Worcester Polytechnic Institute
Robotics Engineering Program

SailBot 2019-2020 Final Report

A Major Qualifying Project Report
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science by:

Mylena Guzman, ME
Irina Lavryonova, RBE/ECE
Kyle Reese, RBE/CS
Andrew Thomas, ME

Project Advisors:
Professor William Michalson
Professor Kenneth Stafford
Date: 12 May 2020

Abstract

The goal of this project was to improve in major ways upon the autonomous robotic sailboat known as the “The Wide Awake.” Using the rules of the International Robotic Sailing Regatta as a guide, the team improved the boat’s mechanical, electrical, and software performance and reliability by improving upon the systems that were already in place from previous years as well as devising new solutions to existing problems. The focus of this year’s MQP was to implement reliable RC control and to pave the way for the implementation of autonomous operation in future years.

Authorship

Section	Authors
Abstract	Irina Lavryonova
Introduction	Irina Lavryonova
Background	
Sailing Basics	Irina Lavryonova
Lake Attitash	Irina Lavryonova
Past WPI SailBots	Irina Lavryonova, Andrew Thomas, Mylena Guzman
Purpose and Project Goals	
Mechanical Goals	Mylena Guzman, Andrew Thomas
Electrical Goals	Irina Lavryonova
Software Goals	Irina Lavryonova, Kyle Reese
Methodology	
System Overview	Irina Lavryonova
Mechanical Systems	Andrew Thomas, Mylena Guzman
Electrical Systems	Irina Lavryonova, Mylena Guzman, Kyle Reese
Software Designs	Irina Lavryonova
Results and Future Recommendations	
Definite Goals	Mylena Guzman, Irina Lavryonova, Kyle Reese, Andrew Thomas
Stretch Goals	Mylena Guzman, Irina Lavryonova, Kyle Reese, Andrew Thomas

Contents

Table of Figures

Table of Tables

1. Introduction

The 2019-2020 SailBot Major Qualifying Project (MQP) is a continuation of three previous MQPs. The basic premise of this project is to build an autonomous robotic sailboat that includes design and control components influenced by the rules of the International Robotic Sailing Regatta (IRSR). The goal of this year's MQP is to repair and improve upon previous systems in order to create a technologically sound base for future use of the boat. Following are the tasks from the IRSR that served as goals for what the boat should accomplish by the end of the MQP.

- A fleet race that uses manual control to maneuver around a triangular course
- An endurance test that has the boat sail around a rectangular course marked by buoys for seven hours

To this end, the team will improve upon the keel, movable ballast, rigid wing sail, and electrical and software systems. The keel cavity in the hull was fragile and prone to leakage, while the movable ballast had a tendency to travel outside of its design limits, as shown by Figure 1. In order to resolve the leak issues, the team resealed the keel cavity while slimming the profile of the keel insert in order to allow more material to define the cavity. The movable ballast now has hard stops at the end of its designed travel limits with electromagnetic switches complimenting the system at either end by controlling the power to the system.

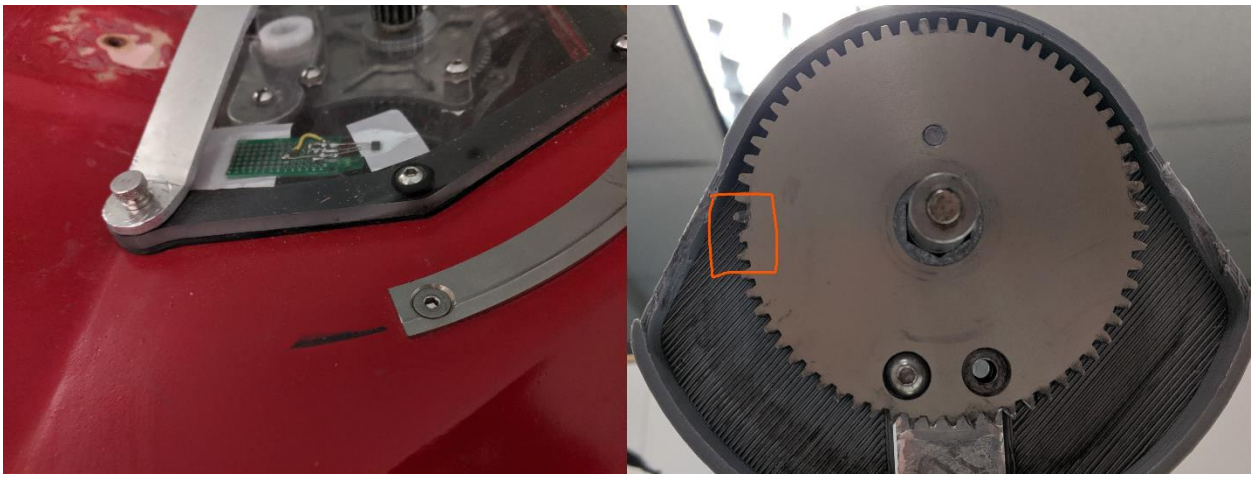

Figure 1: Hull shows markings of overextension and the movable ballast gear has a chipped tooth, marked by a red square.

Electrical and software systems were also improved. The relative wind direction encoder on the rigid wing sail provided unreliable readings and the system could not communicate with the hull. Improper electrical connections to the encoder caused the spurious readings, while the communications issue was resolved after the team integrated a dedicated library into the code. Furthermore, the microcontroller in the hull proved inadequate to implement the goals of this year's team. As such, it was replaced with a single board computer running Linux for embedded systems.

2. Background

2.1. Sailing Basics

A required topic for making a robotic sailboat that sails well is knowing how to sail a regular, manual sailboat. This sailing basics section covers the points of sail, which determine the position of the sail relative to the direction of the wind, and the right of way rules, which are crucial to follow to enable safe sailing by making the SailBot's actions predictable to fellow mariners.

2.1.1. Points of Sail

A sailboat uses many of the same physical concepts to develop locomotion as an airplane. A sailboat uses the sail to develop lift in much the same way that an airplane's wing develops lift. Changing the angle of attack, which is the angle between the chord of the sail and the direction of the relative wind, determines the direction of lift relative to boat orientation, while the strength of the relative wind determines the magnitude of the lift vector. The points of sail are different hull directions which will determine the sail trim and speed at which the sailboat can move. Figure 2 below shows five of the six points of sail.

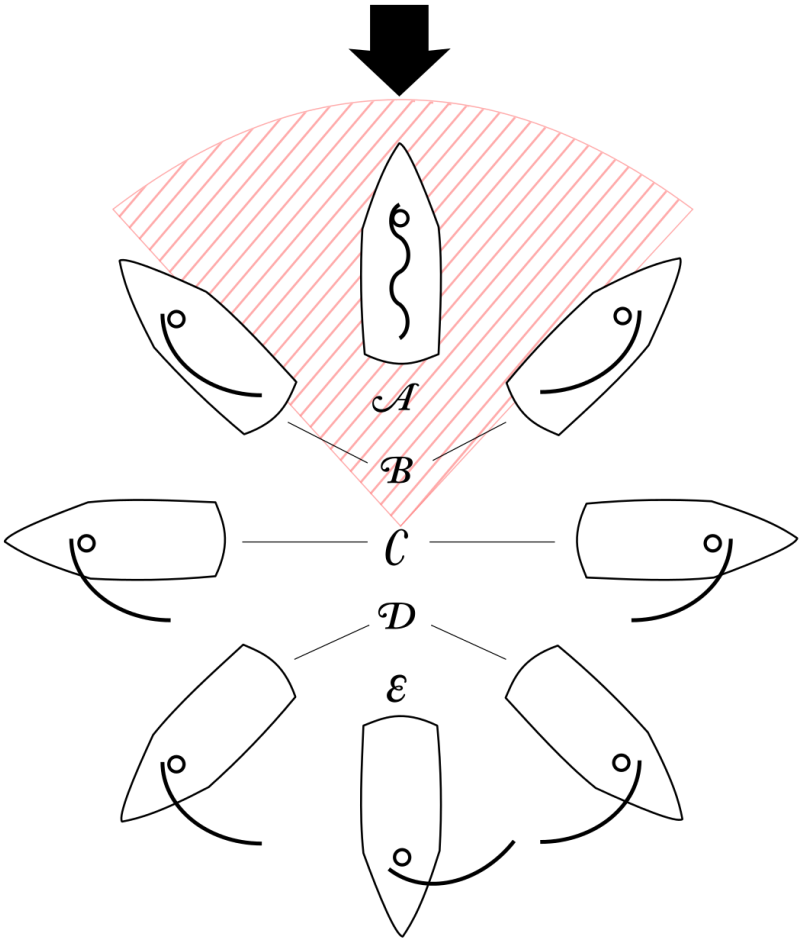

Figure 2: Diagram showing the points of sail [1].

