot{D} = (-\sin\theta)u + (\sin\phi\cos\theta)v + (\cos\phi\cos\theta)w \tag{82}$$

Euler angle rates of change:

$$\begin{bmatrix}\dot{\phi} \\ \dot{\theta} \\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & -\cos(\phi)\sec(\theta)\end{bmatrix}\begin{bmatrix}p \\ q \\ r\end{bmatrix} \tag{83}$$

2.5 Autonomy and Localization

The concept of autonomy within the field of aircraft has been an ongoing research topic for the past few decades. The conversation centers around the two main questions of navigation: Where am I now? And where am I going? The answers to these questions are essential for the construction of a guidance system that would allow for autonomous flight to be achieved. The exact methods for creating an autonomous vehicle vary, but the main system design consists of two major components: inertial sensing and localization. Autonomous vehicles must be outfitted with a sensor array for data collection to provide the system with answers to the two central questions governing navigation and guidance [4].

A sensor array may include a multitude of sensors, but overall, the data collected must include the linear accelerations, the pitch, roll and yaw rates, and position of the aircraft. The vehicle’s accelerations and angular rotation rates are typically measured with inertial sensors [4]. UAV’s typically feature tri-axis accelerometers to measure the linear accelerations of the aircraft as well as rate gyroscopes for the measurement of the pitch, roll, and yaw rates of rotation. This data allows for the determination of both the attitude of the aircraft and speed and direction it is moving. Traditionally, a GPS (global positioning system) module is included to determine the vehicle’s position regarding its longitude and latitude, along with an altimeter to determine the system's distance from the planet’s surface below. Combining all this data together, the aircraft is successfully able to determine its position while in flight and where it is heading next [15].

Localization can be defined as the determination of a vehicle’s absolute position with respect to its environment. This task is a much greater challenge for underwater vehicles because GPS systems lose satellite connections and become unreliable. Several alternative localization methods include the use of LiDAR, optical or radio beacons, or onboard cameras which utilize various computer vision algorithms [16]. One such algorithm is optical flow, which is a technique that tracks the motion of identifiable features from frame to frame. Alternatively, computer vision can be used to identify the location of the vehicle relative to fiducial marker placed in its environment. Doppler velocity log (DVL) is another localization technique specifically used by underwater vehicles. This sensor determines the rate and direction at which the craft is moving via doppler shift, bouncing sound waves off the base of the body of water. AUV’s must use these or similar methods to determine their position due to inability to achieve GPS lock. Traditional altimeters which operate on barometric pressure readings are also unusable for underwater vehicles and an alternative depth sensing solution is required. This is commonly done with a waterproof pressure transducer. Directly measuring the depth below the water’s surface provides a useful datapoint to augment the chosen localization method for accurate determination of a vehicle’s position in 3-dimensional space [15].

With the data collected from inertial sensors and localization systems, the next step of the system is to process these measurements for use in vehicle guidance. First, some signal processing is performed with a low-pass filter to remove high-frequency noise from accelerometer and gyroscope data. The improved data is then passed to a Kalman filter to ensure the data being used is as accurate as possible. The Kalman filter implements sensor fusion while taking into account the intrinsic accuracies and biases of each sensor to produce the best possible estimate of state variables. With this now filtered data, the vehicle is able to determine its current state and the necessary movements to get to the wanted second state [17]. The aircraft will continue to use the incoming data to determine its next steps until the mission it was sent on is complete, allowing the vehicle to operate with little to no input from the operator, creating a simplified and safer user experience. Importantly, autonomy enables vehicles to operate in locations and environments where remote piloting is not possible.

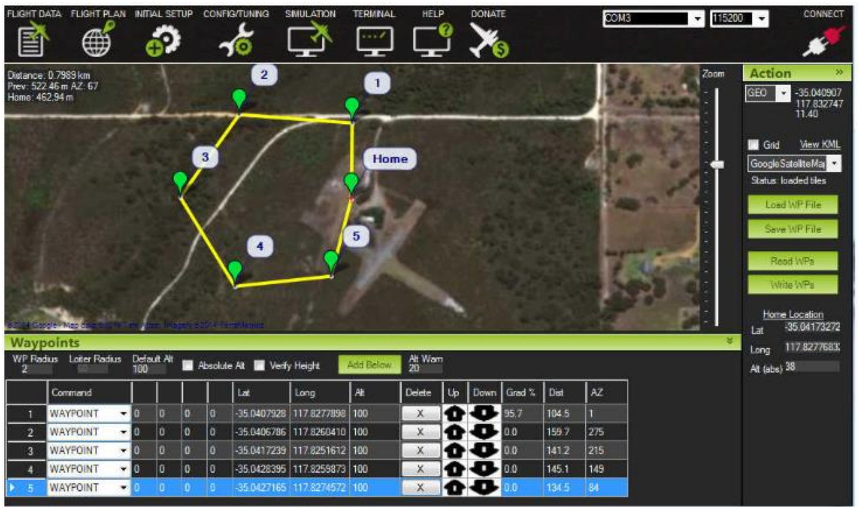
3 Technical Sections

3.1 Quadrotor Flight Controllers

3.1.1 Commercially Available Flight Controllers

When deciding on a flight controller to use to control the quadrotor, the controls team compared the options of commercially available flight controllers and the creation of a custom flight controller. The abilities of each controller were weighed with the importance of functionality and customizability in mind. First, commercially available controllers were researched and compared in an attempt to find a controller that was already built and functional that could be customized for the specific needs of the project.

Two popular controllers that are available are the PixHawk and Navio 2. Both operate with ArduPilot and PX4 flight control software. PX4 is an open-source autopilot for drone development and is suitable for both aircraft and underwater vehicle control. ArduPilot is a similar controller designed for multiple types of aircraft with a created “Mission Planner” software that makes it user friendly and simple to learn. The team decided to move forward with ArduPilot due to its intuitive design and the multitude of resources available for reference when it comes to development of a controller for a new aircraft. For the hardware portion of the initial controller design, the team compared the capabilities of the Navio2 and Pixhawk, finding the PixHawk to be better suited to the needs of the aircraft in development.



COM3 115200

Distance: 0.7989 km
Prev: 522.46 m AZ: 67
Home: 462.94 m

GEO -35.040907 117.832747 11.40

Grid View KML
GoogleSatelliteMap
Status: loaded tiles

Home Location
Lat -35.04173272
Long 117.8277683
Alt (abs) 38

WP Radius 2 Loiter Radius Default Alt 100 Absolute Alt Verify Height Alt Warn 20

	Command					Lat	Long	Alt	Delete	Up	Down	Grad %	Dist	AZ
1	WAYPOINT	0	0	0	0	-35.0407928	117.8277898	100	X			95.7	104.5	1
2	WAYPOINT	0	0	0	0	-35.0406786	117.8260410	100	X			0.0	159.7	275
3	WAYPOINT	0	0	0	0	-35.0417239	117.8251612	100	X			0.0	141.2	215
4	WAYPOINT	0	0	0	0	-35.0428395	117.8259873	100	X			0.0	145.1	149
5	WAYPOINT	0	0	0	0	-35.0427165	117.8274572	100	X			0.0	134.5	84

Fig. 3.1. Mission Planner User Interface.

With these selections made, the team moved on to test flight control using a test drone and a PixHawk with ArduPilot Mission Planner. Initial testing with the system provided insight into the capabilities of commercially available flight controllers, and their customizability. Our aircraft’s unique configuration and mission makes the need for customization in a flight controller a top priority. ArduPilot and other software like it does not allow for the configuration of the aircraft to change while in flight, so it was not well suited to our needs.

3.1.2 Custom Flight Controllers

After the team ruled out the possibility of using an existing commercial flight controller, the focus shifted to custom flight controllers. There are a multitude of microcontrollers suitable for the construction of a flight controller including the MSP430, Teensy, and Arduino. All of these microcontrollers operate using C/C++ code that can be customized for our project's needs. While searching for a hardware solution, we also were searching for a possible software architecture to be able to start the construction of the controller. That's when we landed on dRehmFlight VTOL, a customizable flight control software package that is designed for quadrotors and people who want the ability to fully customize the controller to their needs. The controller package recommended that it be run on a Teensy microcontroller and uses C/C++ language in the Arduino IDE, making it the best choice for our aircraft.

3.1.3 dRehmFlight VTOL

After the testing and comparison of the flight controller options available, the controls team settled on the dRehmFlight controller that operates on a Teensy 4.0 microcontroller with Arduino software. The controller code was created by Nicholas Rehm and is meant for multirotor aircraft stability using the Arduino environment to allow for ease of use and customization. The controller was created with the intention of providing coders and aviators with the toolbox they need to create their own custom rotorcraft projects. The coding package comes with a predefined controller structure with all necessary functions included for flight control. [18]

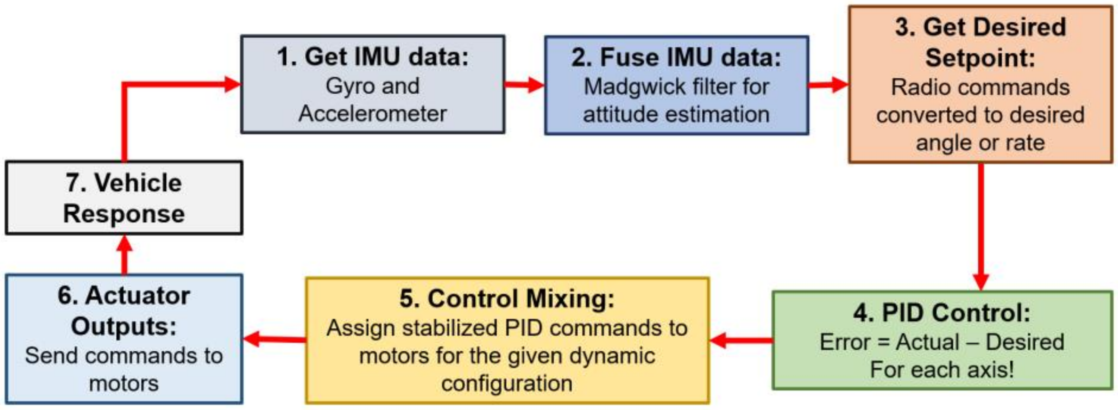


Fig. 3.2. dRehmFlight Controller Architecture. [18]

This control structure contains all radio inputs, actuator outputs, sensor filtering, and PID controllers built that are necessary to successfully control a custom aircraft. The user is able to follow a guide to make specific customizations to the setup, structure, and mixer for their specific aircraft. These aspects of the dRehmFlight controller package make it ideal for the design of the quadrotor. [18]

3.2 Electronics

The electronics on the quadrotor are organized into four major subsystems. These subsystems include power, propulsion, sensing and localization, and the ballast system. The details of each subsystem are provided in the remainder of this section. At the center of the electrical system is a Teensy 4.0 microcontroller that serves as the primary flight computer (pictured in Fig. 3.3Fig. . This microcontroller was chosen because of its ease of programming, compatibility with dRehmFlight VTOL, and superior processing power compared to alternatives like the Arduino Nano. The Teensy 4.0 is responsible for executing the main flight control loop, performing the bulk of the processing, commanding the motors, and interfacing with the various peripheral devices. The microcontroller is mounted on a prototyping PCB with terminals for the various input and output devices. An image of this board is shown in Fig. 3.4 with the components labeled. Additionally, the functional block diagram shown in Fig. 3.5 illustrates the relationships between the major components of the electrical system.

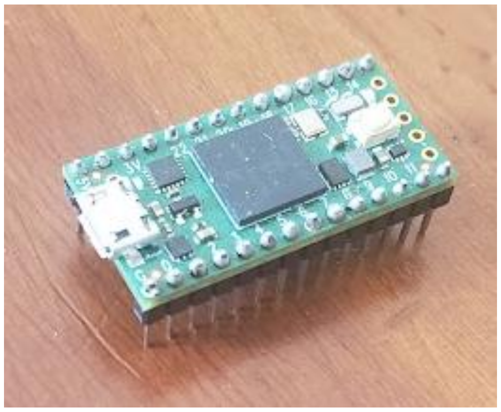


Fig. 3.3. Teensy 4.0 Microcontroller.

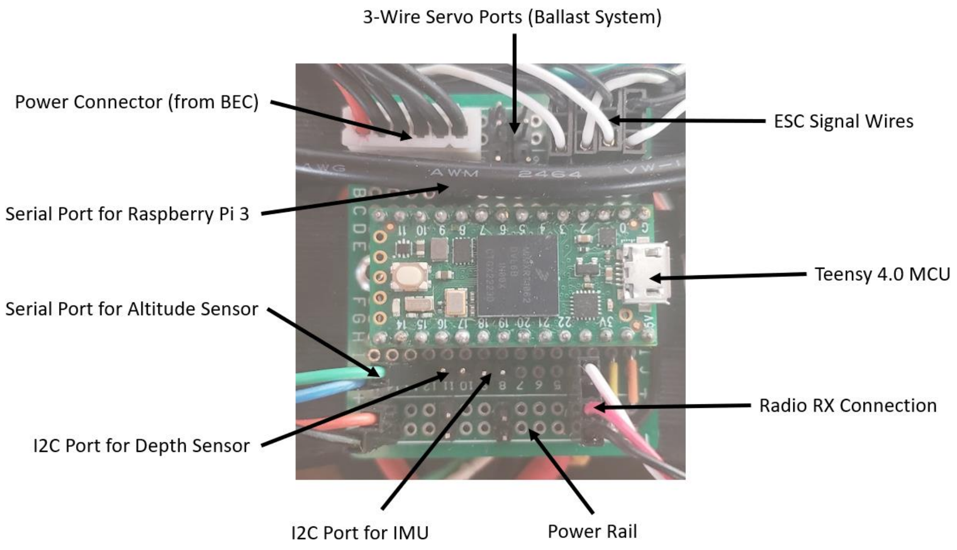

3-Wire Servo Ports (Ballast System)
Power Connector (from BEC)
ESC Signal Wires
Serial Port for Raspberry Pi 3
Teensy 4.0 MCU
Serial Port for Altitude Sensor
Radio RX Connection
I2C Port for Depth Sensor
I2C Port for IMU
Power Rail

Fig. 3.4. Main Electronics Board with Components Labeled.

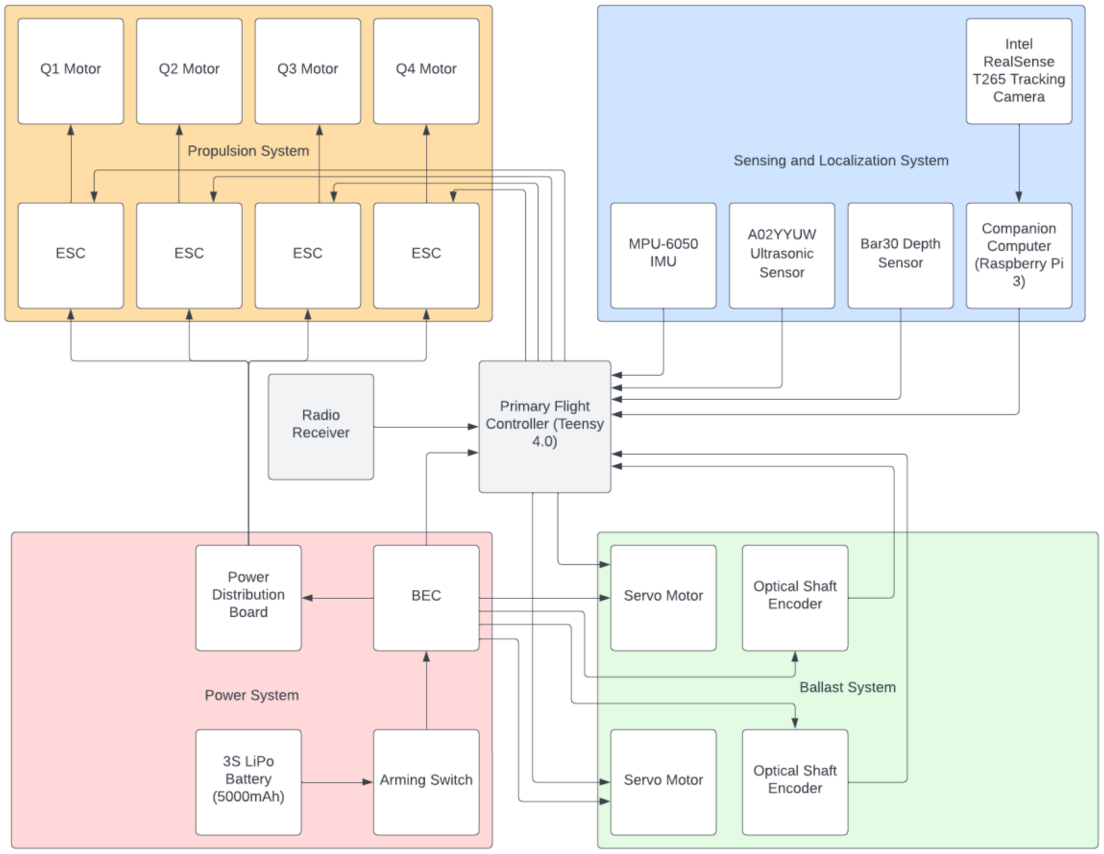
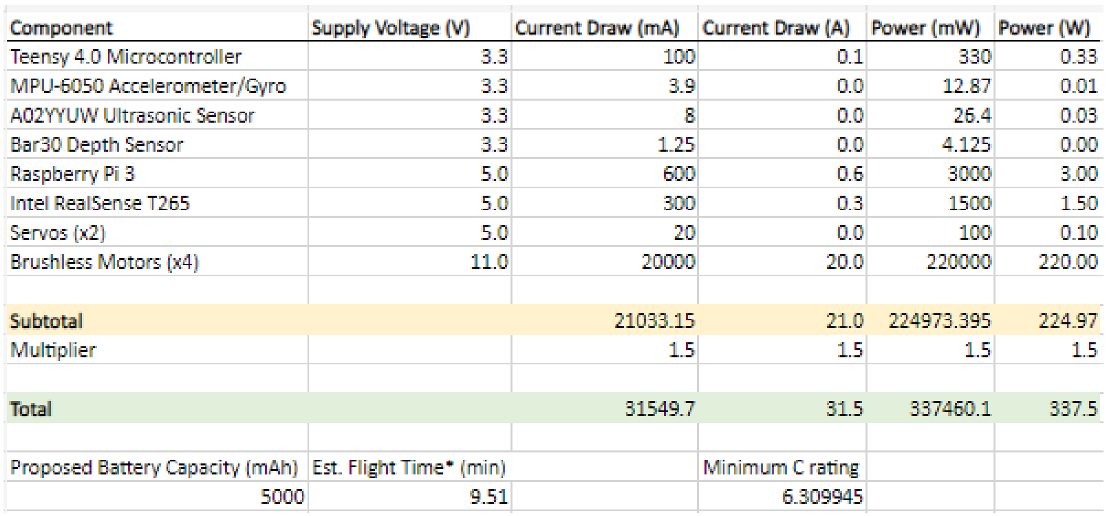


Fig. 3.5. Functional Block Diagram of the Electrical System.

3.2.1 Power Subsystem

The first electrical subsystem to discuss is the power system. The power system is responsible for applying electrical power to the remaining components. The system is powered by a 3S lithium polymer (LiPo) battery. The selected battery capacity was 5000mAh as determined from an analysis of the power budget show in Fig. 3.6. The 500mAh capacity was chosen to give a flight time of about 10 minutes with the motors assumed to be running continuously at 50% throttle. This battery sizing was driven by the power budget analysis for aerial flight due to the greater power requirements compared to aquatic operation. The battery is connected to a battery eliminator circuit (BEC) which then connects to a power distribution board (PDB). The BEC is essentially a voltage regulator that passes the full 11.1V from the battery to the PDB while providing a second output that has been dropped down to 5V. The PDB is used to deliver the battery voltage to the propulsion system. A large capacitor is connected to the battery terminal on the PDB to serve as a brownout capacitor, storing charge in order to smooth out the power supply in response to transients caused by the motors. The 5V output from the BEC is wired to the power and ground rails on the main circuit board. These rails are used to power the microcontroller and the components of the sensing and localization system and ballast system.



Component	Supply Voltage (V)	Current Draw (mA)	Current Draw (A)	Power (mW)	Power (W)
Teensy 4.0 Microcontroller	3.3	100	0.1	330	0.33
MPU-6050 Accelerometer/Gyro	3.3	3.9	0.0	12.87	0.01
A02YYUW Ultrasonic Sensor	3.3	8	0.0	26.4	0.03
Bar30 Depth Sensor	3.3	1.25	0.0	4.125	0.00
Raspberry Pi 3	5.0	600	0.6	3000	3.00
Intel RealSense T265	5.0	300	0.3	1500	1.50
Servos (x2)	5.0	20	0.0	100	0.10
Brushless Motors (x4)	11.0	20000	20.0	220000	220.00
Subtotal		21033.15	21.0	224973.395	224.97
Multiplier		1.5	1.5	1.5	1.5
Total		31549.7	31.5	337460.1	337.5
Proposed Battery Capacity (mAh)	Est. Flight Time* (min)		Minimum C rating		
5000	9.51		6.309945		

Fig. 3.6. Power Budget for Quadrotor Design.

3.2.2 Propulsion Subsystem

The propulsion system is responsible for moving the quadrotor through the air and through the water. The major components of the propulsion system are the motors and electronic speed controllers (ESCs). The quadrotor features four three-phase brushless DC motors for propulsion. The motor chosen for the vehicle was the Mt2213 935kV Brushless Motor manufactured by Emax. These motors are 27.9mm in diameter and have a mass of 53g. This model was selected because it offers sufficient thrust for the expected mass of the quadrotor and has an excellent thrust to mass ratio. Brushless motors are also easily waterproofed which makes them an ideal candidate for our application in designing and amphibious AUV. Fig. 3.7 depicts the selected motors. The motors were equipped with 1045 propellers, corresponding to a length of 10 inches and a blade pitch angle of 4.5 degrees. The three wires from each motor are connected to the output terminals of an electronics speed controller. Bullet connectors were used in the wiring between the motors and ESCs to allow for them to be easily connected and disconnected as needed. The chosen ESCs are rated up to 35A, which comfortably exceeds the expected load from our motors. The input terminals to the ESCs are wired to the power distribution board in order to supply 11.1V to the motors. The signal wires from the ESC are wired to the main circuit board to receive commands from the microcontroller. The 4 in 1 ESC used for the final vehicle is shown in Fig. 3.8.

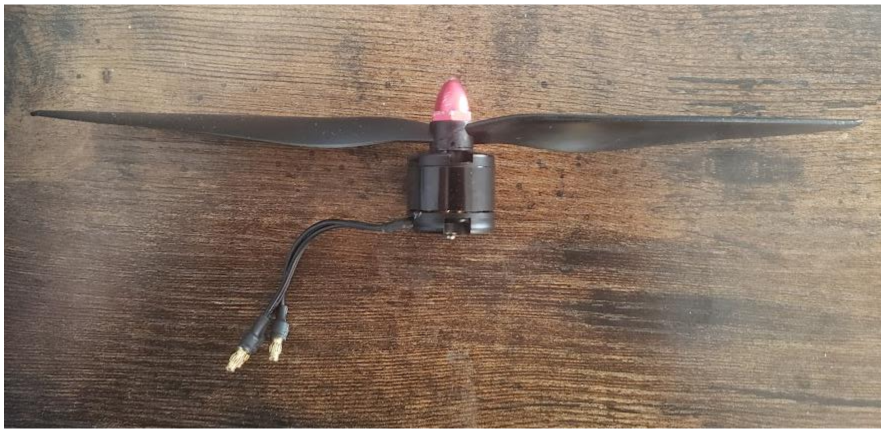


Fig. 3.7. Emax Mt2213 935kV Brushless Motor with 1045 Propeller.

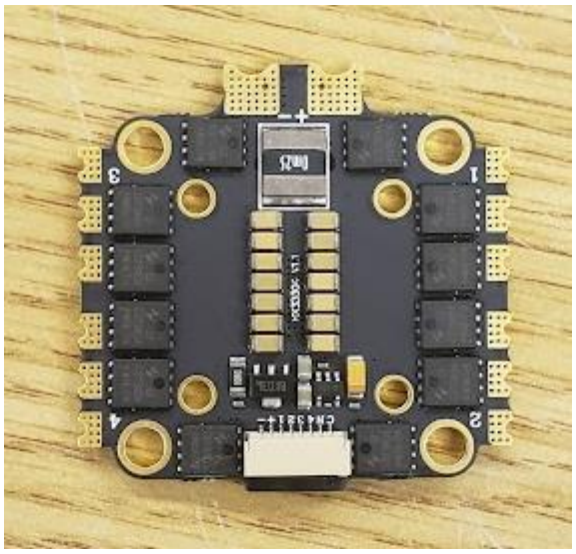


Fig. 3.8. 4 in 1 Electronic Speed Controller.

3.2.3 Sensing and Localization Subsystem

The sensing and localization system is responsible for helping the quadrotor to make sense of the world around it. To complete the autonomous mission successfully, the vehicle must know its altitude during aerial operation, depth underwater during aquatic operation, and its position and orientation in space at all times. This is accomplished with the use of four sensors: an inertial measurement unit (IMU), ultrasonic rangefinder, pressure-based depth sensor, and tracking camera. More detailed information on the specifics of the chosen sensors is given in Section 3.3.1. The IMU, rangefinder, and pressure sensor are all wired to the main circuit board and read directly by the microcontroller. The tracking camera is more complex and requires the use of a companion computer to perform complex computation. A Raspberry Pi 3 microprocessor (shown in Fig. 3.9) is used to interface with the tracking camera. The Raspberry Pi then communicates pose data from the camera to the primary flight computer over Serial.

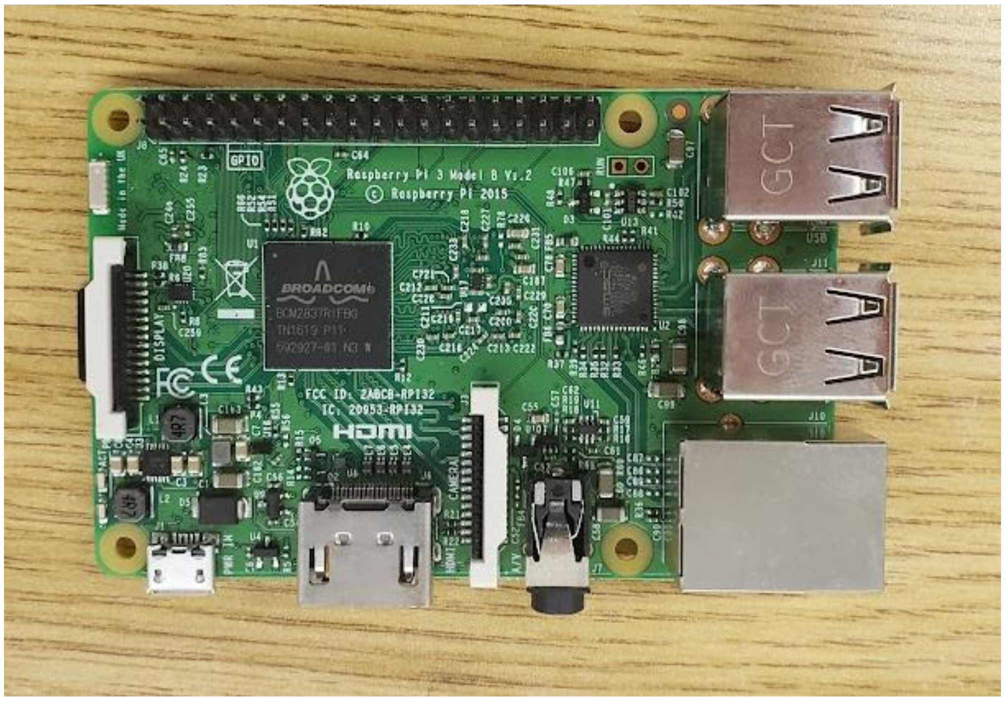


Fig. 3.9. Raspberry Pi 3 Microprocessor.

3.2.4 Ballast System Electronics

The ballast system electronics include the sensors and actuators needed to control the volume of water within the ballast tanks. The system is actuated by a pair of servo motors, with one for each syringe. The servos are SG-90 continuous rotation hobby servos which are powered from the 5V rail on the main circuit board. Continuous rotation servo motors were required due to the stroke length of the rack and pinion linear actuators which extend and retract the piston within the cylinders. More information is provided in Section 3.8. A pair of 5-wire optical shaft encoders are used to measure the linear displacement of the piston within each cylinder. These devices are also wired to the main circuit board. The Teensy 4.0 microcontroller reads the signals from the shaft encoders to determine the present state of the ballast system, and commands actuation of the servos when directed by the flight control loop. The SG-90 continuous rotation servo is shown in Fig. 3.10 and the optical shaft encoder is shown in Fig. 3.11.

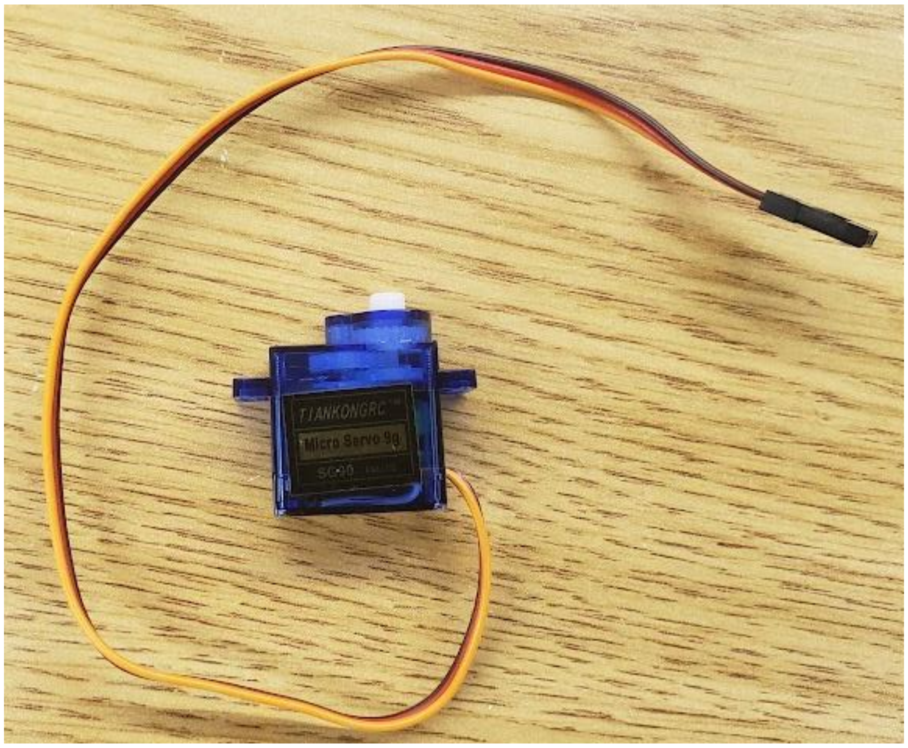


Fig. 3.10. SG-90 Continuous Rotation Servo Motor.

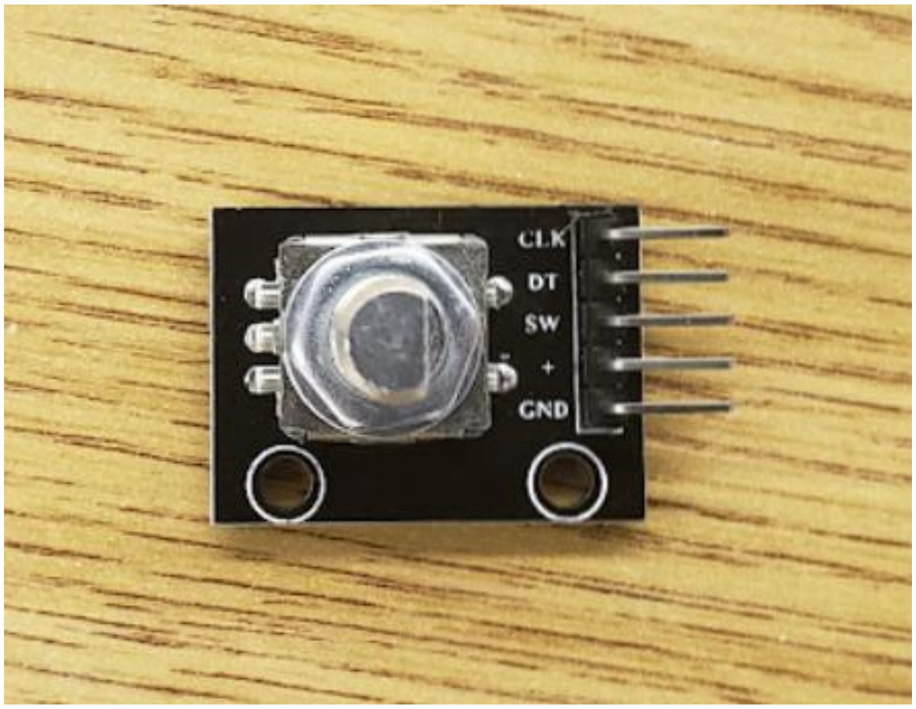


Fig. 3.11. 5-Wire Optical Shaft Encoder.

3.3 State Estimation

3.3.1 Onboard Sensors

Attached to the Teensy 4.0 microcontroller is a complete sensor array to allow for the controller to collect all of the information it needs to achieve stable flight. The Teensy is the wanted flight controller as it is able to run the controller code at high speeds, making it the ideal choice for our aircraft. The controller code takes in data from the IMU, altitude sensor, depth sensor, and tracking camera to allow it to know its position as well as send commands to the motors in order to remain stable while in flight.

3.3.1.1 IMU

For the IMU, the controls team decided on the MPU6050 as it was the model recommended by the dRehmFlight controller. The MPU6050 provides the controller with six axes of data with its internal Gyroscope and Accelerometer. The Gyroscope provides three axes of rotational velocity, and the Accelerometer provides three axes of measured gravitational acceleration. This data provides the controller with knowledge of the motion of the quadrotor along the three axes of motion as well as the orientation of the quadrotor while in flight.

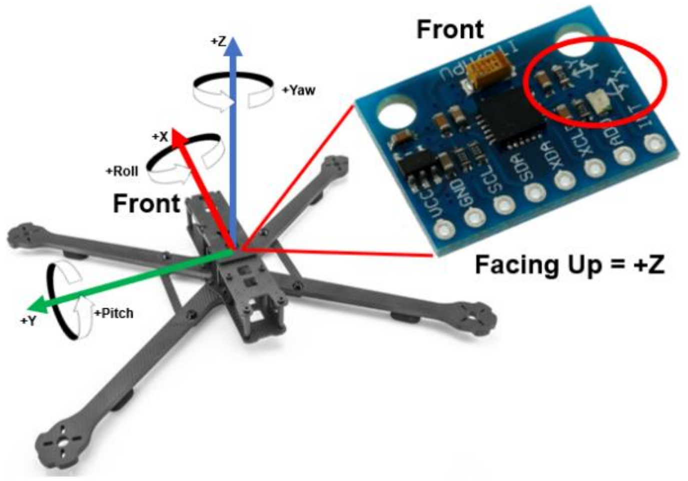


Fig. 3.12. IMU Orientation on Quadrotor. [18]

3.3.1.2 Altitude Sensor

For the altitude sensor the controls team decided to use the A02YYUW Waterproof Ultrasonic sensor. With the design of the quadrotor needing to account for the craft being able to be operated both above ground and above and underwater, the sensor used for determining altitude needed to alot for those aspects of the design. A sensor utilizing lasers was ruled out as an option due to the reflective nature of the surface of the pool that could mess with the accuracy of the readings. For the purpose of our design, an ultrasonic sensor that operates using ultrasonic pulses to determine the distances of objects from the sensor was the best option. The ultrasonic sensor is also waterproof and able to be placed underwater without having to be inside the enclosure, allowing it to be able to provide the necessary position data needed by the flight controller to determine its current location and state. [19]

Fig. 3.13. A02YYUW Waterproof Ultrasonic sensor. [19]

3.3.1.3 Depth Sensor

Once underwater, to continue to track the vertical position of the quadrotor, a depth sensor was added to the array. Upon selection, the controls team selected the Bar30 High-Resolution 300m Depth/Pressure Sensor, as it best suited our needs concerning the design of the craft. The sensor that was selected operates up to 300 meters deep within a body of water and is accurate within 2mm when in freshwater. This sensor will provide the controller with the location of the craft while underwater, allowing it to successfully maintain a specified depth while moving across the pool as well as assistance with the air to water transition. [20]

Fig. 3.14. Bar30 High-Resolution 300m Depth/Pressure Sensor. [20]

3.3.1.4 Tracking Camera

When it comes to state estimation, one of the simplest routes for an aircraft is using GPS to determine the aircraft's position in air. When it comes to a craft that operates underwater, GPS is no longer an option. So, for the purposes of this project, state and location estimation of the quadrotor needed to fall on the abilities of a complete sensor array with the ability of localization. To obtain this needed position data, the controls team decided a tracking camera would be the best course of action when it came to sensor selection for this task.

3.3.1.4.1 Camera Selection

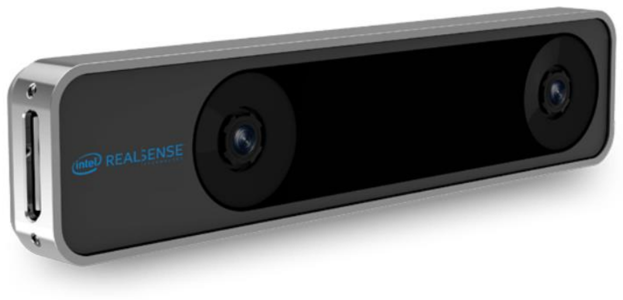


Fig. 3.15. Intel RealSense T265 Tracking Camera. [21]

To best suit our needs, we needed a camera that tracked itself, not just objects in its field of view. The T-265 tracking camera made by Intel fulfills these requirements. Intel has a variety of tracking and depth cameras that are able to determine the location of the robot attached to it, but the T265 is the one that best suits the needs of this project. The camera comes with the RealSense viewer, as shown in Fig. 3.16. [21]

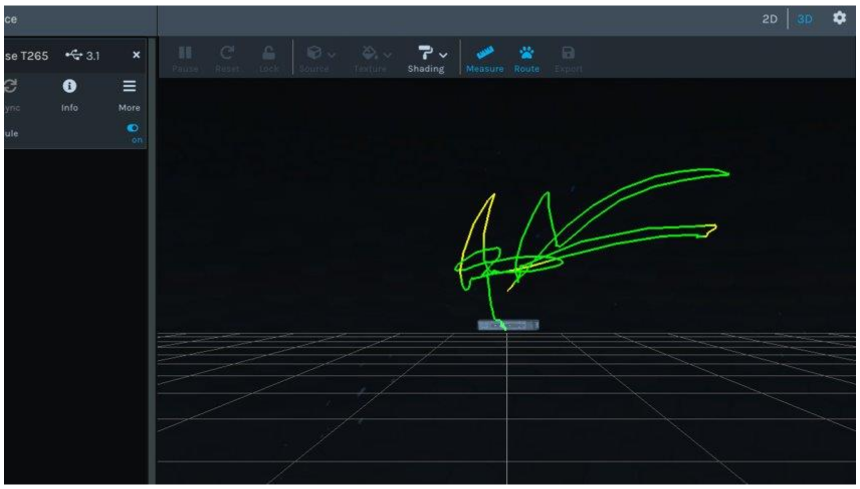


Fig. 3.16. RealSense Viewer 3D View.