The arrow in the figure above shows the direction of wind and each letter shows a boat in a different point of sail: A is in irons, B is close-hauled, C is on a beam reach, D is on a broad reach, and E is running. The point of sail the figure above does not show is a close reach, which is an angle between a close haul (B) and a beam reach (C). The area shaded in red is the "no-go zone" where a sailboat will be in irons.

As the "no-go zone" implies, it is impossible for a boat to sail directly upwind. However, a sailor can use a maneuver called tacking, which involves alternating between being close-hauled to either side.

Like any force vector, the lift generated by a sail can be decomposed into three components. The forward component is the primary force propelling the boat and results in forward motion. The sideways lift component creates a moment that would immediately capsize the boat if not for the keel ballast and the crew moving to counterbalance the heeling moment.

2.1.2. Right of Way

In addition to points of sail, the other crucial piece of sailing information to know is the right of way rules. They ensure that all participants in marine traffic are predictable, which allows for safety and efficiency [2].

In the case of right of way between two sailboats, the right of way depends on the relative wind direction, whether the boats are on the same tack (which direction the sail is pointing), and the downwind position of each sailboat. See the USCG Navigation Rules document, Rule 12, for official language [3].

The basis for marine right of way rules between different classes of vessel is the same as those for land and air vehicles, which is that the least maneuverable vehicle has the right of way in an encounter between different types of vehicles. In the case of a sailboat, it will have the right of way over a powerboat, but with some exceptions. An example of such an exception is overtaking, where the overtaking vessel forfeits its right of way [3]. Additional rules exist for head-on (Rule 14 of [3]) and crossing encounters (Rule 15 of [3]), as well as special circumstances such as a vessel not under command or with restricted maneuverability (Rule 18 of [3]).

.

2.2. Lake Attitash

Lake Attitash is split between Amesbury and Merrimac, MA near the Eastern MA/NH border. The lake is relatively shallow with an average depth of only 11 ft [4]. This is starkly different from Lake Quinsigamond's average depth of 21 ft in its shallower Southern basin, which is where the IRSR has taken place in previous years [5]. The shallow nature of the lake coupled with mainly open air outlets means that the temperature of the lake most likely follows the air temperature relatively closely for a body of water. This suggests that advection fog is unlikely over this lake.

In order to develop a better understanding of the weather that the boat will encounter during the IRSR on June 7 - June 12, 2020, the team looked at the weather data from the area for the past three years. The nearest weather station with a history record long enough was at Portsmouth International Airport (KPSM) in NH. Figure 3, Figure 4, and Figure 5 show plots for the week of the competition and four days to either side for the years of 2017, 2018, and 2019 respectively. The units for the plots are °C for temperature, mm for precipitation levels, and km/h for the wind speeds. The data shows that the

average wind speeds in the area are about 10 kts with temperatures in the 15-20 °C range. There has been significant precipitation at the beginning of the week consistently for the past three years. Furthermore, the 13-day period shown in the figures below had an average of 4.3 days with fog, or approximately 33%. Fog is a major consideration for manual boat control, as it will reduce visibility. However, as noted above, this is unlikely to be advection fog and is more likely radiation fog that dissipates quickly as the air becomes warmer as the sun rises.

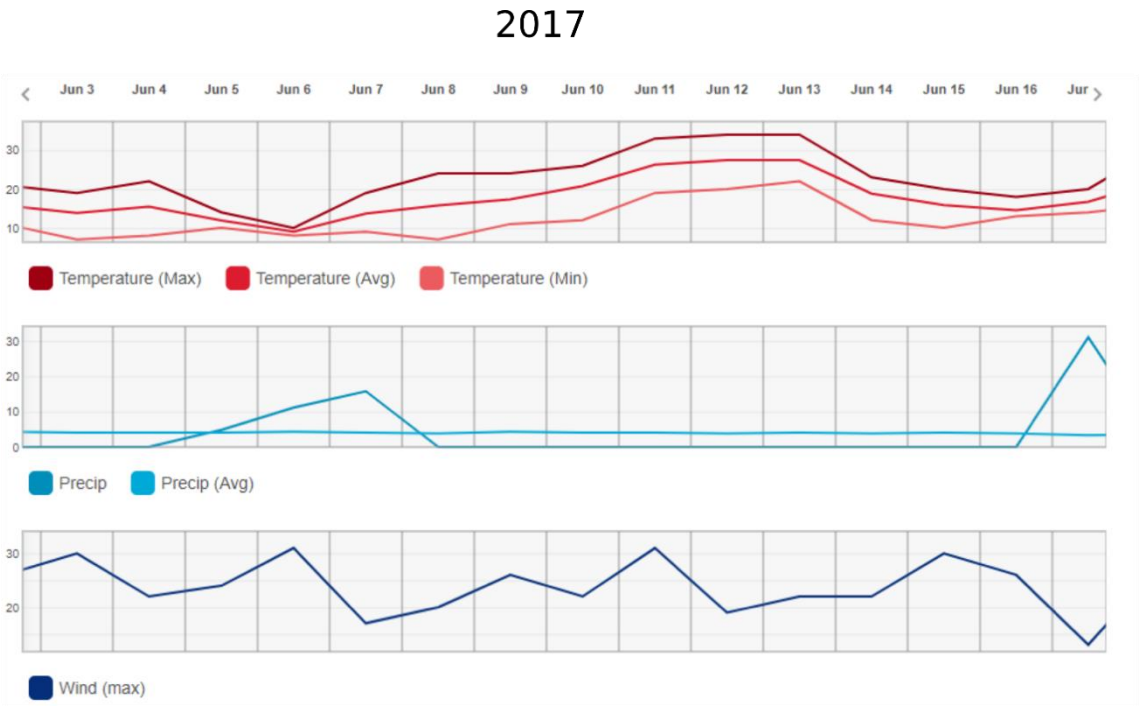

Figure 3: Weather data from KPSM for mid-June 2017 [6].

2018

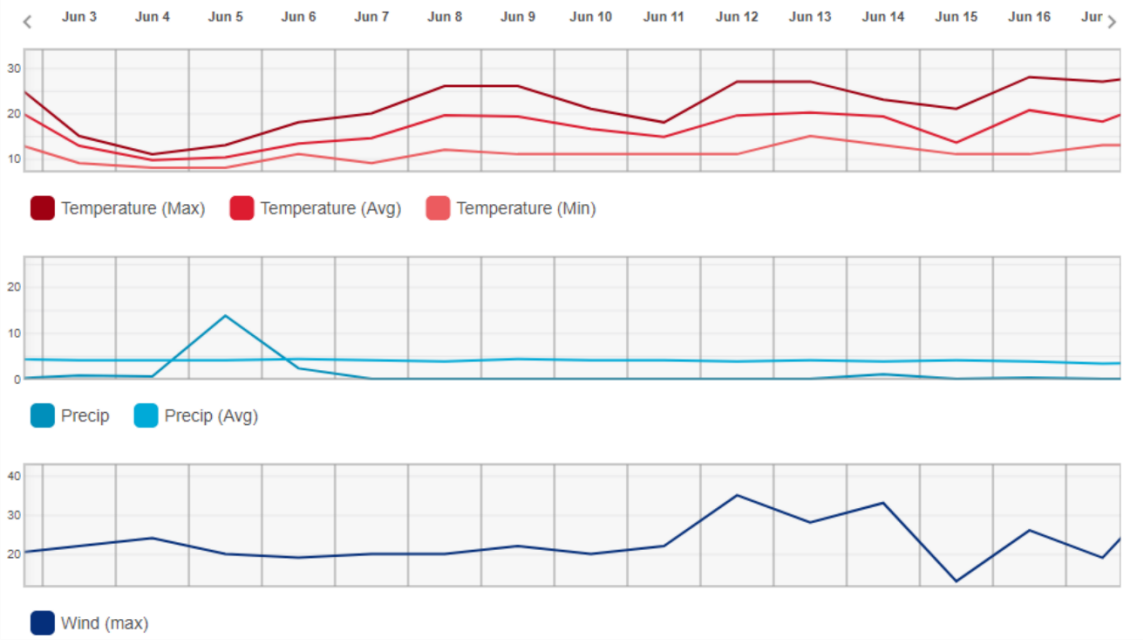

Figure 4: Weather data from KPSM for mid-June 2018 [6].