As shown above, the camera is able to track its own movements using the provided program. This data is collected by the viewer and presented visually on a digital plane. The color of the path denotes the stability of the movement of the camera with the green being completely stable, and the yellow being slightly unstable. The application also provides the user with a view of what each camera sees as well as the readings of the data being collected by the internal IMU on a set of two-dimensional planes. All this data will be collected and processed using the companion computer on the quadrotor, providing the controller with the position data needed for stable flight. [21]

3.3.1.4.2 Companion Computer Selection

To allow the camera to provide position data to the Teensy Microcontroller, a companion computer needs to be included in the control architecture. There are two main companion computers that work with the Intel Realsense camera and interface, the Jetson Nano and Raspberry Pi. These companion computers are similar in memory, and processing power. They are both Linux based and can run the programs necessary to achieve the functionality needed for successful autonomous flight and data collection through the camera interface. The team settled on using a RaspberryPi3 due to the amount of information that is available on it being used with the camera, as well as it provided us with all the necessary capabilities to run the camera and communicate the collected position data from the camera to the Arduino over the serial monitor.

3.3.1.4.3 Integration within the System

To operate the camera with the companion computer, the software interface must be installed. For the Raspberry Pi, the Intel Realsense SDK is needed to allow for the camera sensor to communicate with the computer. The SDK is open source and supports program development with python and other languages. The download for the SDK also comes with other tools needed to collect data from the camera. Once the interface is set up, the camera is connected via a USB cord, and you are able to collect the position, velocity, acceleration and rotational data for each frame collected by the camera. This data is then sent to the Teensy microcontroller over the serial monitor and then used within the control algorithm to achieve stable flight.

3.3.2 Madgwick Filter

With all the data collected by the sensor array, it must be combined in order to determine the state of the quadrotor. To combine this IMU data, the dRehmFlight controller utilizes a Madgwick filter, also known as a Madgwick Orientation Filter. The filter was created by Sebastian Madgwick in 2009, to combine and filter the data collected by the gyroscope and accelerometer of an IMU. First the filter takes in the three axes of collected data and does an initial filter of the sensor noise. Each of these sets of data are then used to approximate the orientation of the aircraft from the accelerometer and gyroscope separately. These orientation calculations are then put through a fusion algorithm, which provides an estimation of the rate of change of orientation of the aircraft. The filter then calculates the aircraft’s orientation by numerically integrating this function. The structure of this filter has a lower computational load which allows it to be used at lower sampling frequencies and thus it is suited for smaller platforms. This low computational load also makes this filter an ideal choice for use within real time applications. [22]

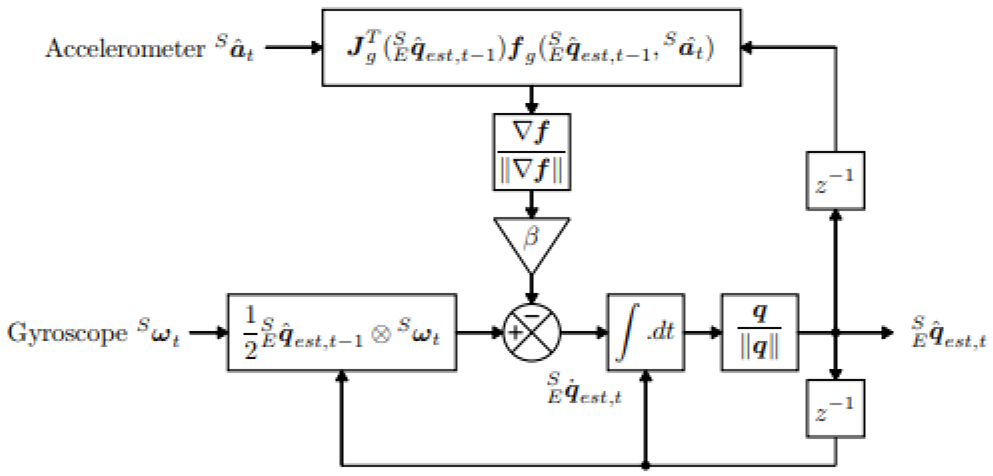

Accelerometer ${}^S\hat{a}_t$
$J_g^T({}^S_E\hat{q}_{est,t-1})f_g({}^S_E\hat{q}_{est,t-1}, {}^S\hat{a}_t)$
$\frac{\nabla f}{\|\nabla f\|}$
β
Gyroscope ${}^S\omega_t$
$\frac{1}{2}{}^S_E\hat{q}_{est,t-1} \otimes {}^S\omega_t$
$\int .dt$
$\frac{q}{\|q\|}$
${}^S_E\hat{q}_{est,t}$
${}^S_E\dot{q}_{est,t}$
z^{-1}
z^{-1}

Fig. 3.17. Block Diagram Depicting the Working of the Madgwick Filter. [22]

3.4 Control Architecture

3.4.1 Control System Overview

The control system for the quadrotor is responsible for maintaining stable flight of the vehicle and executing movements commanded by a pilot or autonomous routine. The overall control architecture is divided into four hierarchical layers. The highest-level part of the system is the motion planner. This is only a feature of the autonomous flight mode and is implemented with the state machine as detailed in Section 3.5.3. The motion planner produces a desired altitude and position for the quadrotor at any given time. One level below this are the altitude and position controllers. The altitude controller is responsible for determining the baseline throttle value for all four motors that is needed to achieve the desired altitude. Similarly, the position controller is responsible for determining the desired pitch and roll angles needed to translate the vehicle to the specified position or to hold it at the current position. In the manual flight mode, the throttle value and angle setpoints are instead commanded by the pilot's instructions from the radio transmitter.

The next level down is the attitude controller. The attitude controller takes the angle setpoints as inputs and generates corrective commands that must be added or subtracted from specific pairs of motors to stabilize the vehicle about the angle setpoints. The outputs from the attitude control are fed into a control mixing block along with the baseline throttle value. This command mixing block determines the desired speed for each of the four motors based upon the input values and the geometry of the quadrotor. Finally, the lowest level of control is implemented by the ESCs which command the motors to spin at the desired speed. This framework explains how a high-level flight path is translated into four motor speeds for control of the vehicle at any given time. Each layer of this control architecture is explained in more detail in the following section. The control architecture diagram shown in Fig. 3.18 illustrates this hierarchical structure more clearly. Fig. 3.19 details the function of the control mixing block and shows the relationship between the controller outputs, electronics speed controllers, and actuators.

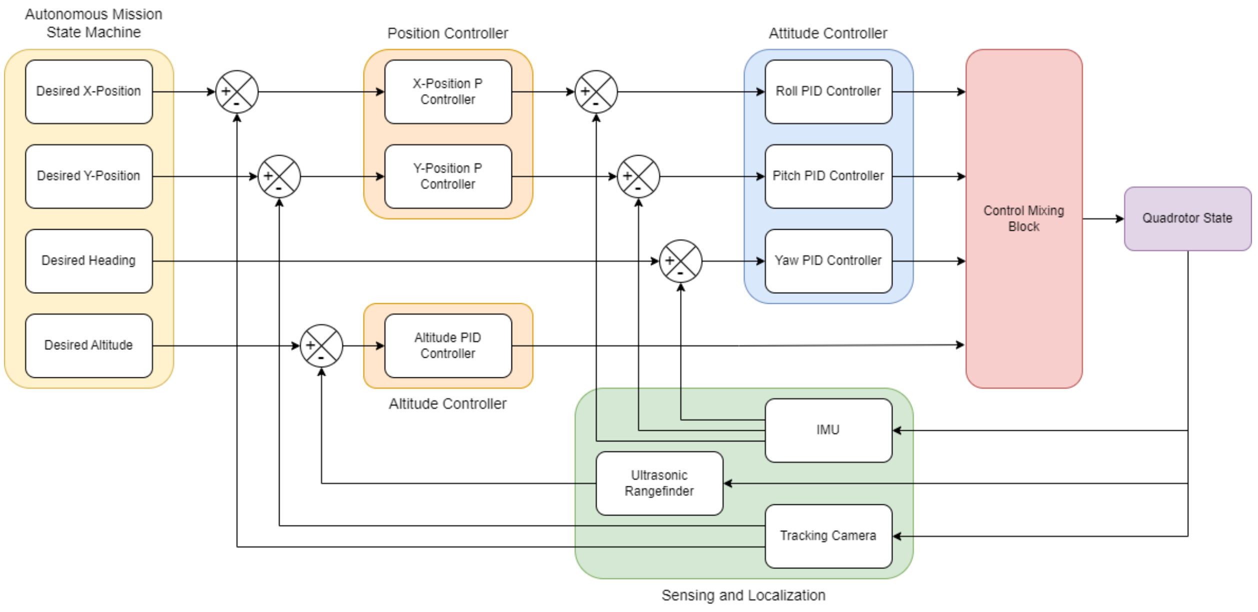


Fig. 3.18. Control Architecture Diagram.

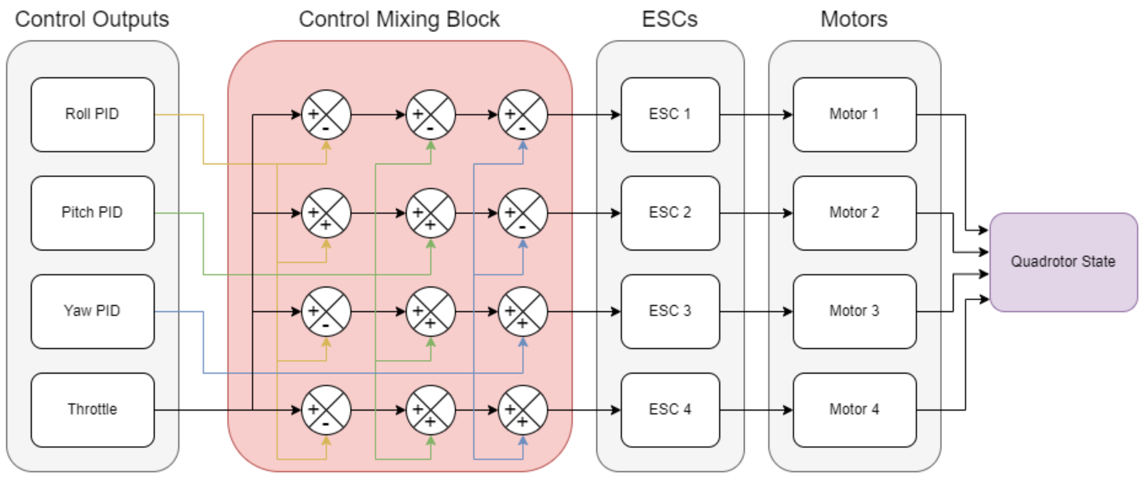


Fig. 3.19. Control Mixing Block Diagram.

3.4.2 Electronic Speed Controllers

Electronics speed controllers (ESCs) are commercially available motor controllers that are commonly used by hobbyists. ESCs are generally used for speed control of three coil brushless DC motors. These devices take an input signal from a microcontroller and use embedded electronics to produce a three-phase signal to be sent to the motor from three output terminals. ESCs require a PWM or oneShot125 signal from the microcontroller to specify the desired angular speed of the motor. The oneShot125 protocol refers to a signal that is set to HIGH for a pulse width ranging from 125ms to 250ms which corresponds to the full range of operating speeds. ESCs may be found in standalone packages, where one device is needed for each motor, or integrated into power distribution boards that feature multiple ESCs in one device. When power is applied to an

ESC, the device performs a distinct beep code upon startup to indicate that it has been armed successfully.

The electronic speed controllers on our vehicle constitute the lowest level of the control architecture. After determining the desired speed for each of the four motors, the calculated motor speeds are scaled to pulse width values to be sent to each ESC over the oneShot125 protocol. The ESC is responsible for maintaining the speed of the motor at the desired set speed. This is done with embedded software that is abstracted away from the end user, who may simply wire the ESC between the microcontroller and motor. However, it is useful to perform a calibration procedure to ensure the best performance from ESCs. The calibration procedure begins by powering on the ESCs while sending the command for maximum throttle. The ESC responds with a characteristic beep code to indicate that it has entered calibration mode. The pilot then throttles the motor command down to zero, which is followed by another beep code from the ESC. This completes the calibration process and ensures that the commands given by the microcontroller are correctly scaled to the full-scale range of motor speeds that can be produced by the ESC.

3.4.3 PID Control

The control architecture of our amphibious quadrotor vehicle makes extensive use of PID controllers. PID control, or proportional-integral-derivative control, is a powerful technique that is widely used in control applications across a variety of industries. PID is a form of closed-loop feedback control that is useful for driving a system to maintain a specified setpoint. The output of the controller is a summation of three terms: the proportional term, integral term, and derivative term. When combined, the proportional, integral, and derivative terms produce a single control output that is given to the plant. The resulting action is taken, the state of the system is measured again, and the calculated error is fed back into the controller in a closed loop.

As its name implies, the proportional term produces a value that is proportional to the error in the system. The error in the system is defined simply as the difference between the measured state and the desired state. This error is then multiplied by the proportional control gain, Kp, to yield the proportional term. The function of the proportional term is to drive the output to the desired value and to bring the error to zero. The proportional term may be expressed as:

$$P = K_p * e(t)$$

Then next term is the integral term. The integral term adds time dependence to the controller by summing the error in the system over time. A running total of the error in the system is multiplied by the integral gain, Ki, to yield the integral term. The integral term is useful in eliminating steady state error from a system. A persistent deviation from the setpoint value will result in an accumulating error sum over time, causing an increase in the integral term to correct for the error. The integral term may be expressed as:

$$I = K_i * \int e(\tau) d\tau$$

The final term is the derivative term. The derivative term also has time dependence and reflects how the error in the system is changing. The rate of change of the error in the system is multiplied by the derivative gain, Kd, to yield the derivative term. The derivative term is useful in reducing overshoot, dampening oscillations, and improving settling time of a system. The ability to respond appropriately to increasing or decreasing error values improves the overall performance of the controller. The derivative term may be expressed as:

$$D = K_d * \frac{d}{dt} e(t)$$

PID control is a commonly adopted technique because it is relatively easy to implement and effective across a range of applications. A diagram illustrating the structure of a complete PID controller is shown in Fig. 3.20. While the structure of a PID controller is straightforward, the tuning of the control gains is not. Many techniques have been proposed to aid in the tuning of PID gains. Some methods rely on the observation of a system performance in the real world coupled with mathematical formulations. Others rely on the development of an accurate model of the system to calculate optimal gain tuning through simulation. Regardless of the chosen method, considerable time spent is often required to experimenting with the tuning of a PID controller to achieve the desired performance. The desired response of a system may also vary depending on the specific application. It is useful to develop an intuitive understanding of the effect that each term has on the performance of the system. The table in Fig. 3.21 presents a summary of the effect that increasing each control gain has on the performance of the overall system.

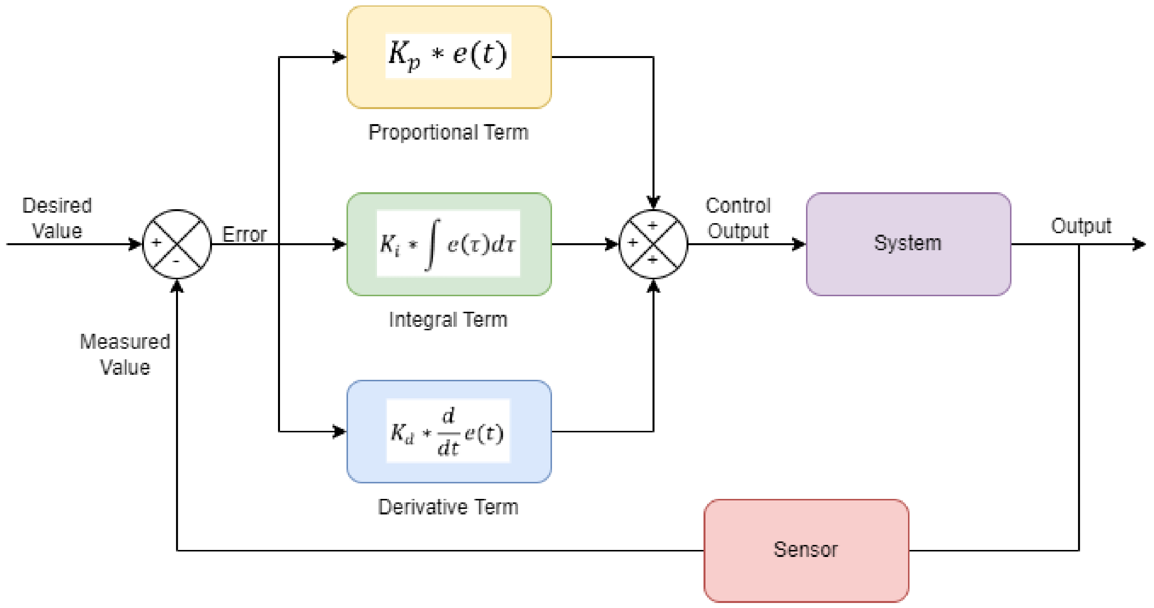


Fig. 3.20. PID Control Block Diagram.

Parameter	***Rise Time***	***Overshoot***	***Settling Time***	***Steady-State Error***	***Stability***
K_P	Decrease	Increase	Small change	Decrease	Degrade
K_I	Decrease	Increase	Increase	Eliminate	Degrade
K_D	Minor change	Decrease	Decrease	No effect	Improve

Fig. 3.21. Effect of Increasing PID Gains on Overall System Performance.

3.4.4 Stabilization and Attitude Control

The second level of the control architecture is the attitude controller. The quadrotor is an intrinsically unstable vehicle and thus requires active stabilization. The purpose of the attitude controller is to stabilize the vehicle about a given pitch angle, roll angle, and yaw rate. Active stabilization of the vehicle's attitude is an essential function of the flight computer even during piloted flight in the MANUAL flight mode. This control is implemented through three independent PID controllers, where one is dedicated to each rotational axis. These PID controllers use the error between the vehicle measured state and desired state as inputs. The measured state refers to the estimated pitch, roll, and yaw angles that are calculated in the flight control loop from IMU data. This is explained in more detail in Section 3.5.2. The desired angles are taken from the pilot's commands in MANUAL mode, and from higher level controllers in AUTO mode. The error in each axis is calculated as the difference between these measured and desired values. The output of these PID controllers is sent to the motor mixing block to command the appropriate response on the system. As a safeguard, a limit is placed on the integral term, such that once it becomes fully saturated, the error will not continue to accumulate with a potentially harmful result.

When the vehicle is not receiving any pitch, roll, or yaw commands, the desired setpoint values default to zero. This means that the vehicle will attempt to keep itself level by maintaining its pitch and roll angles at zero. A vehicle's ability to accomplish this task successfully is imperative for stable flight. The quadrotor's yaw rate is also stabilized to zero to maintain the heading of the vehicle. The gains for these three PID controllers were tuned experimentally to achieve good performance for stabilization. Fortunately, this was not a difficult process because the starting values given in the dRehmFlight VTOL firmware proved to be an excellent starting point. When a pitch or roll command is received, the vehicle stabilizes itself about the commanded angle. The maximum angle that can be commanded is limited to 20 degrees to limit instability. Note that stabilization about a nonzero angle result in a lateral drift due to the misalignment of the thrust vector and gravity vectors. This is a desirable result when commanding the vehicle to move through space, as described in Section 2.3.

3.4.5 Altitude and Position Controllers

The highest-level controllers in the quadrotor's control architecture are responsible for maintaining the vehicle's altitude and position. This functionality is essential for autonomous operation of the vehicle and relies on sensory input from the onboard sensor array. The altitude controller is implemented as another PID controller. The input to the altitude controller is the altitude error, which is defined as the difference between the desired altitude setpoint and the measured altitude. The measured altitude is determined by the most recent reading from the ultrasonic rangefinder. Just as with the PID controllers for attitude control, the altitude controller implements an integral limit to prevent unsafe error buildup which could be damaging to the system. This PID controller differs from the attitude controllers with the inclusion of an additional feedforward term that is input to this controller. This term corresponds to a baseline hover throttle that was determined experimentally. The controller then has the effect of increasing or decreasing this value appropriately in response to the altitude error. The control gains were tuned experimentally as detailed in Section 3.6.3. The output from the altitude controller is sent to the motor mixing block to set the baseline throttle level for all four motors.

While the altitude controller maintains the position of the quadrotors along the vertical axis, the position controller is responsible for holding the craft's position in the longitudinal and lateral axes. Rather than a PID controller, position control is simply implemented with two proportional controllers. The position error in the X and Y axes is calculated as the difference between the desired position and measured position of the quadrotor. The desired position is specified in the flight control code as a chosen point on a grid, where the origin is located at the vehicle's initial position upon startup. The measured position of the quadrotor is determined by the pose estimate given by the RealSense tracking camera. A more detailed description of how this works is given in Section 3.3.1.4. The position error is multiplied by a constant proportional gain to produce a control output. The proportional gain was tuned experimentally to yield a reasonable balance between response time and settling time. The output values for the X and Y axes are used to specify the desired pitch and roll angles of the vehicle respectively. These are the values that are then passed to the attitude control to achieve stabilization about the commanded angles. This causes the quadrotor to travel to and then maintain the specified position. The combination of the altitude controller and position controller allows for the vehicle to be autonomously commanded to any position in three-dimensional space. This is the most fundamental building block for full autonomy, allowing for the flight computer to command reliable point to point travel without any input from a pilot. This capability can be extended with motion planning techniques to direct the vehicle along complex flight paths in a fully autonomous manner.

3.5 Flight Control Software

3.5.1 Overview and Development Environment

The custom flight control software developed by our team is an extension of the dRehmFlight VTOL flight controller published by Nicholas Rehm. Using dRehmFlight as a starting point, the code was modified to meet the unique requirements of this project. Many features provided for the support of different hardware setups were removed to streamline the code for performance and readability. Additionally, many new features were added to support amphibious AUV project, including interfaces with navigation sensors, logic to switch between flight modes, autonomous altitude and position controllers, and a state machine for execution of an autonomous mission.

Software development was conducted in C++ using Visual Studio Code with the PlatformIO extension for embedded microcontroller programming. Collaboration and version control was instituted with a GitHub repository which contains the flight code and other important documents. The Teensyduino driver was utilized to upload code to the Teensy 4.0 microcontroller. The main flight code is contained within the file “Amphibious_AUV.ino.” Additionally, several supporting libraries are referenced by the main file which contain sensor drivers, radio configuration functions, and support for communication protocols. All the files needed to compile and run the flight software are included in the GitHub repository.

3.5.2 Flight Computer Architecture

The flight control software is divided into several sections. The definitions section at the beginning of the file contains important references to other files, declarations of objects and global variables, and enumerations used by the state machine. This section also includes configurable parameters for setting sensor biases, tuning control gains, and setting mission parameters. The next section of the code is the setup function. This function is always called upon startup of the flight controller. The setup function contains calls to functions which calibrate and initialize sensors, perform radio receiver setup, arm the electronic speed controllers, and set the initial conditions of the state machine. The onboard LED on the Teensy microcontroller is turned on while the setup function is running as an indicator to the pilot.

Following the initialization function is the main flight control loop. This loop runs continuously after the setup function is executed once. The main control loop runs several critical functions which will be explained in more detail below. A flowchart depicting the sequence of actions taken each iteration through the loop is given in Fig. 3.22. The control loop is designed to be run at a frequency of 2000 Hz. It is critical to maintain consistent timing due to the time sensitive nature of the control loops which influence the tuning of gain values. This is challenging because the computation time is not constant due to variations between different iterations of the loop. To account for this, a special function is called at the end of the loop to maintain the desired loop frequency. This function records the current system time at the end of the loop with the system time saved at the beginning of the loop to determine how much time has elapsed. The function then institutes a software delay with the use of internal timers to delay the start of the next loop until the correct amount of time has passed. Additionally, the microcontroller’s onboard LED is blinked every 1.5 seconds while the main loop is running to indicate to the pilot that the program is executing the flight control loop.

Several critical functions are performed in sequence during each iteration of the main flight control loop. First, the program reads raw accelerometer and gyroscope values from the inertial measurement unit. These values are subject to a low-pass filter to remove high frequency noise from the measurements. Next, the sensor values are given to the Madgwick filter to update the vehicle’s state estimate. This step involves a C++ implementation of the algorithm described in Section 3.3.2. to fuse the sensor values and combine them with the current state estimate to produce a new state estimate. This outputs the current best estimate of the quadrotor’s roll, pitch, and yaw angles. With the measured state of the vehicle known, the value of radio channel 6 is checked to determine which flight mode is currently selected. The flight computer then determines the desired state of the vehicle depending on the flight mode. In MANUAL mode, the desired state is determined from the pilot’s commands which are received from the radio transmitter. In AUTO mode, the desired state is determined from the high-level control and path planning algorithms which guide the craft through its mission. More information about the different flight modes is provided in Section 3.5.3.

Now that the flight computer knows the current state and the desired state of the vehicle, the system must respond to achieve the desired state. At this point, the attitude stabilization controller is called to generate values for stabilization about the desired pitch and roll angles and desired yaw rate. These values are then sent to a control mixer, where they are added or subtracted to the baseline throttle level for each motor in order to achieve the desired response. A detailed explanation of the dynamics from which this mixing is derived is presented in Section 2.3. The control mixer outputs commands for each of the four motors between 0 and 1. The controller checks then checks the value of radio channel 5 to monitor the state of the throttle cut switch. The motor values are all set to 0 in the event that the throttle cut switch is activated. The resulting values from 0 to 1 are then scaled to pulse width values between 125ms and 250ms to be sent to the ESCs over OneShot125 protocol to appropriately command the motors to the desired speeds. Finally, the system pulls the currently available radio commands on channels 1 through 6 to be used in the next iteration of the loop.

Additionally, there are a few other actions that are not executed every loop. These include reading the distance value from the ultrasonic sensor and reading the pressure value from the depth sensor. These actions are not performed during every iteration of the loop because they cannot be completed quickly enough. The time required for a pulse emitted from the rangefinder to reflect off an object and return to the detector is longer than an iteration of the loop. Therefore, in order to collect reliable altitude data, the sensor must be polled at a reduced frequency. A similar technique is employed with the depth sensor, because a single conversion to read from the sensor can take up to 40ms which exceeds the duration of one loop iteration. Decreasing the loop frequency to accommodate for these sensors is undesirable because it would hurt the performance of the controller and decrease the stability of the vehicle. For this reason, a counter was implemented to poll the sensors once every 200 iterations of the loop. This allows for the flight control loop to continue running at a fast speed of 2000 Hz while only polling the sensors at 10 Hz. This has no substantial impact on performance as this rate is perfectly adequate to obtain good altitude and depth data.

After the main control loop, the final section of the code contains all the supporting functions that are called by the setup function and main loop. These functions contain concise blocks of code that are specific to the particular tasks referenced above. Beyond the critical functionality outlined in this section, some additional functions are maintained for debugging purposes. Most notably, there are a wide array of functions which print various values to the Serial monitor for use in troubleshooting when the vehicle does not perform as expected. These include functions to print sensor values, controller outputs, motor commands, and other values as needed. During nominal operation, these functions are all commented out in the code to improve performance as printing values takes considerable computation time. Functions for calculating sensor biases and calibrating the ESCs are also included for occasional use in setup and troubleshooting.

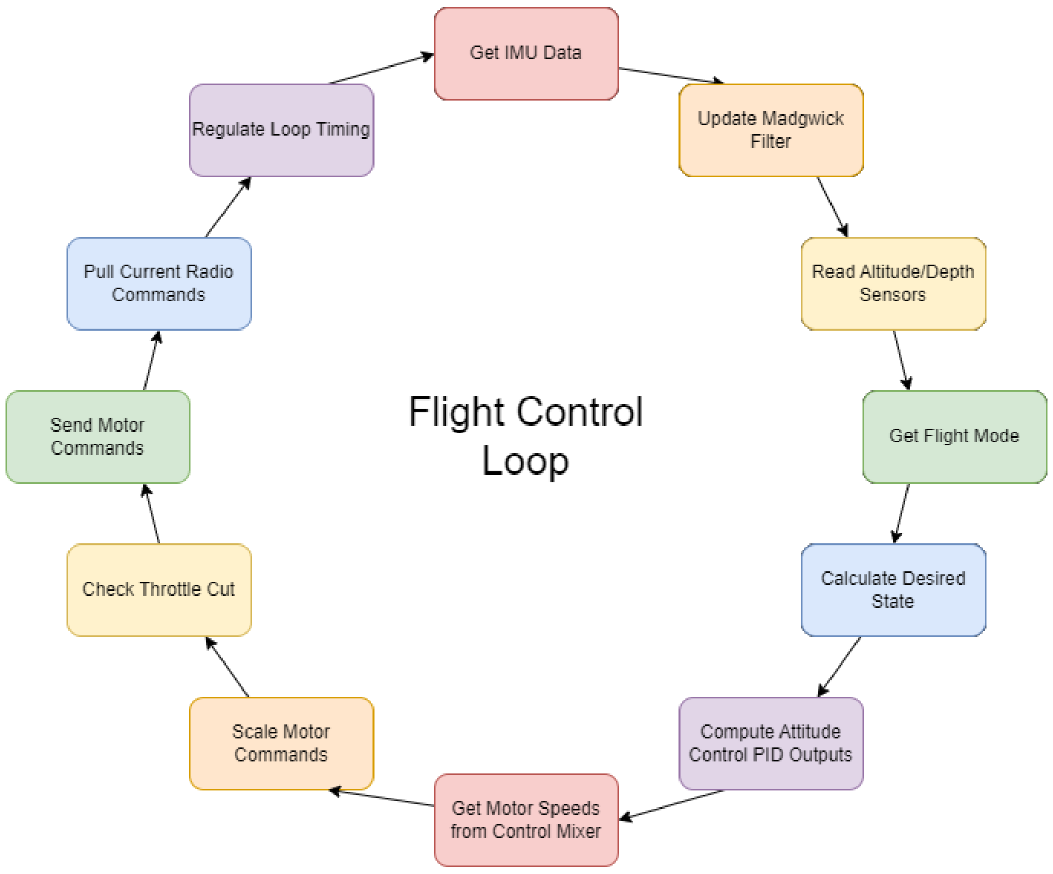


Fig. 3.22. Main Flight Control Loop Flow Chart.

3.5.3 State Machine

A state machine was implemented to control the quadrotor as it switches between different modes of operation. A hierarchical structure was chosen to best accomplish this task, with an inner layer and an outer layer. The outer layer is used to switch between different flight modes. The available flight modes are MANUAL, HOVER, and AUTO. Manual mode allows for direct control of the quadcopter with the radio transmitter for piloted flight. Hover mode is used for instructing the vehicle to hover at a fixed point in space without controller input. Finally, auto mode is used to execute a pre-programmed autonomous mission with the quadrotor. Switching between flight modes is accomplished with an auxiliary switch on the radio transmitter. The quadrotor always begins in MANUAL mode upon startup. A pilot may then toggle the multi-position auxiliary switch to change the operating mode to HOVER or AUTO as desired.

The inner layer refers to a secondary state machine within each of the defined flight modes. These states are used to transition between various functions within each operating mode. All flight modes have a STARTUP state that is used to reset controller values and perform any necessary initialization. In manual mode, a second state called NORMAL allows for piloted control of the vehicle. In hover motor, the HOVER state keeps the vehicle stable at a fixed set point in space, until a radio signal transitions the vehicle into the LANDING state, at which point the vehicle slowly descends to the ground for a landing. The AUTO mode features many states to execute the vehicle's autonomous mission. A unique state exists for each movement or action taken by the quadrotor, such that these steps can be completely sequentially to carry out the mission.

The state machine is implemented in C++ with the use of nested switch case statements. One enumeration was written to list the various selectable flight modes for the vehicle. When the main flight control loop reaches the first switch statement, the program executes the code for a particular flight mode based upon the value of the flight mode variable which is set by the position of the auxiliary switch on the transmitter. Within each case, a second switch case statement is used to transition between states within the specified flight mode. An enumeration for each flight mode lists the possible states within that mode. The current state within each flight mode is determined by logic that is specific to the current state. The STARTUP state is the initial state whenever a flight mode switch is executed. After this, the state transition logic is specific to the different flight modes as described in the preceding paragraph.

3.6 Test Quadrotor Vehicle

3.6.1 Test Vehicle Construction

The Navigation and Controls sub-team designed and built an early prototype quadrotor to begin software and control system development in advance of the completion of the final vehicle. The test vehicle was constructed from PLA with a fully 3D printed frame which was acquired from the online 3D modeling community Thingiverse. The frame consists of a top plate, bottom plate, and four arms which serve as mounting fixtures for the motors and landing legs for the craft. The vehicle was assembled with standard M5 bolts and hex nuts. The team chose to use an existing frame design to save on design effort and to have confidence that a previously tested design would have good stability for flight. The fully assembled test vehicle is shown in Fig. 3.23 with the RealSense camera facing front. A top view is shown in Fig. 3.24 and a bottom view is shown in Fig. 3.25.

The test quadrotor was then outfitted with the custom electronics assembly described in Section 3.2. The IMU was placed on the center of the top plate, close to the vehicle's center of mass to optimize the sensor's performance. The sensor was mounting atop a square of soft foam to provide dampening from the vibration of the frame induced by the motors. This isolation from vibration is important to minimize noise in the sensor readings. Excessive vibration results in poor IMU data that is unusable for achieving stable flight. The prototyping PCB with the Teensy 4.0 microcontroller was mounted behind the IMU on the top plate. The power distribution board and BEC were mounted to the bottom plate. The four motors were mounted on the dedicated fixtures on the ends of each of the arms, with the ESCs mounted underneath the arms. The 3S LiPo battery used to power the system was mounted underneath the craft and secured with Velcro straps. The ultrasonic rangefinder and depth sensor were mounted on a 3D printed bracket bolted to the bottom plate at the front of the craft. The sensor mount is shown in Fig. 3.26.

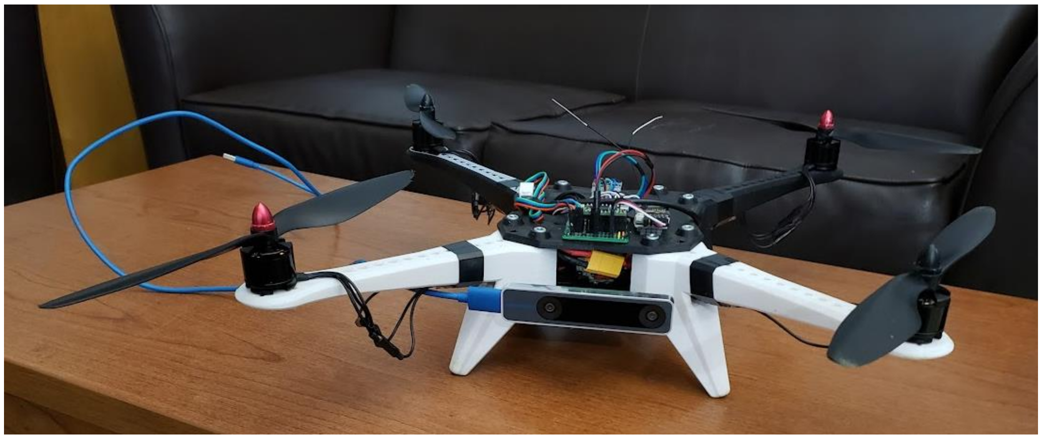


Fig. 3.23. Fully Assembled Test Quadrotor Vehicle.

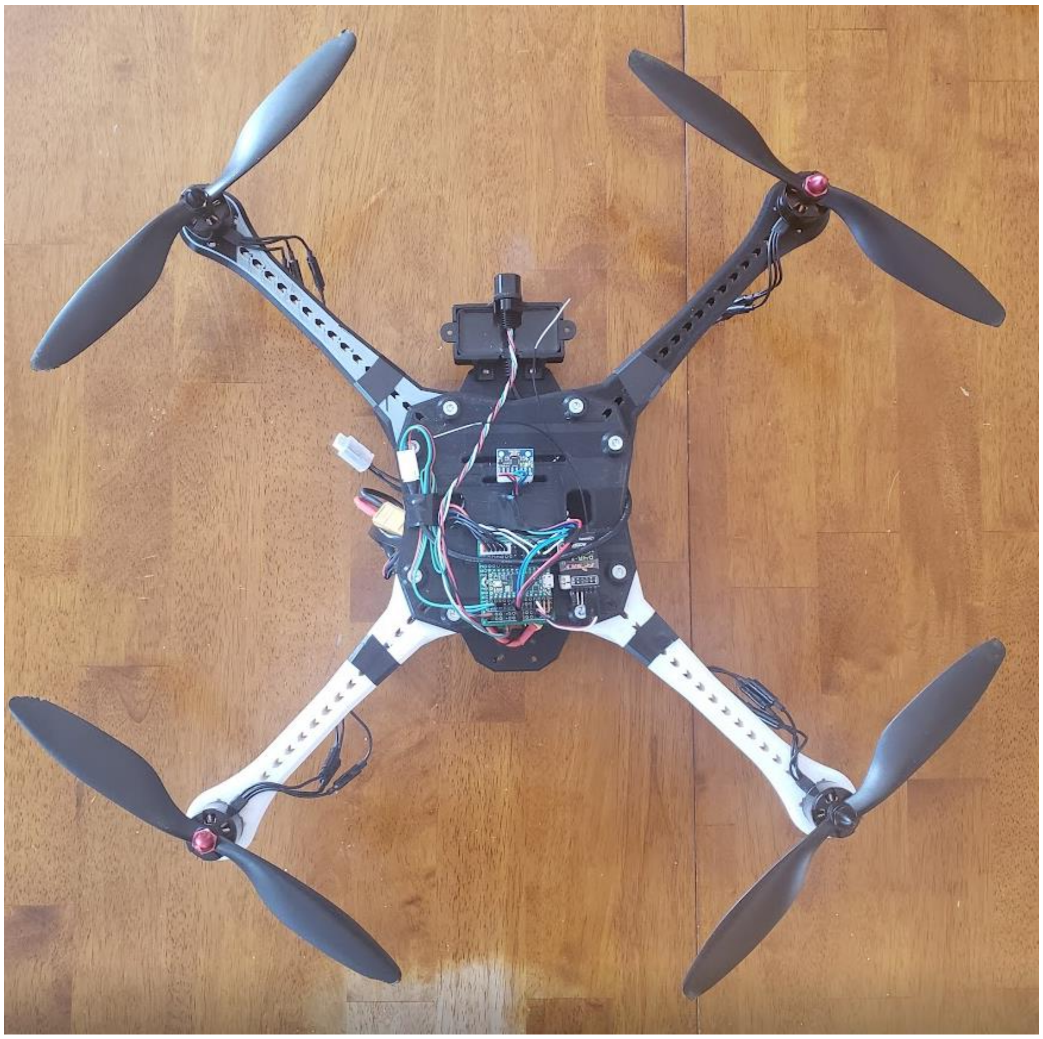


Fig. 3.24. Top View of Test Quadrotor.