2019

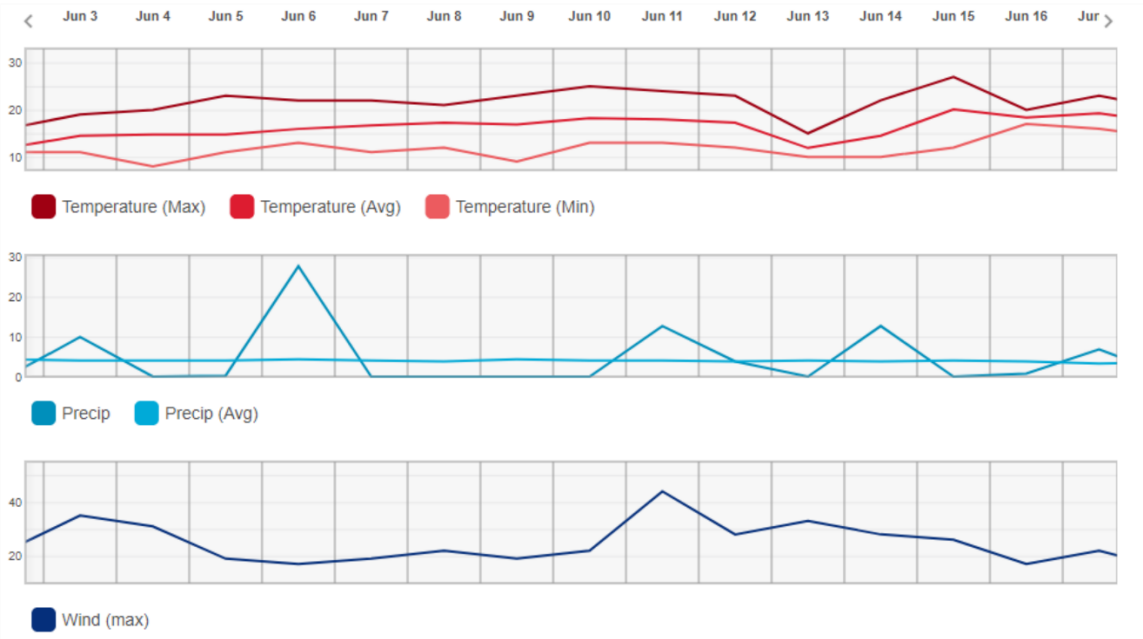

Figure 5: Weather data from KPSM for mid-June 2019 [6].

2.3. Past WPI SailBots

The SailBot project has existed since 2016, making this the project's fourth iteration. Each year experienced successes and challenges. This section presents a brief history of the most important systems, including the state of the project at the start of this project year.

2.3.1. Mechanical Systems

The mechanical system has stayed in large part the same between 2017 and 2019. The major improvements to the mechanical systems last year were mainly to the keel in an attempt to make the boat more balanced, the rigid wing to be lighter and stronger, and the rudders to increase their effectiveness. The overall mechanical system consists of the keel, hull, rudders, and rigid wing sail with a trim tab. There also used to be a winch with a sail, but this year will attempt to steer away from this solution towards a rigid wing. All of the mechanical components were custom made throughout the years of the MQP.

Hull

The 2016-2017 Sailbot team made a hull using a form donated by the US Naval Academy, while the following year made a hull with a custom form inspired by racing yachts. The custom form allowed more consideration for the various mechanical systems, so the 2018-2019 year improved upon the custom form hull instead of making a new one. The biggest change to the hull last year was in the keel cavity. A redesigned keel necessitated a larger and more complex profile. The forces seen by this area make sound construction of this area imperative. However, the structure is accessible only from one side, and the shape of the cavity complicates work in this area with either hands or tools. As a result, this area is hard to maintain and repair, resulting in multiple leaks.

Sails

The first year of SailBot was done as two MQPs. One was working on the sail, while the other was working on the rest of the boat. The MQP working on the sail made a rigid wing with a trim tab that would take the place of a cloth sail. While each year both a cloth and rigid sail were available, the rigid sail either was incompatible with the boat design or failed to meet one or more of the following criteria.

- Must effectively sail under normal wind conditions under all points of sail
- Must have a method to stop generating thrust force
- Must be wirelessly connected to the boat's controls
- Total height from bottom of the keel to the top of the sail must be less than 5 m.
- Must be easily disassembled for transport

The previous year improved upon the design of the sail by making the mast more rigid and reducing the overall weight of the boat. However, they were unable to overcome the communication issues that have plagued the rigid sail for many years. The communication issues forced the previous year to use zip ties to fix the rigid wing sail into position.

Winch system

The 2017-2018 year designed a winch system, which tensioned the main sheet without getting it tangled while slack. This was a response to the issues experienced with the rigid sail. The team last year attempted to address the rigid sail issues described above but were unable to resolve all of them and

therefore ended up using zip ties to hold the rigid wing sail in place. Because of this, the winch system was no longer needed and was removed from the boat.

Keel

The keel was one of the main systems improved upon by last year's team. They moved the keel bulb slightly fore of the previous location because the previous design and position created moments that made the boat sail with the stern deeper in the water than the bow. The newly designed keel also improved on the structural integrity and shape of the keel to decrease drag. The changes to the design of the keel, however, necessitated a slight modification to the keel cavity in the hull. Due to the shape of this area, it is difficult to maintain and repair, making it problematic.

Rudders

The hull made using the US Naval Academy form had one rudder, while the custom form hull has two. The previous year improved upon the two rudders of the custom form hull by making them larger. This increased their effectiveness, allowing the boat more directional control. A single servo has always controlled the two custom form hull rudders, which means that they create some drag. However, analysis in previous years has shown that the drag is negligible, and the simple design decreases the weight and complexity of the tiller system.

Movable ballast

The 2017-2018 year introduced the movable ballast onto the boat. The ballast is a 7.3 kg block of lead cantilevered on a carbon fiber arm. A powerful automotive window motor with a worm gear drives the ballast, but the combination of play in the gearbox and the movement profile of the system allows the ballast to rotate past its design limits. When the ballast over-rotated, the shape of the hull and the weight of the ballast caused the gears to disengage leaving the motor unable to move the ballast back into the operating range, thus locking itself in a damaging position. This causes damage on both the hull and the gears, as shown in Figure 1.

Airmar

The Airmar is a weather station that includes a variety of sensors such as an IMU, GPS, and anemometer. The Airmar is located at the stern, but previous teams have expressed the concern that the sensor sees wind flow that is disturbed by the sail that is in close proximity to the sensor. Previous teams have also raised concerns over whether the sampling rate of the Airmar is sufficient. If these concerns are confirmed, the Airmar will need to be relocated or another wind sensor located in a different location will have to be designed or procured.

2.3.2. Electrical Systems

The previous iteration of this project (2018-2019) introduced significant modifications to the boat's electrical system. The years before 2018 used the NMEA 2000 bus for communications and power distribution, a BeagleBone Black with Ethernet as the central processor, and the Airmar weather station for sensing wind and vessel state information. All electrical components used the NMEA 2000 standard with the notable exception of the motor controllers, which is why the 2016-2017 year developed the SCAMP. The SCAMP translated the motor controllers' CAN messages to NMEA 2000, which is a derivative of CAN. Unfortunately, the implementation details of the overall system were sparsely documented, which impeded the 2018-2019 year team from effectively continuing the progress of previous years.

In order to ease development, they replaced most of the computing elements for ones on which development was easier and faster. As a result, as of the beginning of this project year, the central processor was an Arduino Mega with both Ethernet and USB shields and the movable ballast was controlled by a Talon SRX motor controller connected to a CTRE HERO Development board. The winch was no longer part of the boat since it was not required for the rigid wing sail. The only computational element that stayed the same was the Airmar weather station.

Central processing

Before the previous year, the BeagleBone Black controlled both the winch and the movable ballast and used the Airmar 220WX WeatherStation to measure wind and vessel state information. Since the previous year, an Arduino Mega took the BeagleBone's role as the central processor, but it no longer directly controlled the movable ballast. The Arduino Mega was still supposed to read the data from the Airmar, but it is not clear that it ever accomplished this due to how late in the project the change took place.

Communication to land

As with central processing, the communication scheme to land changed with the previous year. The boat used to connect to shore via two methods. One method was using a 900MHz radio, which provided a long-range, low-data-rate Ethernet connection and allowed the systems to provide telemetry information and software updates. Another method was over Wi-Fi, which was short-range, high-data-rate, and intended for data-intensive tasks, like deploying code. The previous year consolidated the communications to Wi-Fi, which significantly decreased the effective controllable range of the boat.

Microcontrollers and motor controllers

The previous year complicated the microcontroller and motor controller scheme from the years before. The first two iterations of the project used custom motor and micro controllers, which enabled NMEA 2000 and CAN communications throughout the boat. This unified the communications structures and brought them closer to marine standards. However, the degree of custom implementations in conjunction with sparse documentation made it difficult to troubleshoot any issues, which led to the previous year's team decision to switch to a more familiar set of components in the form of a Talon SRX, CTRE HERO board, and an Arduino Mega. However, last year's team was unable to get this hardware and software operational, leaving the boat less functional in this area than in previous years.

On-board displays and cameras

There is an ePaper display mounted in the hull of the boat and a camera on a pole. The 2017-2018 year developed and implemented both. The ePaper display showed boat system status, while the camera connected to a Raspberry Pi Zero with OpenCV running to detect buoys. The previous year did not reimplement either system after the change.