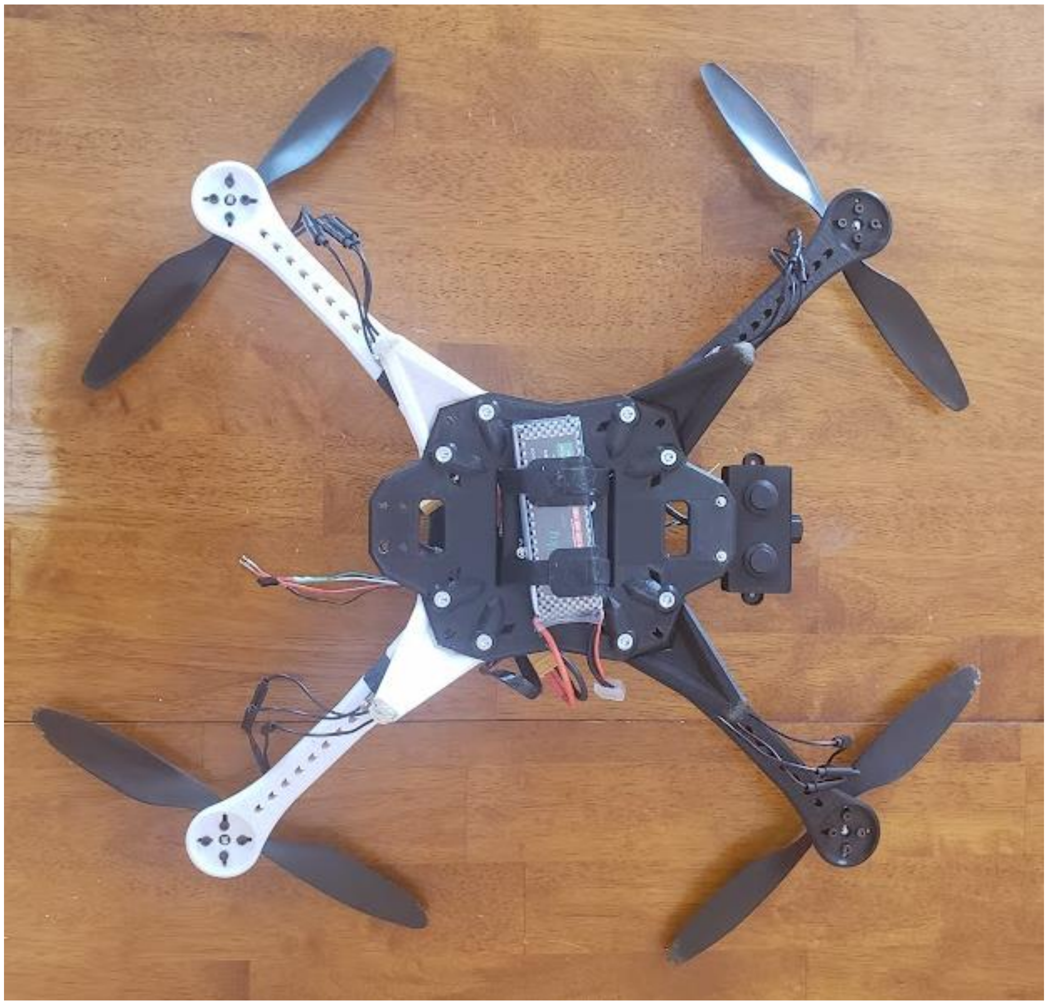


Fig. 3.25. Bottom View of Test Quadrotor.

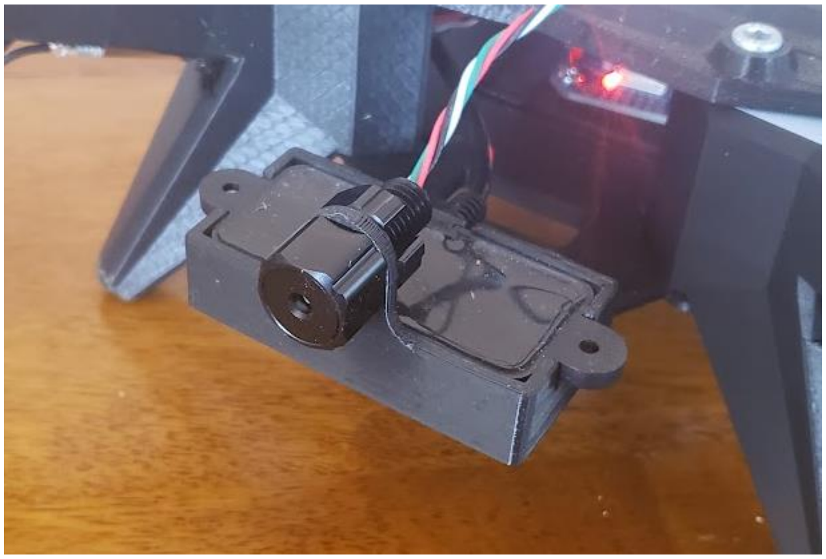


Fig. 3.26. Test Quadrotor Sensor Mount.

3.6.2 Preliminary Calibration and Ground Testing

Once the test quadrotor was fully assembled, a series of ground tests and calibration steps were performed before attempting the first flight. Electrical testing of the system was performed with the use of digital multi-meter. First, continuity checks were performed to ensure that the correct connections were made successfully and that there was no risk of causing a short circuit when power was applied. The battery was then connected, and the system was probed to ensure that power distribution to each of the components was as expected. Next, a simple test program was uploaded to the Teensy microcontroller to verify that it was working as expected. After completing these electrical checks, the next step was to test the various sensors and peripheral devices.

Each of the sensors were first tested independently with the use of dedicated driver libraries. The acceleration and angular rate values from the IMU were visualized in the Arduino Serial Plotter to verify the expected performance. Plots of filtered values from the accelerometer and rate gyroscope are shown in Fig. 3.27 and Fig. 3.28 respectively. Similarly, distance values from the ultrasonic sensor and depth readings from the depth sensor were compared against known values to verify their performance. A printout of altitude measurements from the ultrasonic rangefinder is shown in Fig. 3.29. Having demonstrated that all the sensors work as expected, the sensor drivers were integrated into the main flight code. At this point, a calibration procedure was performed with the IMU to obtain better performance. The quadrotor was placed at rest on a flat surface while sensor readings were recorded for a period of several seconds. The recorded values were averaged and subtracted from expected values to estimate the static bias for every axis of the sensor. These bias values were then stored in the code to be subtracted from future measurements for more accurate readings.

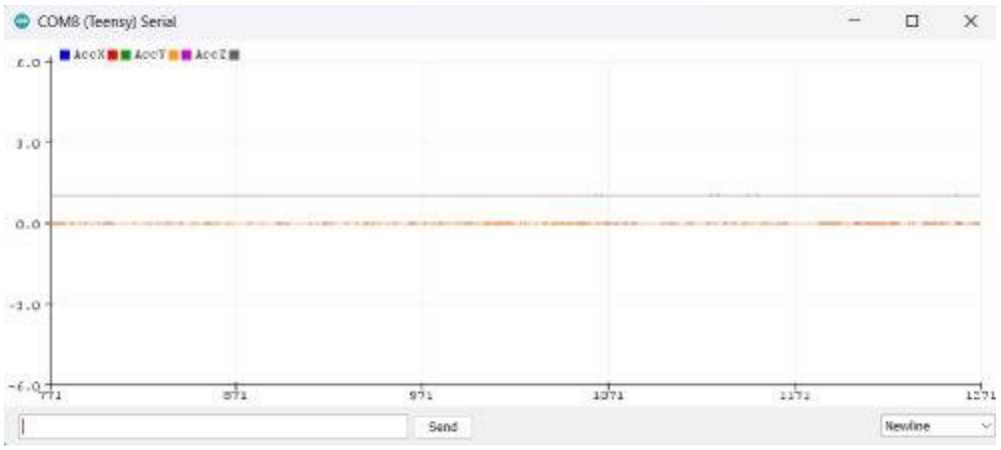


Fig. 3.27. Plot of Filtered Accelerometer Readings.

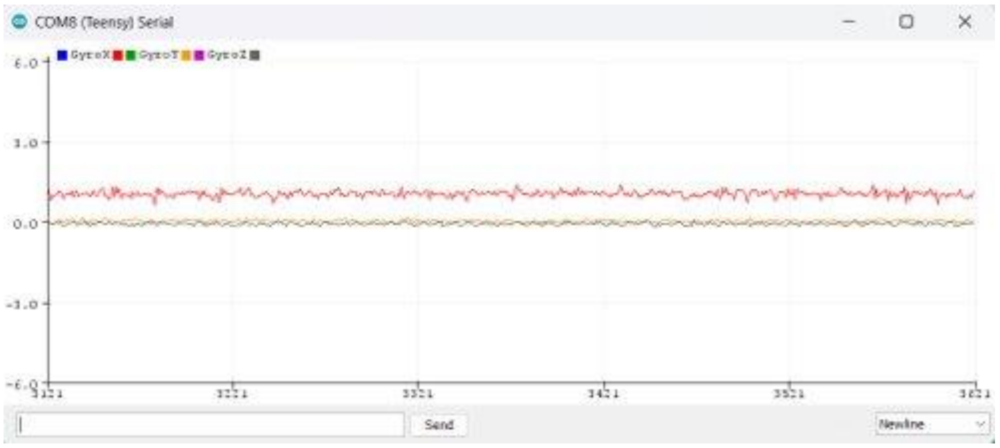
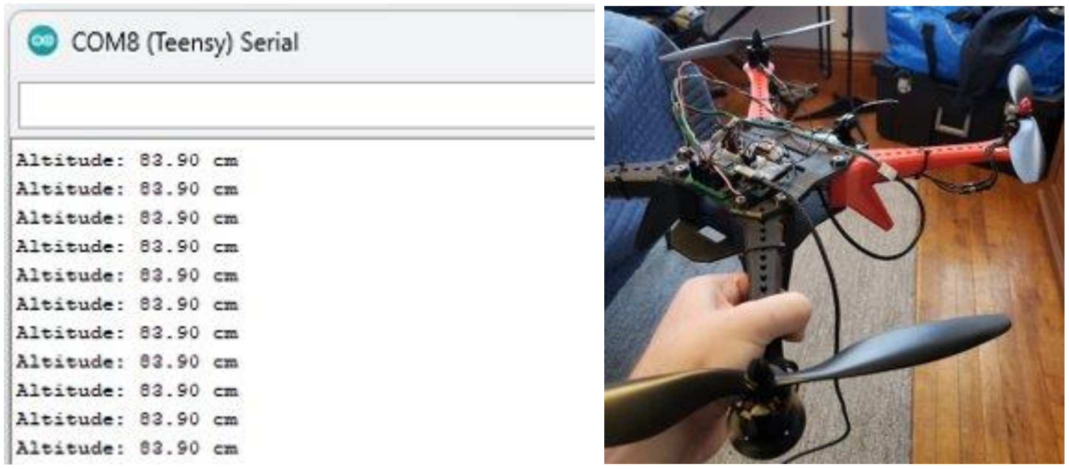


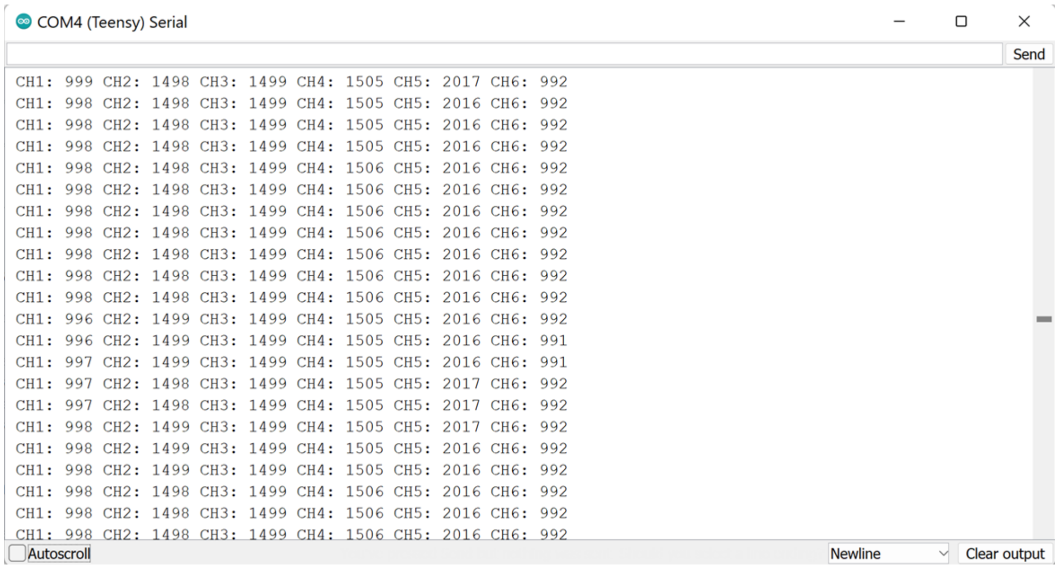
Fig. 3.28. Plot of Filtered Gyroscope Readings.



```
Altitude: 83.90 cm
Altitude: 83.90 cm
Altitude: 83.90 cm
Altitude: 83.90 cm
Altitude: 83.90 cm
Altitude: 83.90 cm
Altitude: 83.90 cm
Altitude: 83.90 cm
Altitude: 83.90 cm
Altitude: 83.90 cm
Altitude: 83.90 cm
```

Fig. 3.29. Printout of Real-Time Altitude Measurements.

With the sensors working, the next step was to configure radio communication between the quadrotor and the transmitter. A new configuration was created on the transmitter to map the joysticks and switches to the desired radio channels. Channels 1 through 4 were mapped to the joysticks to control throttle, pitch, roll, and yaw. The throttle cut switch was mapped to Channel 5 and the flight mode switch was mapped to Channel 6. Once the transmitter had been configured properly, the flight code was uploaded to the quadrotor. The values received from the radio channels were printed to the Serial Monitor in a loop for debugging purposes. The expected range for each radio channel is 1000 to 2000. Every joystick and switch were moved through its full range of motion to verify that the mapping had been performed correctly. Further calibration was then performed on the transmitter to set the full-scale range of every axis from 1000 to 2000, centered about the value 1500 in the neutral position. This is important to guarantee that the pilot's inputs to the transmitter are interpreted correctly by the flight computer. A printout of the values received from the radio transmitter is shown in Fig. 3.30.



```
CH1: 999 CH2: 1498 CH3: 1499 CH4: 1505 CH5: 2017 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1505 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1505 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1505 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1506 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1506 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1506 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1506 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1506 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1506 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1506 CH5: 2016 CH6: 992
CH1: 996 CH2: 1499 CH3: 1499 CH4: 1505 CH5: 2016 CH6: 992
CH1: 996 CH2: 1499 CH3: 1499 CH4: 1505 CH5: 2016 CH6: 991
CH1: 997 CH2: 1499 CH3: 1499 CH4: 1505 CH5: 2016 CH6: 991
CH1: 997 CH2: 1498 CH3: 1499 CH4: 1505 CH5: 2017 CH6: 992
CH1: 997 CH2: 1498 CH3: 1499 CH4: 1505 CH5: 2017 CH6: 992
CH1: 998 CH2: 1499 CH3: 1499 CH4: 1505 CH5: 2017 CH6: 992
CH1: 998 CH2: 1499 CH3: 1499 CH4: 1505 CH5: 2016 CH6: 992
CH1: 998 CH2: 1499 CH3: 1499 CH4: 1505 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1506 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1506 CH5: 2016 CH6: 992
CH1: 998 CH2: 1498 CH3: 1499 CH4: 1506 CH5: 2016 CH6: 992
```

Fig. 3.30. Printout of Values from Radio Transmitter.

After configuring the radio communications, the electronic speed controllers and motors were also tested. The propellers were removed from the motors for safety and to prevent unwanted movement of the vehicle before an actual flight was attempted. The battery was connected to the power distribution board, powering on the ESCs as indicated by the characteristic beep codes. First, a calibration routine was performed by setting the motors to full throttle, waiting for the beep signal from the ESCs, then throttling down. This is done to properly map the throttle input to the full-scale range of motor speeds that the ESC can output. Next, the throttle was slowly raised to check that all motors were spinning properly. The direction of spin was observed for each motor, and two leads were switched in the case that the direction needed to be reversed. The motors were slowly brought up to full throttle while evaluating their performance, and then throttled down to zero. With all the hardware functioning as expected, the final ground tests could be conducted to test the performance of the controller before attempting the first flight.

The Madgwick filter and state estimation code was tested with the motors off. While running the main control loop on the quadrotor, the calculated pitch, roll and yaw angles were printed to the Serial Monitor. A plot of these values is shown in Fig. 3.31. The quadrotor was physically moved and rotated about its axes while monitoring the plot to verify that the vehicle's attitude was measured correctly. After achieving satisfactory results, the quadrotor was placed on a level surface and the motors were powered on. Next, the motor mixing commands were tested to ensure that the vehicle would respond correctly to commands from the pilot. The transmitter was used to instruct the vehicle to rotate about each of its axes, while observing the behavior of the motors. For example, a command to pitch the craft downward should result in the rear motors throttling up to achieve the correct response. The vehicle's behavior was evaluated systematically for every command to validate the performance. Following this demonstration, the vehicle was lifted and held in the air to check the performance of the stabilization controllers. The vehicle was tilted in each direction to verify that the correct motors would throttle up to restore equilibrium.

Once this test was performed to satisfactory completion, ground testing for the vehicle was complete.

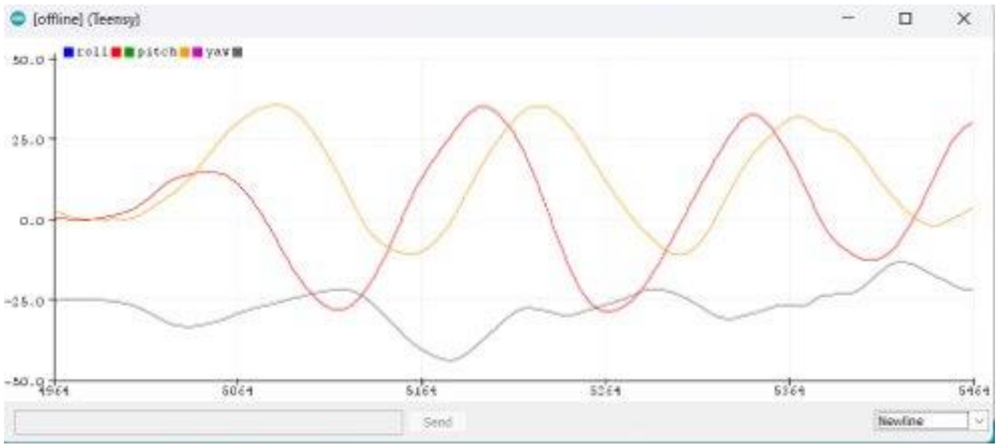


Fig. 3.31. Plot of Pitch, Roll, and Yaw Angles.

3.6.3 Flight Testing

The main goals for the quadrotor were to achieve stable aerial flight under manual control, to achieve autonomous altitude hold capability, to achieve autonomous position hold capability with minimal drift, and finally to achieve point to point autonomous navigation. The first objective was simply to get the vehicle to fly while piloting manually. After performing qualification of the vehicle through calibration and ground testing as described in the preceding section, the quadrotor was powered up for the first aerial flight attempt. Initial efforts were unsuccessful due to a variety of factors including improperly tightened hardware, unbalanced propeller blades, and poor IMU calibration. These issues were troubleshooting in a systematic manner however and resolved, leading to the first successful test flight of the quadrotor. It was found that applying trim corrections on the radio transmitted improved the ease of flying under manual control. Several team members took control of the quadrotor during the first successful test flights and video was recorded to document the milestone. This test was important in verifying the flightworthiness of the vehicle as well as for validating the flight computer's ability to perform state estimation and active stabilization. A still image from the first successful flight is shown in Fig. 3.32.

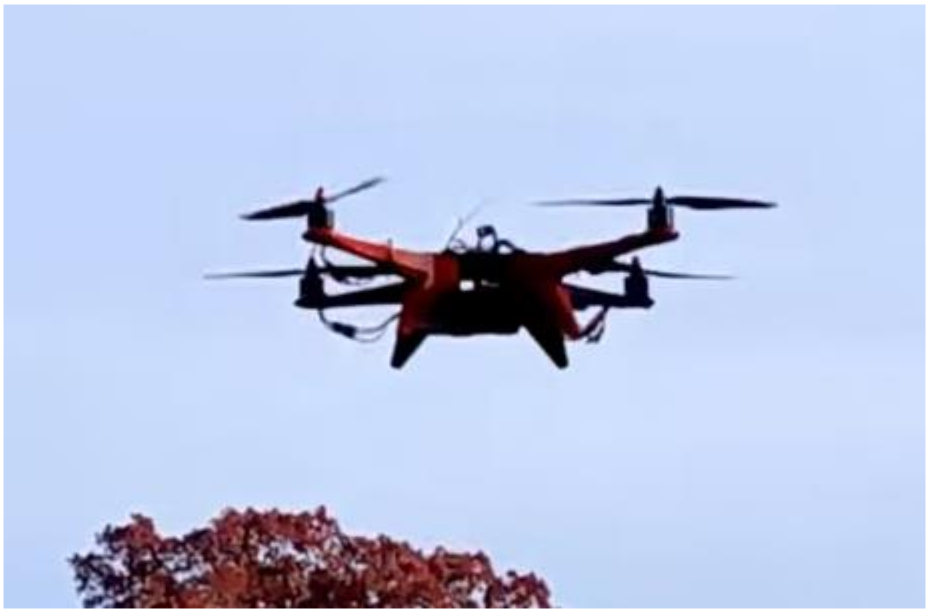


Fig. 3.32. First Successful Flight of Test Quadrotor.

The next major milestone to achieve was an autonomous hover of the vehicle by implementing an altitude controller. While the initial code for the autonomous hover maneuver was written quickly, this proved to be a very difficult task. One major obstacle was the frailty of the test vehicle. The 3D printed arms constructed from PLA were easily broken in crashes resulting from early attempts of autonomous control, which resulted in large delays and setbacks. Another major issue was the discovery of misaligned motor shafts causing imbalances and excessive vibration which contributed too much noise to the inertial sensor readings. This made it impossible for the vehicle to stabilize itself properly and to get off the ground. After troubleshooting these issues to get the vehicle flying, the tuning of the PID control gains was a long process requiring many iterations.

The team documented the control gains used during each test and made calculated changes to the values in order to perform experimental tuning in a systematic matter. The performance of the controller gradually improved over time as the design was improved and more test flights could be performed. Several weeks of extensive testing and experimenting were undertaken to produce satisfactory results. However, the team was ultimately successful in tuning the control to achieve autonomous hovering of the quadrotor at a predefined altitude with our custom flight controller. Note that at this point, the camera was not yet integrated so the vehicle has no knowledge of its position. This means that the vehicle was still susceptible to drift, and while the throttle command was controlled entirely by the flight computer, the pilot was still responsible for controlling the vehicle’s longitudinal and lateral translation. A video was taken to document this important milestone in the development of the custom autopilot software. A still image from this test is shown in Fig. 3.33.

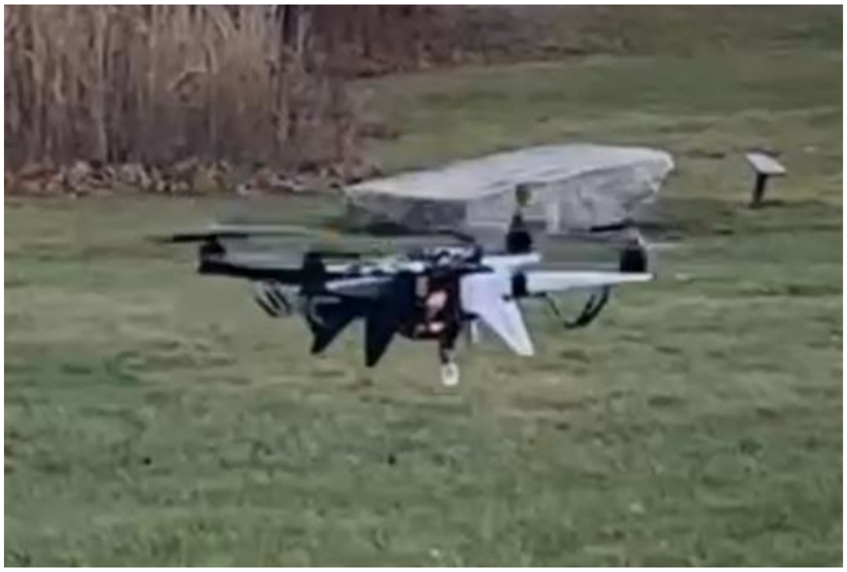


Fig. 3.33. Autonomous Hover of Test Quadrotor.

3.7 Quadrotor Body Design

In this project the drone body serves two primary purposes: structural and waterproofing. While existing commercial options do exist for waterproof drones, our team decided early within the design process to create a custom body. This decision was made due to concerns over waterproofing and size. Any commercial waterproof drones were likely to require significant modification anyhow to accommodate the larger electronics unique to this project – specifically those utilized in control and localization. Instead, a standard quadcopter frame was selected to be utilized alongside additional custom structure and waterproofing. This option was chosen because of concerns about the structural integrity of a custom frame and issues past MQPs faced in balancing a quadcopter frame. The chosen frame is the S500 Quadcopter Frame Kit, a common frame for our size drone. Made of plastic and carbon fiber, the frame had the advantages of being waterproof, light, strong and cheap. In order to fit with the rest of the drone body some components had to be minorly modified while others were removed from the frame entirely. We specifically utilize the 4 arms, bottom plate, and hardware. The top plate, landing legs, and camera mounts were all removed.

To waterproof the drone research illustrated two potential solutions. The first option is to individually waterproof the electronics, typically through a waterproof coating. This method has the advantage of being extremely lightweight, small in size, as well as reliable given a good application. In exchange, assembly is more difficult and once sealed the electronics may be significant work to modify. Due to our uncertainty of electronics choices and likelihood of iteration this method was ruled out. Instead, the option we ended up selecting was to construct a waterproof enclosure. This design decision meant a generally heavier and larger drone but allows for more freedom in electronics choices and changes, as well as the drone body to be developed in parallel alongside the electronic systems. The waterproof enclosure also allowed for more design freedom regarding our other primary structural goal, buoyancy control.

3.7.1 Material Selection

Having decided to create a custom waterproof enclosure the next step was to determine the materials and how it would likely be manufactured. These were factors that could greatly influence further design decisions, particularly due to our limited resources. Some possibilities for materials that we looked at were using a pre-existing waterproof container, a waterproof styrofoam, or even using fiberglass as is common in many boats. Instead, we decided to design for 3D printing. 3D printed parts have a significant advantage over other manufacturing methods since they can be produced extremely quickly, with relatively little work, and cheaply. The lower tolerances and strength of 3D printed parts were determined to be relatively negligible downsides.

The primary issue with typical 3D printed parts was that they are not waterproof. For common filaments – like PLA and ABS – tiny holes are left between the layers during the printing process. Water can seep through these holes and through the infill mesh found on the inside of 3D printed parts, letting parts become waterlogged. This issue is typically addressed by using a waterproof type of plastic with stronger layer adhesion, or by re-melting the outside of parts using a solvent. Our group decided to utilize a common waterproof filament type: PET-G, a strong material most commonly found in plastic bottles. As an extra measure of precaution, it was decided that all external 3D printed parts should be printed with 100% infill, or completely solid. This meant that much of the design had to be thin-walled shells as any solid shape would be extremely heavy.

Other materials utilized in the quadrotor construction include epoxy, hot glue, and acrylic. A small acrylic window was designed in the enclosure for the tracking camera. Acrylic was used as it was the easiest to acquire clear plastic. For our epoxy we specifically used the Loctite brand five-minute instant mix. This was chosen as it was cheap, easily obtainable, and easy to apply. The epoxy was used in permanently attaching various 3D printed components together. Hot glue was again utilized due to its ease of application and acquisition, specifically hot glue was used for attaching parts together non-permanently, in the case that something needs to be switched out, as well as in waterproofing.

3.7.2 Enclosure

The enclosure is the main body of the quadrotor and houses all the primary electronics such as the electronic speed controllers, the battery and the camera. It was designed based on three primary constraints: accessibility, water tightness, and ballast compatibility.

Accessibility means specifically the ability to remove all internal components from the enclosure without any permanent disassembly. Furthermore, accessibility also refers to being able to relatively easily swap batteries. These constraints were chosen so that the electronics could be easily swapable if issues arise, and so that the battery could be switched out during extended testing sessions. To allow for this accessibility a separate box was designed as a mounting surface for all electronics, excluding the battery and camera. Normally held in by detents, this box can slide in and out the top of the enclosure. The battery can be removed in a similar way, sliding out the top. As such the general shape of the enclosure is like an open cup, with flat sides and an open top. The general dimensions of the enclosure were determined based on the electronics sizes as well as the frame hole patterns. The exception to this description is the camera area. Set in front of the battery, this area has an acrylic window and large reliefs to allow removal of the tracking camera. These were required as the camera needed to be mounted horizontally but was far longer than the width of the enclosure, and by extension opening.

The second primary design consideration was water tightness. As discussed in section 3.7.1, 3D printing allowed for a lot of flexibility, but required a generally thin-walled design. This factor, combined with the large clearance holes required for accessibility, meant that a significantly sized hatch had to be implemented into the enclosure. In order to seal this hatch, we decided to create a large flange. The flange has a top plate, large gasket, and bottom plate which are all sandwiched together with numerous bolts around the perimeter. This design was chosen due to the relatively easier access to attachment hardware – since no threads or enclosed nuts would be required in the 3D prints; and relaxed tolerance requirements. Other designs such as a more typical o ring or overlaps required more precise tolerances to be waterproof than a flange which could simply be sanded flat. The primary concern with the flange design was weakness and deformation. Overtightening of the bolts and the elastic nature of the gasket can mean that in between the bolt pattern the flange may buckle apart allowing gaps for water to seep through. This potential issue was addressed by having numerous bolts in a relatively tight pattern and reinforcing the flange plates. The top plate flange plate was designed to glue together on to the rigid carbon fiber bottom plate of the drone frame. Similarly, bolt pattern fits together into some of the pre-existing holes in the frame plate. This was done as both a reinforcement mechanism and to rigidly attach the frame and enclosure together. Similarly, bottom flange plate features a large orthogonal rib and is epoxied to the rest of the enclosure for reinforcement. The other aspect to consider within enclosure waterproofing is cabling. Various wires are required to go from inside to outside the enclosure. These include the motor wires, ballast servo and encoder wires, and wires for the external sensors. While pass through blocks or a commercial waterproofing component could have been used, we instead opted to use a simple hole and seal it with hot glue. This was because of the relatively low depths and pressures the quadcopter was designed to operate in, maxing out at only a couple of meters in depth.

The last major design constraint is ballast compatibility. As discussed in section 3.8, our group opted to utilize a ballast system to control the buoyancy of the quadcopter. Furthermore, the ballast system is designed to facilitate the transition between vertical flight and the horizontal position for underwater travel. These factors placed strict constraints on the overall density of the enclosure, its center of mass, and geometric center. Since the ballast system had to be smaller than the main enclosure, this meant that the enclosure -and more specifically the quadcopter a as a whole - had to have a density near water. This is because the range of effective densities produced by the ballast system had to allow for the quadcopter to both sink and float. In practice this constraint meant that the enclosure needed to be as small as possible. Any increase in enclosure volume meant an increase in overall drone weight, requiring higher motor throttle values to hover, and reducing the operating time. Both the center of mass and geometric center positions are factors crucial to the operation of the ballast system and are discussed further in that section. These factors did however influence the enclosure design primarily in its positioning. In order to keep the center of mass central and aligned with the frame and motors the heavy battery was placed near middle of the quadrotor. Similarly, most of the enclosure space and electronics were placed towards the back of the quadrotor, opposite the ballast system. This positioning allowed for a relatively central center of buoyancy.

Besides these primary design constraints there were a few other minor considerations made during the enclosure design process. One of the first was simplifying 3D printing, and specifically a focus was put on minimizing the amount of support structure required. Another was keeping the enclosure left-right symmetrical. Due to the ballast system and our planned tilt the quadrotor could not be front-back symmetrical, but an effort was made to keep it entirely symmetrical the other way. This somewhat simplified the design and assembly process as well as there would be fewer torquing issues. Hydrostatic Drag was also a minor consideration. Except for those used in racing, most quadcopters ignore air drag as it is negligible for the sizes and speeds they are moving at. This is not the case for most underwater travel and can often be a significant design constraint. While we did consider it, and specifically tried to minimize our cross-sectional area, it was not a major consideration due to the planned slow speeds and short time of underwater quadrotor operation.

3.7.2.1 Enclosure Version 1

Our first version of the enclosure was split into two separate parts, the battery box and electronics box. Each box had an internal rib that was designed to be epoxied together along with the lower flange plate. This split into two components was done to try and minimize the required printing supports, although a last-minute change in the camera positioning meant that both still required significant amounts of support material. Furthermore, the printing process left imperfections on the parts. Two significant imperfections proved to complicate the assembly process. The first was curling off the printing bed. This caused the two surfaces that needed to be glued to be cambered away from one another. This imperfection was addressed by using a large bead of epoxy to fill the gaps. The more pressing imperfection was small holes in various places around the enclosure boxes. Caused by improper layer adhesion, these holes allow water to seep through the otherwise sealed enclosure. Our group temporarily solved this issue by filling any visible holes with hot glue.

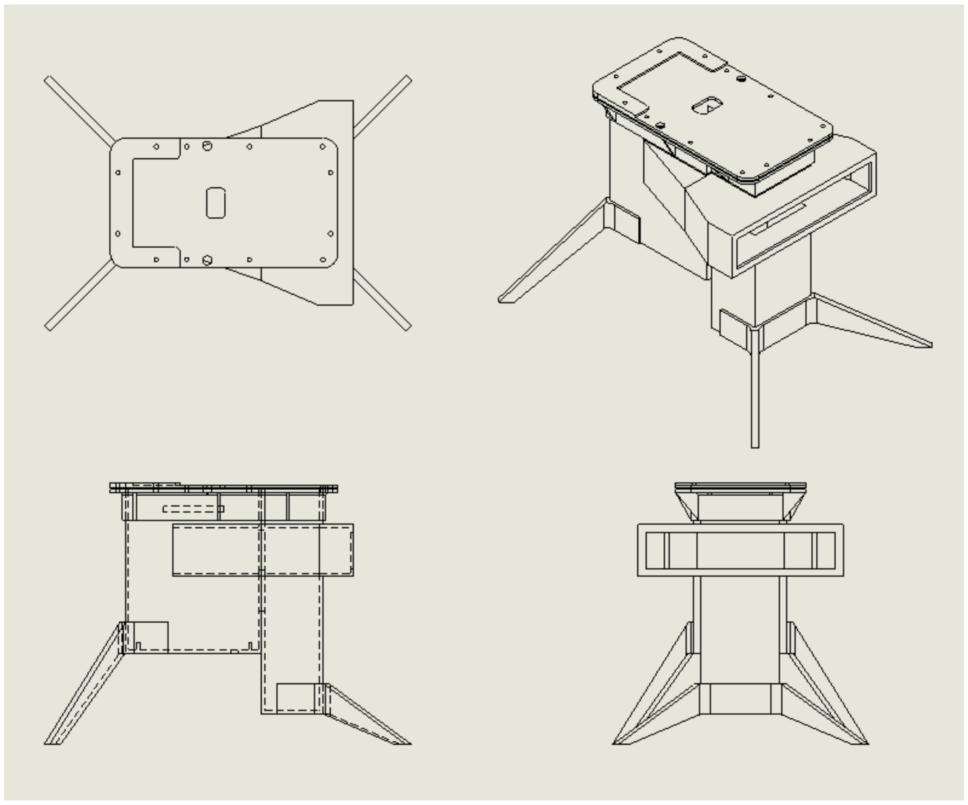


Fig. 3.34. Enclosure CAD Drawings.

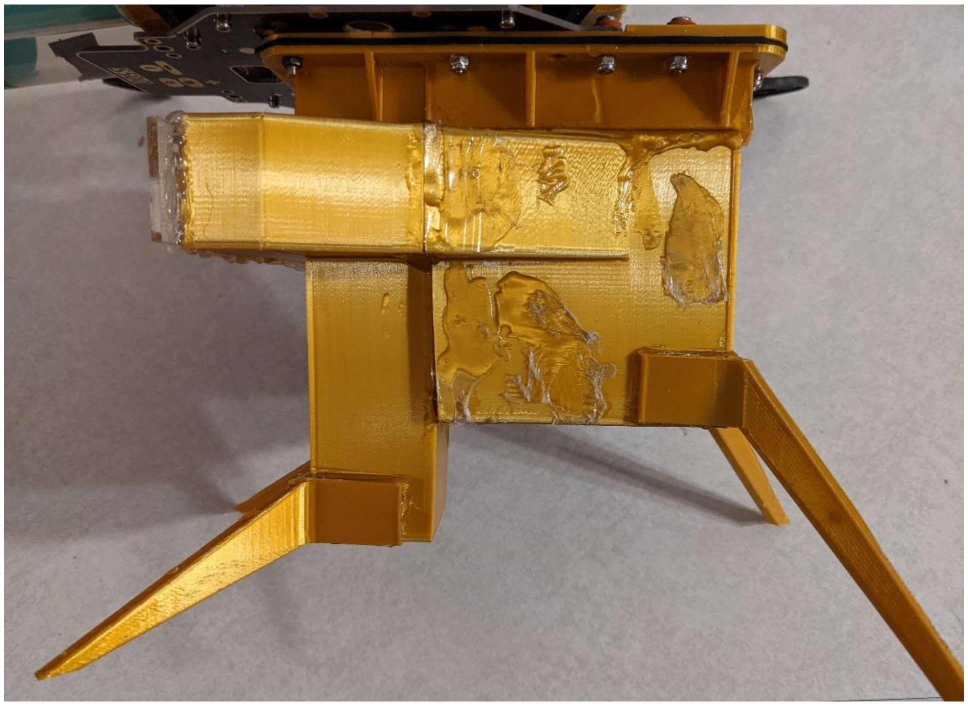


Fig. 3.35. Enclosure Assembly.

Besides the manufacturing defects the first enclosure version had two major design flaws that drove the implementation of a second version. The first design defect was an incorrect relief sizing for the camera. Due to the previously described internal rib for gluing the two components together, the clearance for taking the camera out was not large enough. This prevents the camera from being removed or put in place once the acrylic window has been glued on. This was not an acceptable option considering how much other testing had to be done with the camera. The other flaw, although less critical, is once again found with the camera enclosure area. Its positioning is too close to the top of the enclosure and restricts access to the flange bolts on that side.