Power

The only source of power for the electronics in the hull are two lead-acid batteries in series. This chemistry offers a favorable cost to capacity ratio, but their current draw and energy capacity ratings are considerably lower than that of lithium–based chemistries and they are much heavier. The peak draw from all of the boat's motors exceeds the lead-acid batteries' ratings, accelerating degradation, and ultimately leading to a sharper drop in battery voltage below levels required to power vital components

2.3.3. Software Systems

The previous year's team spent a substantial amount of time understanding the code from the year prior. This was especially difficult as there was very little documentation available. This code is available on the GitHub organization, but the previous year eventually scrapped it in favor of developing on a different set of boards. Because it took a great amount of time to decide to start from scratch, this year started with essentially templates and pseudocode for the various events of the IRSR.

The previous year started their development on the new boards by attempting to establish communication between the Arduino Mega and the Trim Tab's Teensy 3.5 over Wi-Fi. This was ultimately unsuccessful and led to the decision to zip ties for the rigid sail in competition instead.

3. Purpose and Project Goals

An autonomous sailboat has the potential to be completely self-powered as it uses wind for propulsion and does not need a crew. The only power requirement is for the motors, sensors, and processors on-board, but the power for these systems can be generated using renewable methods, such as solar and wind power, and stored in batteries. This property of a robotic sailboat makes it autonomous to the fullest definition possible, which makes it particularly suited to very-long-term operations in relatively calm waters in areas or tasks where supervision or refueling is difficult or impossible. Examples of such missions include long-term data collection and continuous monitoring of large areas.

The purpose of this MQP was to create an autonomous sailboat that built upon the progress and challenges of previous years. To do this the team developed a series of goals for the project pertaining to keel construction, integration of existing and new sensors, and implementation of new software. These goals reflect the state the team expected to see the project in at the end of this iteration. Goals are broken into two categories. Definite goals are tasks that are highly likely be completed by the end of the year and are a priority for the team. Stretch goals are tasks that are desirable but may be difficult to accomplish within our timeframe and are therefore a lower priority. The following chapters will describe the mechanical, electrical, and software design choices that the team made to complete these goals.

3.1. Mechanical Goals

3.1.1. Definite Goals

Prevent moveable ballast over-rotation

The 2018-2019 SailBot MQP identified the moveable ballast as a part of the boat that would need significant future improvements. This is because the existing design made no effort to prevent the moveable ballast from over-rotating, causing it to lock itself in a damaging position where it could not rotate back under its own power. This led to problems such as damage to the outer hull and significant wear to the main gear of the ballast. Figure 1 earlier in the report shows this damage. Solving this problem will require both hardware, software, and electrical measures to ensure the ballast can operate without damaging the boat.

Re-work keel insert area

Although the keel insert can fit into the keel insert cavity, the bottom portion of the insert is not flush with the hull of the boat like originally intended. Refining this fit to ensure that the keel insert is fully inside the boat will help to ensure maximum stability and support for the keel. Additionally, the keel insert cavity is under particularly high stresses, making it prone to developing leaks. The waterproofing in this area needs to be improved to reduce the chance of leaks occurring. Lastly, the cavity for the old keel insert is still visible and accessible. Since this is no longer in use, it should be sealed.

Redesign trim tab electronics housing

Most of the electronics for the trim tab are in a foam enclosure about 2/3 of the way up the rigid sail. This enclosure is not very durable and has sections that show significant wear. The mountings for the electronics are also not very strong and certain parts have fallen out of place in the past. Lastly, the servo in the enclosure cannot move through its full range of motion due to interference with the foam. Redesigning this area with additional reinforcements and improved part connections will contribute significantly to improving the durability of the trim tab enclosure.

Optimize AirMar location

The AirMar user manual recommends that it be placed at least three feet away from other electronics to prevent interference. Currently the AirMar is located about a foot and a half above the back of the boat, which is where most of the electronics are located. Moving the AirMar to a different location could reduce interference from electronics and could allow it to measure wind with less interference from the rigid wing.

Improve rudder efficiency and durability

The existing rudder/tiller assembly is already a very well-designed system. While a large redesign is not necessary, tweaks in certain areas could significantly improve the durability and smoothness of motion for the system. Additionally, improvements to the rudder finish can be made to further reduce the drag produced while sailing.

Mount additional sensors

Some mechanical, electrical, and software goals will require the addition of new sensors to solve issues with the boat. Areas targeted for additional sensors are the rudders and movable ballast. These sensors will need to be attached in a way that preserves the waterproofing of the boat and ensures the sensors will not break under normal operating conditions.

3.1.2. Stretch Goals

Reduce the weight of the boat

While not a critical concern, reducing the weight of the boat helps in several areas. By reducing weight overall, and centering it as much as possible, there would be several benefits such as improved boat handling, reduced pitching moment, more battery life, and ease of transport.

Refinish boat to fix small damages and chipped paint

Since this is the third year of use for the current SailBot hull, the boat has accumulated several small dings, dents, and scrapes. While these issues are purely cosmetic at this point, they will progress and eventually require maintenance. Additionally, refinishing will improve the visual look of the boat.

Make boat stand more portable

The existing boat stand is very bulky, difficult to move, and not easily disassembled or transported. While this is not an issue while the boat is in the lab, if the team needs to display the boat somewhere outside of driving distance either a new or modified stand will be required.

3.2. Electrical Goals

3.2.1. Definite Goals

Change power source

The lead-acid batteries that the project used for the past three years are approaching their end of life and no longer maintain specified charge or output the specified 12V output. Lead-acid batteries are also particularly heavy per unit charge and offer a narrow nominal output area. A Lithium Ion chemistry can offer a more stable voltage output across the state of charge and a higher capacity at a lower weight.

Document all new electrical components

It is difficult to work with and troubleshoot electronics with large custom undocumented solutions. There are several custom circuits in this project, all without schematics. Creating schematics will enable future teams to start working with the circuits quickly and painlessly.

Add sensing to mechanical systems

There are several mechanical systems are not mechanically constrained and thereby prone to damage. Namely, the movable ballast and the rudder tiller can both over extend and damage the driving mechanisms, shown in Figure 1 and Figure 18 respectively. Adding limit sensing will protect these systems from damage by enabling feedback in the control algorithms.

3.2.2. Stretch Goals

Convert to NMEA2000

NMEA2000 is a marinized communications protocol based on the CAN bus protocol. It specifies both the message format, as well electrical characteristics of the protocol. Converting to this communications protocol will bring the SailBot more in line with marine industry standards, allowing for easier integration with marine products in the future.

3.3.Software Goals

3.3.1. Definite Goals

Write software for control systems implemented in the previous iteration

Due to the timing of the switch to a different control system, there was very little software written for the control system. Writing software for the control system will enable basic functionality on the boat and enable complicated control work in future years.

Collect data from the AirMar

As mentioned in the Optimize AirMar location section of the Mechanical Goals, the current location for the AirMar is not ideal. As such, it would still be useful to collect data from the AirMar in its current location to determine whether the data from this location is useable. This will also lay the groundwork for collecting data from the unit in its new location, both for testing and use in normal operation.

Integrate new sensors into the code

New sensors will be added to the systems to sense limits of mechanical systems, as mentioned in section Add sensing to mechanical systems of Electrical Goals. These sensors will only be useful if they are integrated into the software to shut off power to the motors.

Make software architecture diagrams

Making software architecture diagrams will serve two purposes. The first is it will serve as an incentive to make systems modular as making the diagrams will force the team to think about the structure of the system. Secondly, the diagrams will be useful for future teams as they will enable a high-level view of the architecture.

Enable communications between hull and trim tab

Due to the construction of the rigid wing sail, it cannot produce power unless controlled by the trim tab. Thus, it is essential to enable communication between the hull, which controls the overall logic of the boat, and the trim tab, which influences the direction of the force vector produced by the rigid wing sail.

Convert from Arduino to BeagleBone Black

The original iteration of this project used a BeagleBone Black as the main control board. After difficulty running this control system and understanding the underlying code, the team taking over decided to switch to a board that they knew how to use. This board was the Arduino Mega. Unfortunately, the Arduino Mega, even with additional shields, is technologically inadequate for some of the tasks that the team had set out to do. Converting back to a BeagleBone Black, an embedded Linux system, will enable much more powerful computing with the capability to run multiple processes at once and to achieve goals that are more involved.

3.3.2. Stretch Goals

Set up navigation based on AirMar and vision

Data provided by the AirMar will enable navigation using methods often used in aircraft, explained in section 7. Combining this data with vision at terminal points for more active navigation will allow the boat to have an accurate conception of its pose, allowing for the integration of more autonomy.

Convert to NMEA2000

This goal is cross-disciplinary, with both electrical and software aspects. The description and purpose of this task is described in Electrical Goals, under the same title.

Introduce autonomy

The SailBot competition include challenges that require autonomy. This functionality used to exist in the control system, but was lost with the switch to the Arduino. Returning this functionality will be essential to success in the competition, but there is a significant amount of groundwork to do beforehand.