3.7.2.2 Enclosure Version 2

Due to the issues with the first enclosure design, a second major revision was designed and manufactured. The primary change was the relocation of the camera bay. For easier access to the planned bolt pattern the camera bay was shifted downwards to the bottom of the enclosure. Furthermore, the large relief cuts were rotated from a horizontal orientation to nearly vertically. These changes minimized the required support structures and allowed for the primary enclosure to be printed in one piece instead of two. The lowered camera bay was also able to be utilized as a large front foot for the drone, eliminating some of the required leg supports. As with version 1, a separate flange piece was also printed and epoxied on. While moving the camera bay lower was somewhat detrimental to the desired center of mass and volume locations, the location difference in modeling proved to be negligible. Furthermore, the new version and orientation proved to be smaller in volume than version 1, allowing for a lighter overall drone.

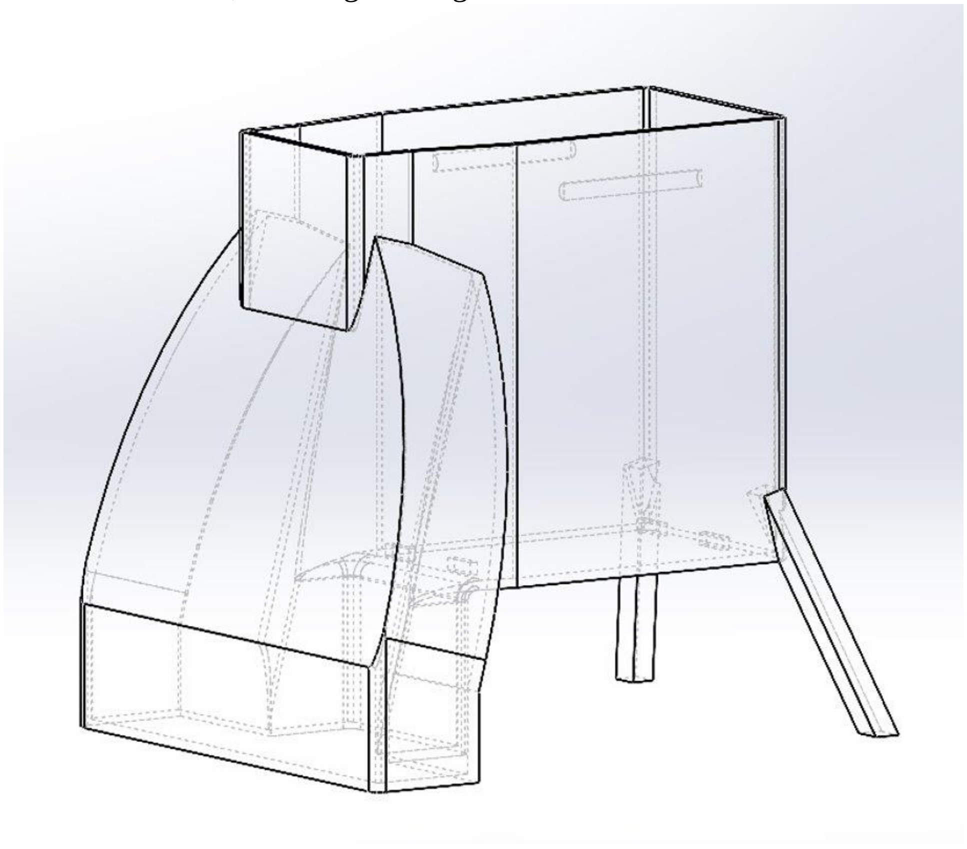


Fig. 3.36. Enclosure Version 2 CAD

While eliminating joints within the enclosure helped significantly with assembly time; other issues were caused by the increased size of the print. In particular, the greatly increased height of the enclosure print led to far more layer shifts and imperfections in the printing process. This meant that the overall enclosure had small holes almost everywhere. These issues likely appeared in version 1 but went unrecognized due to the larger leaks. In order to waterproof the enclosure, we utilized a commercial rubberized waterproof spray for much of the testing. While it required many coats, waterproofing with an external sealant ended up working. For later revisions a rubberized paint sealant was used over the spray for a thicker coat and faster drying times. This methodology worked even better in waterproofing but had the downside that adhesives, especially hot glue, struggled to adhere properly. For both the spray and paint we utilized flex seal brand sealant.

Despite significant improvements to the bolt pattern accessibility, the flange and gasket design proved to not be waterproof. The soft 3d printed material was unable to provide adequate clamping pressure on the rubber gasket. As an attempt to increase clamping pressure a beveled top plate, as opposed to flat, was tested. In theory this design was supposed to localize the clamping pressure to the inner corner of the enclosure, ensuring that the rubber gasket complied enough to fill in the gaps. While better, this flange-gasket again did not work. The hatch design that did end up working well was surprisingly to just glue the flange halves together, specifically with hot glue. The hot glue would adequately fill in any gaps and cracks while its low adhesive strength allowed for the flange to simply be pried apart after testing. Furthermore, the adhesive temperature of the hot glue is lower than the glass transition temperature of the PET-G, allowing us to reheat the glued flanges with a heat gun and reuse the hot glue. This methodology proved to be easier, faster, and more waterproof than the gasket and bolt pattern and would be how the hatch was sealed for much of our testing.

3.8 Ballast System

In order to traverse the water body, the structures sub-team discussed the different ways in which the quadrotor could move underwater. One potential way was to have a buoyant vehicle and treat the buoyancy force as an inversed gravity, requiring the quadrotor to constantly push itself underwater. This methodology was quickly ruled out for being too inefficient. and unstable. Another method would be to have the vehicle normally sink, and to simply hover in the water with the motors. This method however has significant concerns with exiting the water since motors would be required to push the quadrotor across the entire water-air transition. The third option, and the one we chose, was to actively control the buoyancy of the quadrotor. This allows the quadrotor to float on the water surface, passively sink to the desired depth, travel underwater, then float back up to the surface. By having a buoyant state, the quadrotor propellers would naturally be pushed up out of the water, eliminating the need for a powered water-air transition.

Furthermore, an active buoyancy system has other advantages for underwater movement. If the buoyancy at depth is near neutral, the quadrotor cannot translate a tilted hover into horizontal movement like it does in air – since it would just push itself upwards. This is typically addressed by tilting the quadrotor so that rather than vertical thrust the motors produce horizontal thrust. This also has the advantage of greatly improving underwater efficiency by reducing drag and greatly improving angle thrust losses. This tilting motion can be done propulsively, with the two front motors rotating at a slower speed and the back two rotating at a faster speed. However, this tilting motion adds significant control complexity since the quadrotor can only be dynamically stable in one position, vertically or horizontally. An active ballast system can solve this issue. Demonstrated by the looncopter, a ballast system can not only change the density of a vehicle, but also the position of its center of mass, allowing a shift in where the quadrotor is dynamically stable.

Inspired by the looncopter, our ballast system was designed to provide stability in two primary cases. When the ballast system is empty, and the quadrotor floats, the center of mass and buoyancy are aligned so that the system floats vertically, in the same orientation as aerial flight. When the ballast system is ~75% full, and the quadrotor sinks, the center of mass and buoyancy is instead aligned so that the quadrotor is dynamically stable in a horizontal position to travel underwater.

3.8.1 Hydrostatic Analysis

For this project, the theoretical motions had to be verified to determine if the system would perform as intended. Modeling is a simple analytical method for verifying unique motions that our drone will have to take, such as the air-to-water transition. The team decided to pursue a hydrostatic study of our drone system to accomplish this goal. The system in question is being modeled as an inverted pendulum underwater. To determine the forces acting on the system, two unique principles were used to help identify how the forces interacted within the system: Archimedes Principle and Centroids.

Archimedes Principle states that the upward buoyant force that is exerted on a body immersed in a fluid is equal to the amount of displaced fluid by the volume of the body itself. This would mean that our drone, when either partially or fully submerged in water will exert a force upward on the body, dependent on how much of the body is underwater. Due to the specific point at which this analysis is being conducted – at the air-to-water transition – we can assume that the drone is nearly fully submerged, and thus the dynamics when at the water surface can be neglected.

The center of mass of the drone – the location where the Force of Gravity acts – can be determined by assuming uniform density among components. Treating the drone as a composite shape, one can use the equations for a centroid below to generate three dimensional coordinates for the center of mass by compiling the X, Y and Z coordinates of each component, and treating them as a different composite in the overall shape.

$$Center\ of\ Mass = (\bar{X}, \bar{Y}, \bar{Z}) = \frac{\sum_1^N (x, y, z)_i M_i}{\sum_1^N M_i} \tag{84}$$

A similar technique can be applied to solve for the Center of Buoyancy, where the mass of each component is replaced with the density of water multiplied by the volume of the component. This acts as a stand in for displaced water by the individual component.

$$Center\ of\ Buoyency = (\bar{X}, \bar{Y}, \bar{Z}) = \frac{\sum_1^N (x,y,z)_i \rho_{H20} V_i}{\sum_1^N \rho_{H20} V_i} \tag{85}$$

These two principles are implemented in the diagram below, modeling the Centers of Mass and Buoyancy, as well as the acting Forces of Buoyancy and Gravity. Other parameters are defined in the diagram as geometric parameters to help define the system.

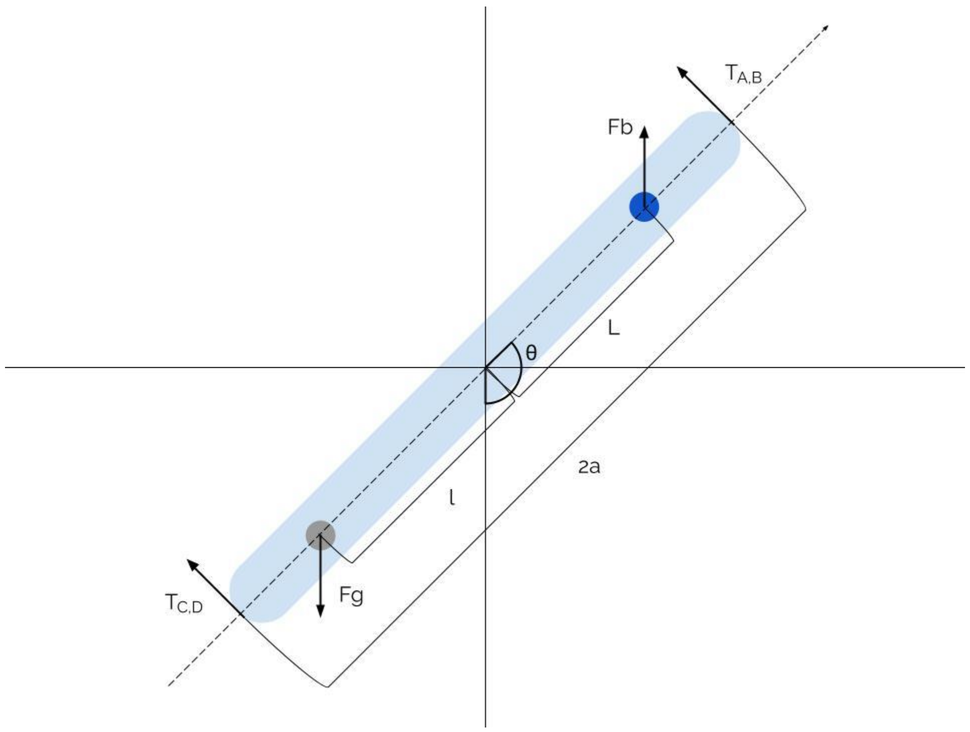


Fig. 3.37. Image of Proposed Model.

For the following model, the following variables are defined:

Table of Variables	
m	Mass of the Drone
I	Moment of Inertia of the Drone
g	Gravitational Constant
l	Distance from the Axis of Rotation to the Center of Buoyancy
L	Distance from the Axis of Rotation to the Center of Mass
a	Arm Length
ρ	Density of Water
V	Volume of the Drone
F_g	Force of Gravity acting at the C.o.M.
F_B	Force of Buoyancy acting at the C.o.B.
$T_{a,b,c,d}$	Thrust generated from the 4 propellers

With a stable force of buoyancy and no other external forces acting on the body, the only way to adjust our vehicle to fly horizonal to the pool surface would be to shift the center of mass. The ballast system was designed to intake water to add mass to the system and adjust the center of mass of the vehicle. Adding mass to the system without additional volume can shift the center of mass with respect to the center of buoyancy, creating a moment that can rotate the vehicle. From this, our group began pursuing designs to implement an active ballast that would be able to shift the center of mass of our vehicle enough to turn our vehicle underwater.

3.8.2 Ballast Version 1

From the start our group decided to utilize syringes for our ballast system. While large ballast systems typically use internal tanks and pumps, at the scale of our quadrotor these pumps would add significant weight and complexity. This is particularly true since unlike a ballast system for a boat which has access to air at any time, any air displaced by our ballast needs to be accounted for and reused when trying to empty the ballast system underwater. Excluding pumps leaves only the option of changing the quadrotor volume. Syringes were the clearest option to do so, offering an extremely proven dynamic seal design cheaply and with low weight. Syringes also had the advantage of offering a large variety of sizes, an important aspect since our ballast system was sized based on the total displacement of the quadrotor.

To move the ballast system a small servo was selected based on the approximate torque required and available electronics voltage. Initially a custom servo horn lever and a slotted plate were designed, and 3D printed. The slotted plate would be glued to the syringe plunger while the lever would be pinned in the slot. A 180-degree rotation of the servo motor would be a full extension or retraction of the plunger. This system was mirrored into a set of two to increase the ballast size and create symmetry, although the second mechanism was never assembled for this test. All components were held together by a 3D printed external shell.

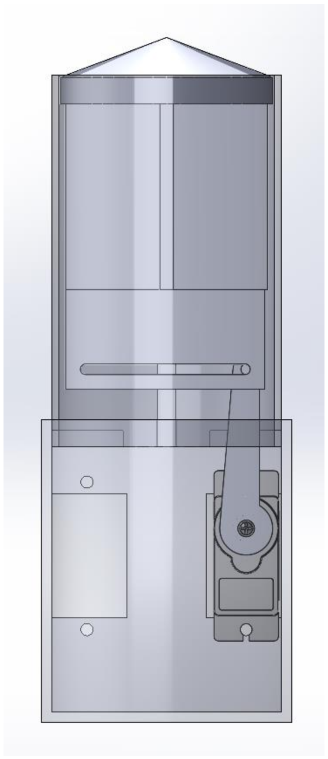


Fig. 3.38. Ballast System Version 1 CAD.

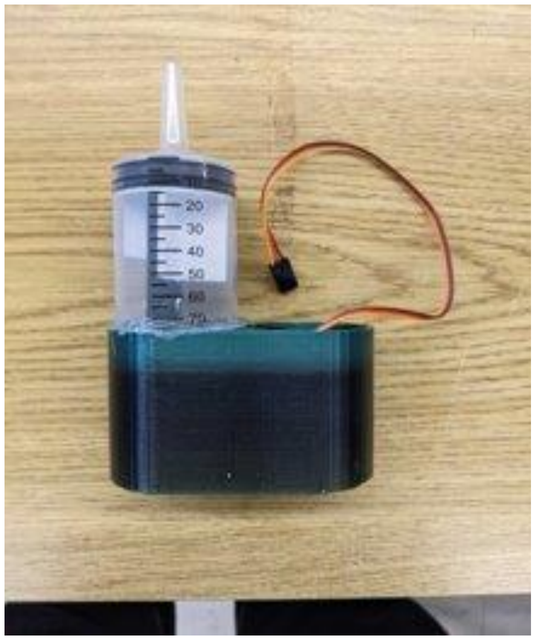


Fig. 3.39. Ballast System Version 1 Assembly.

The first ballast test had many issues. The first issue the team encountered was that our 3D printer did not have the required resolution to match the servo horn spline. For the initial assembly the part was melted and pressed on to the servo, but this would not be a reliable option for future systems. The other, more important, issue was a flaw in the overall mechanism. Due to the unexpectedly high friction force of the syringe the servo motor barely had enough torque with that moment arm. This combined with the significant torque losses due to the leaver angle meant that the servo was unable to move at all. The tolerances within the 3D printed enclosure for the servo mounts were also off.

3.8.3 Ballast Version 2

For the second ballast test we decided to change the linear motion mechanism. Another option could have been to increase the servo motor strength, but this would require a larger ballast system and even then, the larger servo options were limited by our electronics voltage. The smaller 5-volt servos shared a power distribution system with the raspberry pi, but larger ones would require a separate power distribution system.

Instead, we opted for a rack and pinion design since a small pinion gear greatly reduced the effective lever length and since the mechanism has a relatively constant torque curve along the entire ballast system, as opposed to the sinusoidal torque curve of the lever design. The rack and pinion mechanism also allowed for the ballast system to be as long as required, rather than limited by the lever arm length. In exchange, however, the mechanism required a larger range of motion than a typical servo can provide. We opted to continue using the same servo, but as a continuous rotation variant. Continuous rotation servos are basically just throttleable electric motors, so they lose the position tracking of a standard servo. This state tracking was re-implemented in the third ballast version using a rotary encoder but was not used for this test.

Both the rack and pinion gears were semi-circular toothed gears that were 3D printed. To get around the minimum resolution issue experienced with the servo arm in test 1 the pinion gear was designed to interface with a pre-existing servo horn that was purchased with the servos. The enclosure was once again 3D printed, although only 1 mechanism was designed to be assembled from the start.

Ballast test 2 proved to be a success. The rack and pinion gears meshed surprisingly well, and the servo was able to easily move the syringe. Tolerances were also far more precise than the first version.

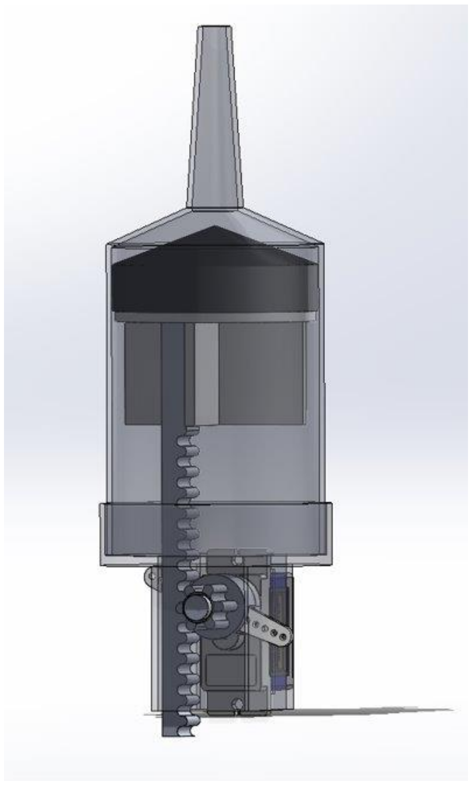


Fig. 3.40. Ballast System Version 2 CAD.

3.8.4 Ballast Version 3

Version 3 of the ballast system was designed to not be a test, but instead designed to integrate directly with the quadrotor and enclosure. Like previous designs the system uses a rack and pinion mechanism with small FS90 continuous rotation servo motors arranged in symmetry with two syringes.

The syringes are 150 ml syringes with a 40mm internal diameter. These were cut to be ~9 cm long. The syringes were chosen based on diameter so that two would fit reasonably well within the arms of the quadrotor frame. The length was chosen based on our target ballast volume of ~10% of the quadrotor volume. While a larger ballast system would likely be better for stability and buoyancy, size was limited by the available space and mass. The 10% target, or ~200 mL of volume, was evaluated to provide an adequate shift to the center of mass without being too large.

Unlike previous iterations of the ballast system which had been planned to be mounted vertically, this version finalized on a horizontal system to avoid potential collisions with the propellers. Overall, this decision did not change the mechanism very much but did significantly change the mounting possibilities and ballast enclosure area. Specifically, this new positioning puts the ballast system practically inside the frame. As a result, the ballast system was designed as a replacement top frame plate and clearance had to be added to avoid the arms.