4. Methodology

4.1. System Overview

The basic systems of an autonomous sailboat can be broken down into three categories: mechanical, electrical, and software. In simple terms, the components of the mechanical system interface with the environment to produce forces and moments, which enable the boat to move in the water. The electrical system serves as the conduit for signals between the microcontrollers and mechanical actuators, as well as providing an on-board source of power. Lastly, the components of the software system process sensor data to determine vessel state and controls the logic that determines the best course of action based on the vessel state and mission.

4.2. Mechanical Systems

The connection between the keel and the hull is a vital one since it is subject to high stresses caused by the bending moment of the keel. This high stress environment makes this area of the hull especially prone to damage and failure. Additionally, the tolerances between the aluminum insert and the hull cavity are incredibly tight, leading to damage during insertion and difficulty inserting the keel. This allowed the inside of the hull cavity to chip and wear down heavily. Previous MQPs had issues with leaks forming here. This year was no different with a leak developing in this area early into the project.

4.2.1. Keel Insert

The connection between the keel and the hull is a vital one since it is subject to high stresses caused by the bending moment of the keel. This high stress environment makes this area of the hull especially prone to damage and failure. Additionally, the tolerances between the aluminum insert and the hull cavity are incredibly tight, leading to damage during insertion and difficulty inserting the keel. This allowed the inside of the hull cavity to chip and wear down heavily. Previous MQPs had issues with leaks forming here. This year was no different with a leak developing in this area early into the project.

To fix this issue, the keel insert area was resealed using epoxy resin. However, in order to ensure that the new layer of epoxy did not interfere with the keel insert, a model of the keel insert maintained space in the hull during the curing process. This polycarbonate plastic model, shown in Figure 6, was laser cut into the shape of the insert and then filed down on the sides down to match the keel insert's taper. Polycarbonate was chosen as the material for the model because epoxy does not bond to it as well as it does to aluminum, which is the material of the real keel insert. This allowed for the removal of the insert without damaging the new layer of epoxy.

Figure 6: Polycarbonate insert for sealing the keel insert cavity

While this method proved mostly successful, the tight tolerances prevented the keel from fitting into the newly sealed cavity as well as it used to. To solve this, the team sanded the new epoxy layer in areas where it was interfering with keel insertion. The team also sanded the sides of the keel insert slightly, as well as the edges at the top of the insert so that the tip of the keel was less likely to tear chunks of material out of the hull when it did not slide perfectly into the cavity. Although this succeeded in allowing the keel insert to fit, it re-opened the leak in the hull. This necessitated a change in strategy. Potential leak locations on the outside of the boat were identified and were sealed using small, localized amounts of epoxy. Then on the inside of the boat several layers of epoxy were spread on the area around the cavity, as shown in Figure 7. This area was discovered to have a significant number of scratches and damages, which could have been the cause of the leak. Since the keel is not inserted on this side of the cavity large amounts of epoxy could be added without affecting the fitment. This method worked and water testing proved that the leak was sealed. By carefully filing the epoxy and sanding the aluminum insert, the keel could be inserted into the cavity the same amount that it could at the start of the project.

Figure 7: Epoxy on the interior side of the keel insert cavity.

4.2.2. Movable Ballast

Due to the amount of power and momentum involved in the moveable ballast, one safety measure to prevent over-rotation would not suffice. Instead, it was decided to use physical hard stops in combination with electronic sensors that cut power to the ballast motor when it is approaching the end of its designed rotation limits.

Store-bought angle brackets were initially meant to serve as hard stops as they are easy to obtain, approximately the right size, and cheap. However, impact tests showed that these angle brackets could not withstand an impact from a small weight at moderate speed, never mind the heavy moveable ballast. Therefore, it was determined that machining our own hard stops would provide more strength and flexibility with the design. A rough hard stop was machined out and it was mounted on a thin piece of plywood to simulate being attached to the hull. This showed that hull was likely not sturdy enough to have the hard stop mounted directly to it. To solve this issue a backing plate was added to the design to increase strength and spread out the force of impact over a larger area of the hull.

After initial testing, a final hard stop design was to be machined, but during the preparation process, it was discovered that as designed, the hard stop was very difficult to fixture in the milling machine. From the advice of one of the lab machinists, the team searched online for 8020 mounting brackets. The triangular design of these brackets, along with their size, made them a perfect fit. These brackets are also significant stronger than any kind of store-bought angle bracket. The brackets are mounted to the boat using two ¼ inch bolts and reinforced with an aluminum backing plate on the inside. However, this did not provide any waterproofing for the connection through the hull. Since rubber gaskets would have provided too much wobble, the team instead secured the bracket using marine adhesive. This provided both the necessary waterproofing and increased the strength of the connection to the hull. In order to

soften the impact, rubber was added to the front of the bracket, where the movable ballast would make contact should the magnetic sensors and absolute position sensing fail. The hard stop mounting solution with the rubber is shown in Figure 8.

However, hitting the hard stops would still cause damage to the gears since the motor gearing provides high torque that would exert significant forces on the gears. Therefore, some type of sensor and software controls would be needed to prevent damage. The team decided that Hall-Effect magnetic sensors were the best choice since they allowed the sensor to be protected from water on the inside of the boat while the sensing magnet could be attached to the moveable ballast. The sensors were positioned just in front of the hard stops to sense when the movable ballast is about to hit the hard stops. These sensors are hot glued onto the interior side of the plastic window covering the gearbox, as shown by Figure 9. Despite the plastic being relatively thin and not appreciably affecting magnetic fields, only a strong magnet can generate a strong enough magnetic field to trigger the sensors at a distance greater than the thickness of the plastic. For this purpose, a strong small neodymium magnet was chosen to ride on the movable ballast arm close to the plastic gearbox cover. The magnet is mounted using a 3D printed 2-piece housing that is mounted to the arm using a thru-bolt.

Figure 8: Hard stops mounted on the boat with rubber clamped down during adhesive drying.

However, hitting the hard stops would still cause damage to the gears since the motor gearing provides high torque that would exert significant forces on the gears. Therefore, some type of sensor and software controls would be needed to prevent damage. The team decided that Hall-Effect magnetic sensors were the best choice since they allowed the sensor to be protected from water on the inside of the boat while the sensing magnet could be attached to the moveable ballast. The sensors were positioned just in front of the hard stops to sense when the movable ballast is about to hit the hard stops. These sensors are hot glued onto the interior side of the plastic window covering the gearbox, as shown by Figure 9. Despite the plastic being relatively thin and not appreciably affecting magnetic fields, only a strong magnet can generate a strong enough magnetic field to trigger the sensors at a distance greater than the thickness of the plastic. For this purpose, a strong small neodymium magnet was chosen to ride on the movable ballast arm close to the plastic gearbox cover. The magnet is mounted using a 3D printed 2-piece housing that is mounted to the arm using a thru-bolt.

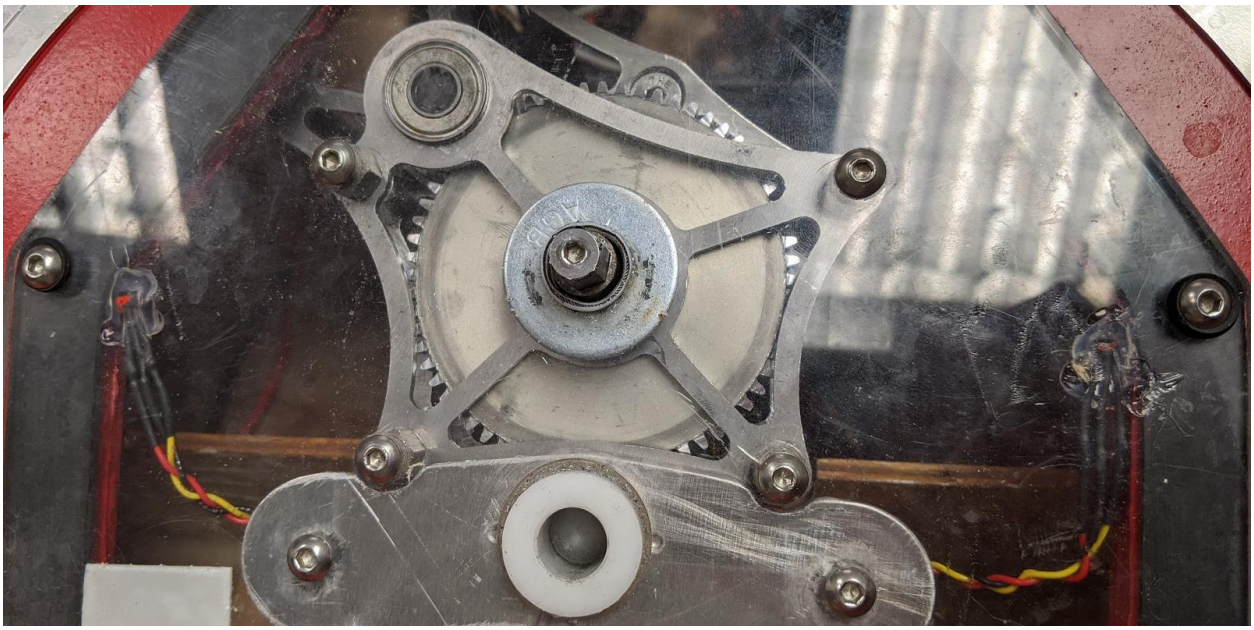

Figure 9: Magnetic sensors mounted at the end of designed travel for the movable ballast.