As a result of these changes the ballast enclosure and frame were split into three parts: the front half holding the syringes, the back half holding the servos and encoders, and a frame to hold everything together. As discussed in section 3.9.2, this more finalized version of the ballast system includes encoders. These encoders are the common ky-40 rotary encoder. They interface with the pinion gear and are held in place by the frame. In fact, the servos serve as a stabilization bearing for the pinion gear that is opposite to the servo motor. This was previously a simple pin and hole in the 3D printed parts. Other changes include a more robust servo mounting system with clips, and new rack gears. Previous rack gears were designed to be glued to a trimmed syringe plunger, but the new design replaces the plunger with an entirely 3D printed part, where the rubber endcap was slotted on. This choice was made so that the pinion gears could abut right against the syringe to stabilize the entire plunger and prevent the rack gear from slipping out or skipping. Other stabilizing features were also designed into the back half where the gears mesh.

All these components were glued together into one watertight, but separate, enclosure that bolts directly to the frame. Any external component was printed in a similar fashion to the enclosure as discussed in section 3.7.1. This separation was done out of precaution so that even if the ballast system leaks none of the expensive electronics would get damaged.

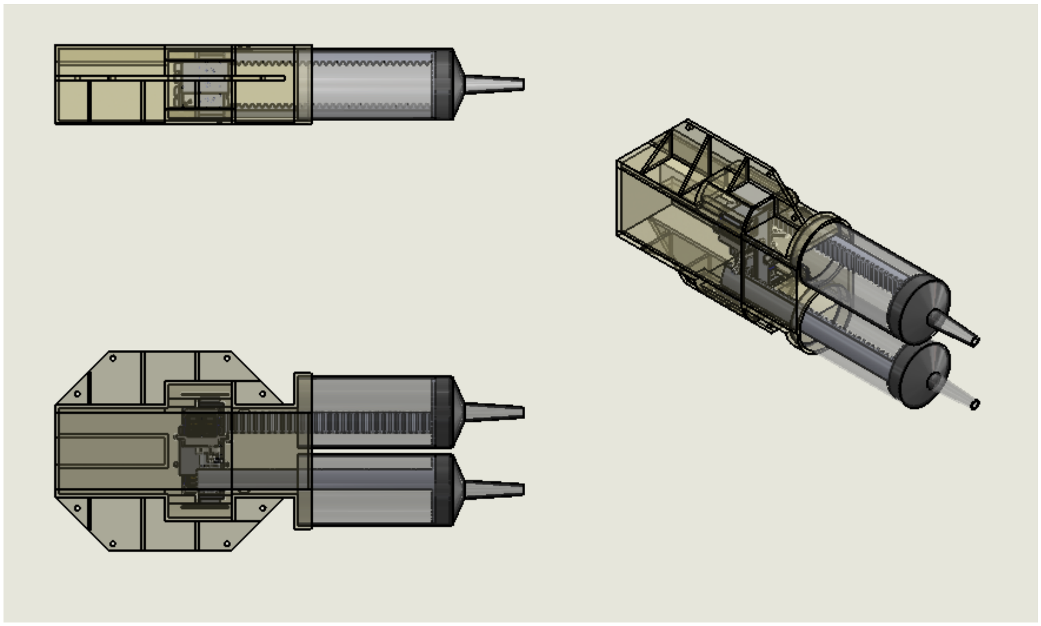


Fig. 3.41. Ballast System Version 3 CAD External View.

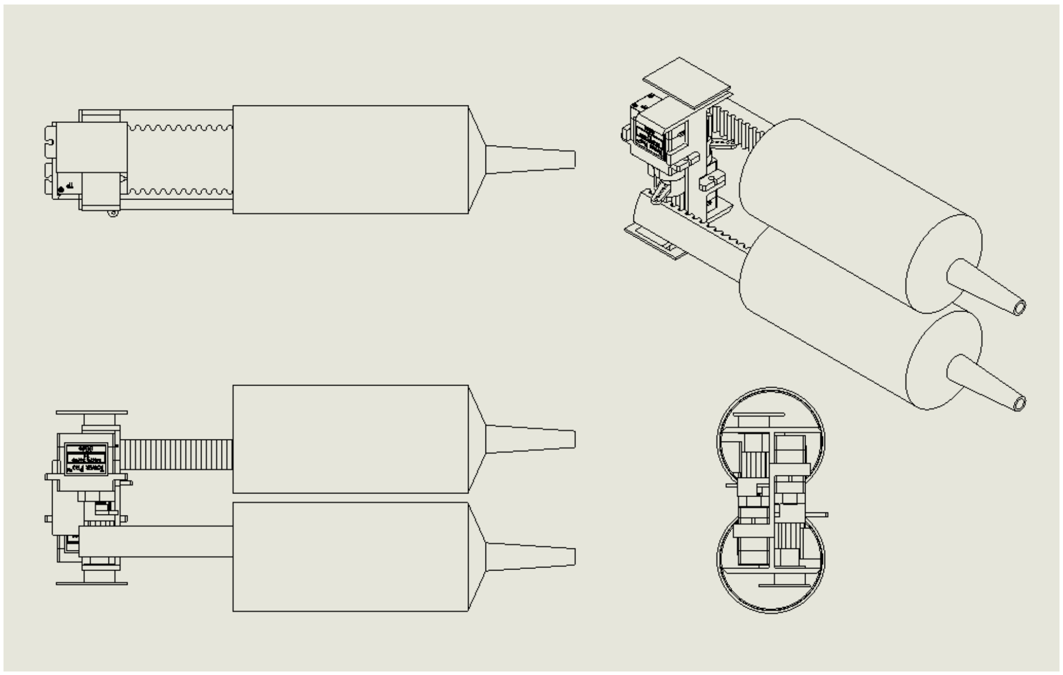


Fig. 3.42. Ballast System Version 3 CAD Internal View.

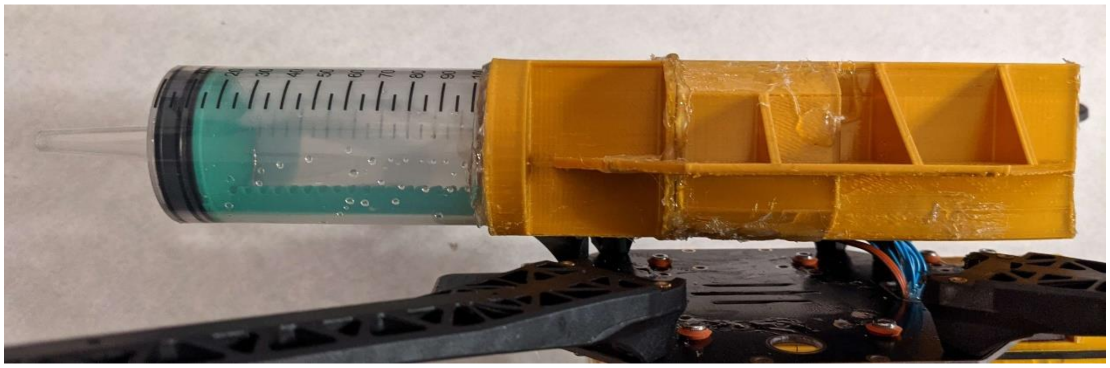


Fig. 3.43. Ballast System Version 3 Assembly.

This iteration of the ballast system experienced some difficulties between fabrication errors and mechanical advantage. Despite the success of version 2, this version struggled to actuate the syringes properly. The first issues came in early development and fabrication of the system, where the adhesive material used to join the modified syringes to the PETG prints – a combination of waterproofing and commercially available hot glue – caused the syringe bodies to warp, creating difficulties in the plunger's ability to move forward and back. Furthermore, the syringes were significantly more difficult to move even outside the warped areas. This was attributed to the aging of the rubber seal over the winter break. To address these factors a more careful fabrication methodology was employed, as well as lubricating the sliding parts with a silicone-based lubricant, to dually act as a further sealant between the water and the electronics inside, and a lubricant for the rubber plunger. This proved to improve the mobility of the plunger but was not enough to assist with the torque required by the servos to actuate the system. Another issue identified and fixed with version 3 is the off-center plunger racks. Because of their position on the plunger edge, a weak design, and fatigue from testing; the rubber seal and plunger head were being canted against the syringe exterior, significantly increasing friction. A replacement plunger rack was implemented with a large brace, and did help, but not enough to allow for smooth articulation of the syringes. Other solutions tested include new servo motors – in case the existing ones had overheated too much from testing and partially demagnetized the DC motors within them; a smaller pinion gear; and even a worm gear drive. None of these variations were able to smoothly articulate the syringe, although the worm gear setup had adequate torque but poor tolerances due to a 3D printed design. Variations on the worm gears with a screw drive mechanism is being considered to decrease imperfections and greatly increase torque.

4 Results

4.1 Summary

Through the research completed, the following results were determined. The project was successful in partial fulfillment of the initial goals. Significant progress was achieved towards the structural design of an amphibious quadrotor. The team's development of a novel ballast system demonstrated the ability to intake and expel water in such a manner to alter the vehicle's buoyancy characteristics. Furthermore, several different waterproofing techniques were tested, and their performance analyzed. This task proved to be difficult, and the waterproofing challenges led to several cases of water intrusion during underwater testing events. The team determined that applying a silicone coating was the most effective method of waterproofing 3D printed components, due to its ability to cover the small holes in the material. A hydrostatic analysis conducted by the team also provided valuable insight into the behavior and control of a quadrotor vehicle when underwater, and importantly during air to water transitions. This work helped to inform the team's careful design of the structure with respect to the vehicle's mass distribution and ballast system placement to achieve the desired performance. While this research presents many critical insights towards the design of an amphibious quadrotor vehicle, the final vehicle constructed by the team was not proven to perform adequately in the real world due largely to manufacturing and waterproofing challenges. We expect this work to lay a strong foundation for further development in this unique research space.

The team also made considerable progress in the development of a flight computer and sensor array for autonomous navigation and localization. The cost of electronics was driven down through the use of inexpensive hobbyist electronics and accessible programming environments. The electrical system for the vehicle was proven to be capable of power regulation to drive 12V brushless motors as well as 5V servos, sensors and control electronics. Additionally, the onboard sensors were shown to produce accurate state estimation which allowed for demonstrated stable flight of the test vehicle. Extensive testing of the RealSense T265 tracking camera also demonstrated the acquisition of reliable localization data, which shows considerable promise is unpiloted navigation between aerial and underwater environments. The completed software includes a unique integration of both aerial and underwater dynamics along with the control logic to switch between the two, which is a feature that is not available with commercial flight controllers. This system was tested in part with isolated subsystems and was shown to successfully achieve autonomous altitude control on the test vehicle. However, the full functionality of the custom flight computer is yet to be tested, as the team was unable to fully integrate the sensor array into the final vehicle and to undertake autonomous flight testing within the timeframe of the project. As with the research into the quadrotor's structure, we believe that the most recent iterations of the electrical system and flight control software will provide considerable value in future work to fully test and extend their capabilities.

4.2 Conclusions

The overall objective of this project was to design, construct and test an autonomous quadrotor capable of flight and underwater locomotion. The overall design created over the duration of the project has been proven to be an effective method of bridging the gap between travel in-air and below the surface of water. Through the creation of a ballast and GPS denied position tracker, the quadrotor prototyped is fully capable of subnautic travel. Through hydrostatic calculations, it was determined and proved that with the center of mass and buoyancy generated by the design would allow for the correct orientation of the craft underwater, thus proving the soundness of the enclosure design and weight distribution. The control system created has been proven to be able to achieve autonomous movements and has been tested through the use of our test vehicle. With all this presented research, and further prototypes and testing, the design created is proven to be a viable solution to the problem presented at the beginning of this paper.

4.3 Recommendations for Future Work

In accordance with the accumulation of research presented within this paper, the team has a few recommendations for the continuation of this project. It is believed the overall mission and goal of the project is able to be achieved with further prototyping and flight code testing on a final vehicle. The first recommendation is to use more commercially available products, rather than custom designs. In particular, the two enclosures would have made more sense as standard containers. A significant amount of time was spent trying to waterproof the 3d printed parts and hatch seal where a commercial product would have fit the use case extremely well. While the design could not have been as custom, a wide variety of waterproof resealable containers are available in the required size range. Commercial products would also be generally stronger and less prone to cracking than 3d printed parts, an issue faced when flight testing inevitably ends.

A second recommendation is to test more frequently. Rather than large full-scale testing, subsystems and components should be tested individually. While this methodology was adopted later within the project, earlier subsystem testing would have helped significantly. Flaws within the ballast system and waterproofing would have been discovered earlier, electronics would not have been lost to water damage, and more overall revisions and changes could have been made.

A final recommendation is to explore other possibilities for optimized underwater travel. While in theory the ballast system and shifting center of mass should work, it added significant mechanical complexity and design constraints. A simpler more symmetric structure that abandons the underwater tilt may be more optimal, especially for tuning air flight characteristics. Active depth control through propeller throttle may also be easier than manipulating buoyancy with a ballast system.

4.4 Broader Impact

The applications of a fully autonomous, amphibious AUV are extensive. From undersea wreckage exploration to military intelligence gathering in foreign waters, a vehicle of this type would have no shortage of uses. Being sponsored by the US Navy, this project holds within it a possible direction the future of joint naval and aerial mission can take. Reconnaissance, data collection, and other designated missions that must navigate both the air and the water become possible with an amphibious AUV.

The work conducted within this project regarding underwater dynamics, conceptual design, and buoyancy manipulation can be applied to other projects, and even the lessons learned in waterproofing and other novel takeaways this team had are valuable to anyone attempting a project with a similar scope. As the world of autonomy evolves, and the boundaries between mediums of travel dissolve, the merging of transportation modes will see more prevalence in everyday life. Being able to seamlessly travel between air and water means overcoming the difficulties that lie within each medium in one vehicle. In this case a drone seems to be the best candidate for this role, but who knows what the future holds.

5 References

[1] G. Warwick, "A Brief History of Rotorcraft Development," Aviation Week Network, 22 August 2018. [Online]. Available: https://aviationweek.com/business-aviation/brief-history-rotorcraft-development.. [Accessed 16 October 2022].

[2] L. Mejias, J.-P. Diguet, C. Dezan, D. Campbell, J. Kok and G. Coppin, "Embedded Computation Architectures for Autonomy in Unmanned Aircraft Systems (UAS)," *PubMed Central,* vol. 21, no. 4, 2021.

[3] "Unmanned Aerial Vehicals: (Uavs)," Encyclopedia Britannica , [Online]. Available: https://www.britannica.com/technology/military-aircraft/Unmanned-aerial-vehicles-UAVs. [Accessed 2 October 2022].

[4] S. Ghazbi Norouzi, Y. Aghli, M. Alimohammadi and A. Akbari, "Quadrotors Unmanned Aerial Vehicals: A Review," IEEE, 2016.

[5] M. J. Monfreda, J. D. Gibbons, A. P. Lepilov, M. Hobson-Dupont, A. S. Knight, M. R. Dupuis and G. M. Mungai, "Design Optimization of a Quad-Rotor Capable of Autonomous Flight," Worcester Polytechnic Institute, Worcester, MA, 2008.

[6] K. E. Gustafson, A. DiCesare and P. V. Lindenfelzer, "Design Optimization of a Quad-Rotor Capable of Autonomous Flight," Worcester Polytechnic Institute, Worcester, MA, 2009.

[7] R. J. D'Angelo and R. C. Levin, "Design of an Autonomous Quadrotor UAV for Urban Search and Rescue," Worcester Polytechnic Institute, Worcester, MA, 2011.

[8] S. L. Friedman, K. P. Hancock and C. M. Ketchum, "Vision-Based Obstacle Avoidance for Small UAVs," Worcester Polytechnic Institute, Worcester, MA, 2015.

[9] J. D. Blythe, K. A. Borowicz and A. N. Hollander, "Autonomous Quadrotor Navigation and Guidance," Worcester Polytechnic Institute, Worcester, MA, 2016.

[10] B. M. Ferrarotti, R. D. Patil, D. H. Driscoll and J. I. Karlin, "Quad-plane Design for Autonomous Cargo Delivery," Worcester Polytechnic Institute, Worcester, MA, 2020.

[11] M. P. Umbricht, C. W. Blomquist, R. J. Bellitto and K. T. Blackstock, "A High Performance Aircraft for the 2020 WPI UAV Competition," Worcester Polytechnic Institute, Worcester, MA, 2020.

[12] N. Hesel, A. Agarwal, M. Runquist and A. Calcagni, "Low-Cost Quadrotor Micro-Aerial Vehicle," Worcester Polytechnic Institute, Worcester, MA, 2022.

[13] H. Alzu'bi, I. Mansour and O. Rawashdeh, "Loon Copter: Implementation of a Hybrid Unmanned Aquatic-Aerial," Electrical and Computer Engineering Department, Oakland University, Rochester, 2018.

[14] D. B. Larter, "Video: Drone flies and swims, grabs the Navy's attention," Navy Times, 10 January 2016. [Online]. Available: https://www.navytimes.com/news/your-navy/2016/01/10/video-drone-flies-and-swims-grabs-the-navy-s-attention/. [Accessed 16 October 2022].

[15] Nortek, "Nortek Wiki," Nortek, [Online]. Available: https://www.nortekgroup.com/knowledge-center/wiki/new-to-subsea-navigation. [Accessed 1 October 2022].

[16] A. Bachrach, A. de Winter, R. He, G. Hemann, S. Prentice and N. Roy, "RANGE - robust autonomous navigation in GPS-denied environments," MIT: Insitute of Electrical and Electronics Engineers, Camberage, Massichusets, 2010.

[17] R. Olfati-Saber, "Distributed Kalman Filterting for Sensor Networks," *46th IEEE Conference on Decision and Control,* no. 0191-2216, 2007.

[18] N. Rehm, "dRehmFlight VTOL Flight Controller," Jan 2020. [Online]. Available: https://github/nickrehm/dRehmFlight. [Accessed 2022].

[19] "A02YYUW Waterproof Ultrasonic Sensor," DFROBOT, 2020. [Online]. Available: https://www.dfrobot.com/product-1935.html. [Accessed 2022].

[20] "Bar30 High-Resolution 300m Depth/Preasure Sensor," BlueRobotics, 2022. [Online]. Available: https://bluerobotics.com/store/sensors-sonars-cameras/sensors/bar30-sensor-r1/. [Accessed 2022].

[21] "Intel RealSense Tracking Camera T265," Intel, 2016. [Online]. Available: https://www.intelrealsense.com/tracking-camera-t265/. [Accessed 2022].

[22] S. O. H. Madgwick, "An efficicent orientation filter for inertial and inertial/magnetic sensor arrays," Univercity of Washington, Seattle, 2010.

[23] M. M. Maia, P. Soni and F. J. Diez-Garias, "Demonstration of an Aerial and Submersible Vehicle Capable of Flight and Underwater Navigation with Seamless Air-Water Transition," Rutgers Univercity, New Brunswick.

[24] J. Jiang, J. Qi, D. Song and J. Han, "Control platform and design and expirament of a quadrotor," Proceedings of the 32nd Chinese Control Conference, 2013.

[25] J. Busquets, J. V. Busquets, D. Tudela, F. Perez, J. Busquets-Carbonell, A. Barbera, C. Rodriguez, A. J. Garcia and J. Gilabert, "Low-cost AUV based on Arduino open sources mircocontroller board for oceanographic research applications in a collaborative long term deployment missions and suitable for combining with an USV as autonomous automatic recharging platform," IEEE, Valencia, 2012.

Appendices

Appendix A: Flight Code

The flight code developed for this project is available online at https://github.com/Michael-Beskid/Amphibious-AUV. The published release titled “Amphibious-AUV_v1.0” includes the final version used in the demonstration of the vehicle. For future work, it is recommended to fork this repository to continue development and add additional classes and features to support different missions while taking advantage of this existing framework created by our team.