Another component of the movable ballast system that was contributing to the over-rotation of the movable ballast was an external gear that was part of an assembly connecting the movable ballast to a gearbox on the internal side of the plastic. An overhead view of the gearbox can be seen in Figure 9 above. The external gear was able to physically rotate slightly because the shaft that the gear was connected to was machined with loose tolerances, allowing the rotation around the gear shaft. A similar issue was found with a large internal gear in the gearbox, which was connected along the same gear shaft. The large gear, due to loose tolerances, was able to rotate slightly along the axis perpendicular to the shaft, which allowed it to touch parts of the gearbox frame and cause significant wear and damage on the gear, particularly the gear teeth. Since the rotation issues of both of these gears had a common cause, the solution was to re-machine the gear shaft with custom tolerances to have a tighter fit. Detailed measurements of the gear shaft and the gears were taken, and a CAD model was made of the gear shaft. During this time, the gear shaft's design was altered to have one end of the shaft be hollowed out. Custom making a new gear shaft made it easier for these upgrades to be made to the gear shaft. The idea was created to add a potentiometer to the gearbox. One end of the potentiometer would be inserted into the hollowed end and fixed into place with Loctite, so that it would rotate with the gear shaft. This allows the potentiometer to measure the rotation of the gear shaft, which can be extended to measure the rotation of the movable ballast. Unfortunately, due to the WPI campus closing, these ideas and designs were unable to be actualized; however, all available documents will be made available to next year's team to continue. Since this process could not be completed, a stress simulation was done in Solidworks on the outer pinion gear, shown in Figure 10. This is the gear that experiences the largest stresses in the movable ballast gear assembly. The simulation showed that under theoretical maximum loading conditions the maximum stress on the gear is 459.3 MPa. The yield strength of the steel used for this gear is 460 MPa, so this simulation shows that the moveable ballast is about as heavy as it can be without causing damage to the gears. To validate this simulation, simplified hand calculations where the gear tooth was treated like a cantilever beam were done. The max stress

calculated this way was 406.3 MPa, which indicates that the value of 459.3 MPa given by the simulation is in the correct ballpark.

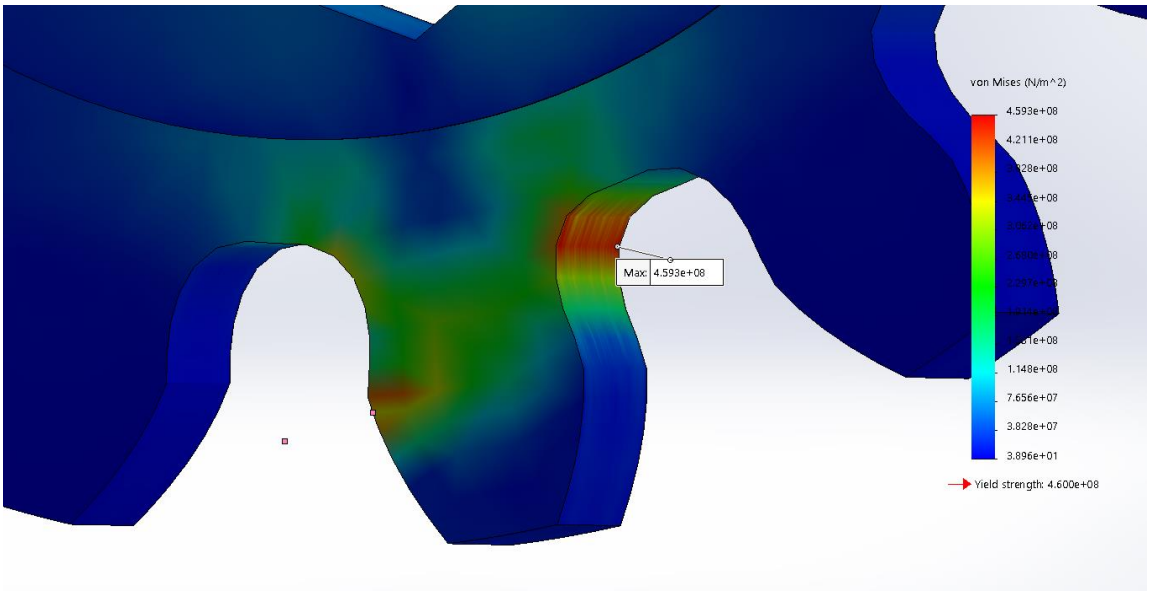

Figure 10: Stress simulation of outer pinion gear

4.2.3. Trim Tab Foam Housing

A polyurethane foam enclosure located at the attachment point of the trim tab to the sail houses the entire electrical system of the rigid wing sail. The existing enclosure suffered from several issues. The trim tab servo and USB port frequently fell out of position when the boat was in use due to poor mounting methods, as shown in Figure 11. The hole on the bottom of the enclosure meant for a rod from the main section of the sail did not have the same reinforcing 3D printed part from the top, allowing damage to the foam around this hole. Additionally, the placement of the servo prevented it from exercising its full range of motion, limiting the trim tab's motion to one side. To solve these problems, a redesigned version of the polyurethane enclosure was constructed.

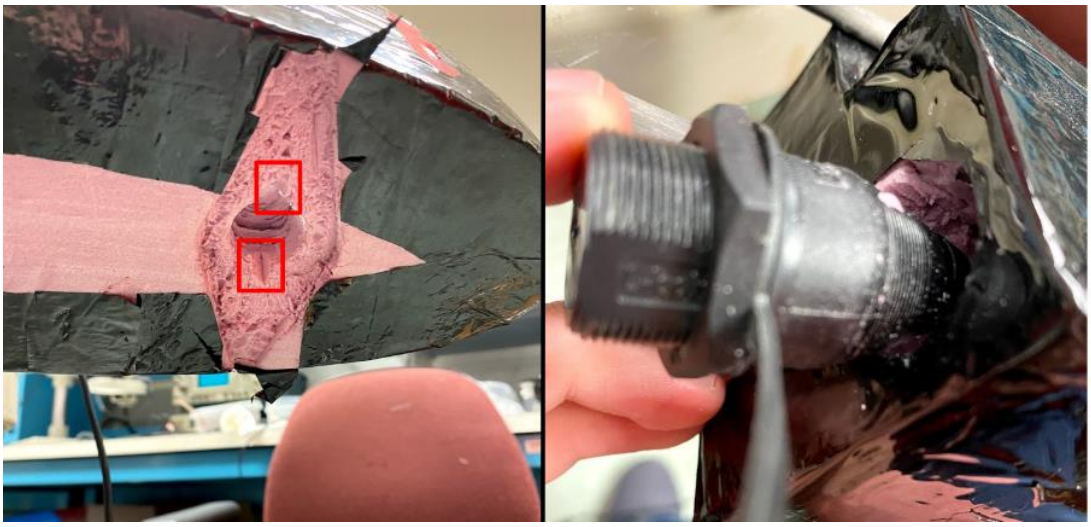

Figure 11: Issues with the trim tab housing. On the left: foam wear. On the right: faulty USB mounting.

To remake the trim tab housing the first major step was to re-cut the foam. The old housing was made up of three layers of 1” insulation foam each cut to different shapes and with electronics embedded in some of them. To get the proper shapes, previous SailBot MQPs used laser cut plywood as a template, as shown in Figure 12. Since the shape of the layers would remain generally the same under the new design, the old plywood templates were used to cut copies of the old design. These copies are also shown in Figure 12. These layers were then modified by hand to create the new redesigned shape. All foam cutting in this stage was done using a hot wire cutter in the lab.

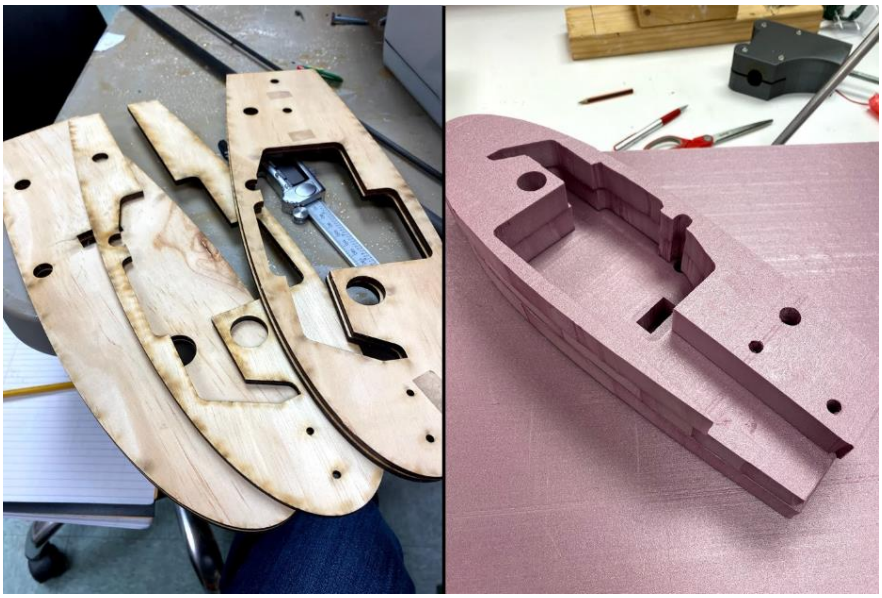

Figure 12: The existing foam cutting templates and the newly cut foam before further modifications.

The most important modification done to the foam layers was in relation to the placement of the trim tab servo. As previously mentioned, the servo crank suffered from reduced range of motion issues that prevented the trim tab from turning equally to both sides. This was because the crank was running into the electronics enclosure inside the trim tab housing when turning one way and the foam when turning

the other way. Moving this electronics enclosure forward out of the path of the servo crank was not possible since it would have caused the enclosure to interfere with the space for the rigid sail rod. Therefore, the only way to solve this issue was to move the servo farther back in the housing. As a result, the slot for the servo on the bottom layer of foam was moved back by 0.2 in. However, this now brought the servo crank and trim tab rod in contact with the foam at the back end of the housing. To compensate for this, two diagonal cuts, shown in Figure 13, were made in the top layer of foam to ensure the crank and rod could move unimpeded. After this the only thing left to do was to adjust the dimensions of the foam around the electronics enclosure. This was necessary because the space where the enclosure fits into the housing was slightly too small in the wood templates. It is unclear why the templates had these incorrect dimensions, but it was extremely easy to just slightly increase the size of the space to allow the electronics enclosure to fit snugly into the housing.

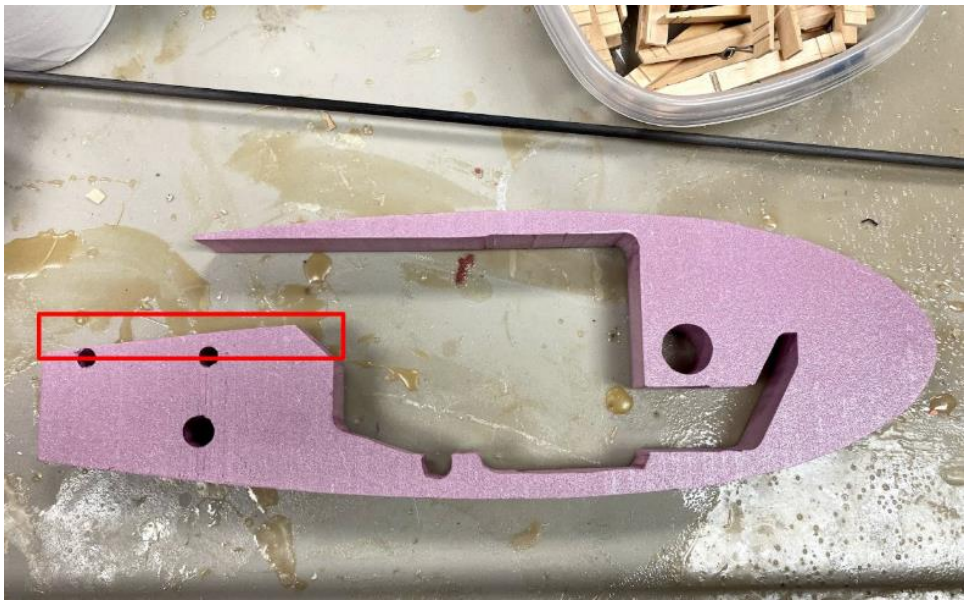

Figure 13: Alterations to the top foam layer.

After the major modifications to the foam had been completed, the next step was to hand cut anything that could not be hot wire cut. This was primarily required to make spaces for electronics and wires to pass through certain areas of foam. A hole was drilled in the front of the middle layer, shown on the left in Figure 14. This is where the wind direction encoder rod is attached. A rectangular cutout was cut on the side of the top and middle layers of foam, shown in the middle of Figure 14. This is for a circuit board with LEDs and a magnetic on/off switch. Additionally, the top and middle layers had a circular hole cut in the back of them, shown on the right in Figure 14. This hole was for the USB charge port. Lastly, a ring was cut out on the bottom side of the bottom layer to allow for a 3D printed part that will be explained in the next paragraph.

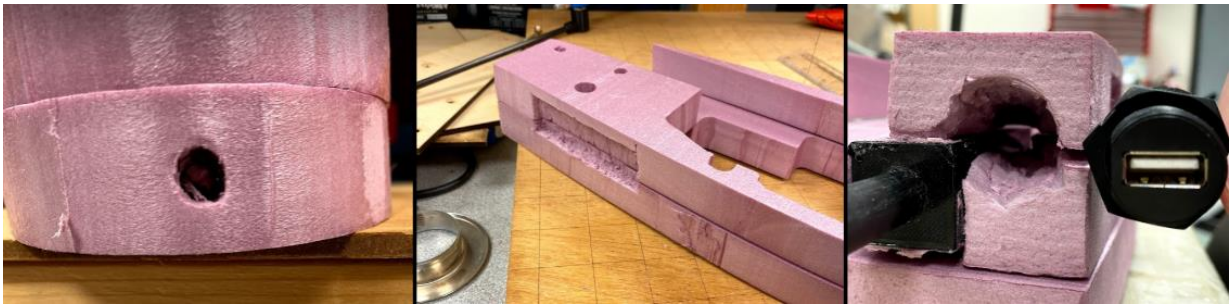

Figure 14: Hand cut sections of foam for electronics.

Certain areas of the housing needed additional support or wear protection, so 3D printed parts were used to solve these issues. As previously mentioned, there was a wear issue with the foam on the bottom of the trim tab. This was primarily caused by small protrusions on the rod that connects the rigid sail to the trim tab housing. Unlike the top of the housing, which already had a 3D printed part protecting this area, there was nothing on the bottom to stop these protrusions from digging into the foam. To solve this issue, a ring-shaped insert with slots on each side was 3D printed to reinforce the connection area, shown in Figure 15. The slots on the side provide space for the protrusions, preventing them from causing damage.

Figure 15: 3D printed part for the bottom of the trim tab housing drying after being glued.

Another area where there was a need for a new part was the mounting of the trim tab servo. Under the old design, the only support for the servo was from a small collar that the servo screwed into. This part was not designed to provide support but instead was simply made to place the servo at the right height for the trim tab rod linkage. Additional stresses placed on the servo during transportation and attachment to the rigid sail, as well repeated stresses placed on it during normal operation made the servo mounting prone to failing. To solve this problem, the servo mounting was redesigned to have a thicker collar and additional flaps to either side, shown in Figure 16. The collar acts the same way in this new design, providing a place for the servo to screw into and acting as a riser to put the servo crank on the correct plane. The flaps on either side add stability in two key ways. First, they provide a wider base for the servo, giving it better resistance against tipping and wobbling. Second, they increase the surface area of the servo mount by 1.5 to 2 inches. This allows for a much stronger adhesive connection between the mount and the foam, making it significantly more difficult for the servo to fall out of place.

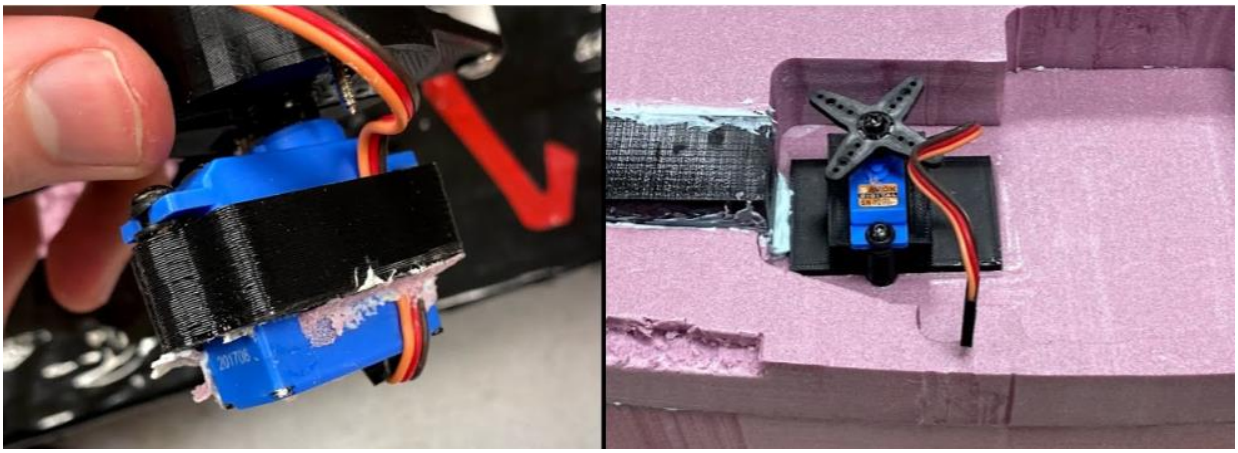

Figure 16: On the left: the old servo mount. On the right: the new servo mount with additional support.

Since there were no major issues with the rest of the electronics for the trim tab, the remainder of the parts for the new trim tab housing were transferred over from the old housing. The layers of foam were glued together using foam adhesive. The USB plug and the trim tab were placed into the foam in the same manner. The 3D printed parts and the wind direction encoder rod were fixtured in place using superglue. Lastly, the circuit board with the LEDs and the magnetic on/off switch was attached using hot glue. After all the components had been assembled the trim tab housing was coated with a black tape layer to provide a clean exterior finish and to protect the foam and electronics inside the housing.

4.2.4. Rudders and Tiller Mechanism

While the existing rudders worked well, the finish on them was not ideal for minimizing drag. The original surface had slight ridges from the fiberglass fibers and bubbles that remained in the epoxy from the mixing process. A smooth surface would reduce drag. To solve this problem the team coated the rudders with two layers of epoxy, as shown in Figure 17. After each layer, the epoxy was sanded down to smooth out the surface finish. After the last layer, the rudders were wet sanded with the finest grit sandpaper available in order to get the surface to be as smooth as possible. The rudders were then washed with water to get the epoxy dust off and leave a smooth surface. To smooth the surface even further, they can be buffed with marine wax.

Refinishing the rudders also brought to light an existing issue with the rudders. There was a slight difference between the two rudders that caused one of the rudders to scrape against the hull of the boat if it was placed in the port side rudder position. This meant that this rudder could only be placed in the starboard side, essentially creating a “port” and a “starboard” rudder. This was undesirable since it was easy to insert the rudders on the wrong sides, which would cause damage to the hull. This issue came from the area near the metal rod at the end of the rudder that was inserted into the hull. For the properly fitted rudder only the metal rod would contact the hull. However, for the problematic rudder both the metal rod and the adjacent section of the rudder would contact the hull. To solve this the adjacent section of the rudder had to be sanded down so that it no longer touched the outer hull. This fix now enables the rudders to be used interchangeably on either side.

Figure 17: Rudders drying after the first epoxy coating

Refinishing the rudders also brought to light an existing issue with the rudders. There was a slight difference between the two rudders that caused one of the rudders to scrape against the hull of the boat if it was placed in the port side rudder position. This meant that this rudder could only be placed in the starboard side, essentially creating a “port” and a “starboard” rudder. This was undesirable since it was easy to insert the rudders on the wrong sides, which would cause damage to the hull. This issue came from the area near the metal rod at the end of the rudder that was inserted into the hull. For the properly fitted rudder only the metal rod would contact the hull. However, for the problematic rudder both the metal rod and the adjacent section of the rudder would contact the hull. To solve this the adjacent section of the rudder had to be sanded down so that it no longer touched the outer hull. This fix now enables the rudders to be used interchangeably on either side.

Another issue with the rudders was that the servo horn that connects to the servo had become stripped, as shown in Figure 18. This was likely due to the screw holding it in place becoming loose and then retightened after the horn had been reseated in a different position. Additionally, the rudder tiller has no feedback, and the servo has a wider range of motion than the tiller, allowing it to over extend the tiller. A new identical servo horn was ordered to replace the stripped one and was fixtured using Loctite. The screw holding the horn in place also had Loctite added to it. Lastly, a support ring was added at the top of the servo horn to minimize side to side forces.

Figure 18: Stripped servo horn

4.3. Electrical Systems

The electrical system experienced some major changes this year. Figure 19 shows the electrical and logical connections as they existed as of this writing. The primary changes were in how the connection to land is done, the addition of new sensors, and the change in the microprocessor complement in the hull. The changes are described in more detail in the following subsections.

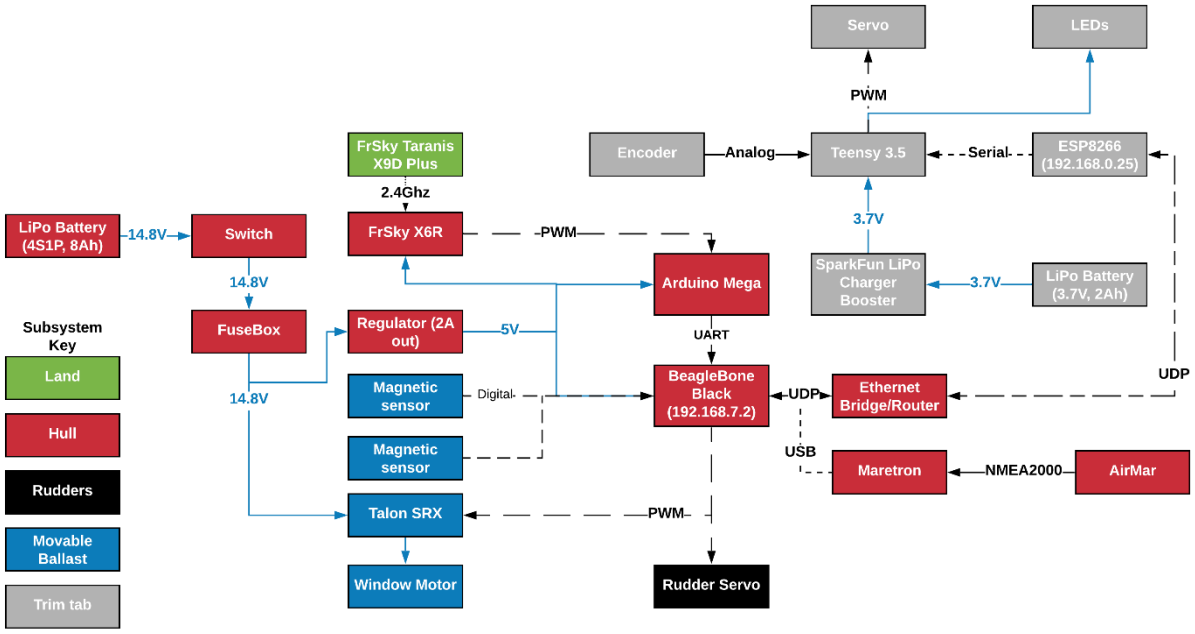

Figure 19: Electrical diagram of this iteration.

4.3.1. Land Connection

The land connection was switched to 2.4GHz using an RC controller meant for model airplanes. This was done to simplify the connection and extend the range of controllability of the boat from shore. While the manufacturer does not list the range on this unit, it is reported to be approximately 1.5 nautical miles, which is significantly longer than allowed by the Wi-Fi used previously [7]. Wi-Fi is still used to disseminate diagnostic information and launch the boat program, but not for RC operation or to switch between autonomous and tele-operated operation.

4.3.2. BeagleBone Black (BBB)

Despite trying to work with the Arduino Mega implemented last year, it was deemed technologically inadequate for all of the design requirements established by this project. Namely, it is impossible to read the Maretron or the Airmar without substantial efforts that would otherwise be directed towards addressing other areas of the boat. As such, the team decided to switch the Arduino to a BeagleBone Black (BBB). Because the Arduino and BBB have the same power requirements and input jacks, very little work was required electrically to make the switch.

Unfortunately, the Arduino could not be completely phased out. The BBB has extremely limited PWM capability, which is required for reading the RC receiver. The board does have PWM input capability, but there are only 4 pins, 2 or which are used for system functions. This is not favorable considering the planned complexity of the system will necessitate the use of at least 4 PWM inputs, and perhaps up to the 6 PWM inputs implemented by the RC receiver. The BBB does support GPIO interrupts, which is how the PWM was read using the Arduino, but upon trying this, it was discovered that the frequencies employed by PWM are higher than those supported by GPIO interrupts on the BBB, so this approach was scrapped and Arduino was used as an intermediary between the BBB and the RC receiver.

Because the BBB and the Arduino Mega use different pin voltage levels, establishing communication between the two boards was not trivial and required a logic level shifter, similar to the Sparkfun BOB-12009. Since the shifter circuit is simple and all of the components were available on hand, an attempt was made at prototyping the shifter in order to validate this solution and avoid the delay and cost associated with purchasing a converter. Unexpectedly, this circuit did not operate as intended and it was modified by trial and error until it appeared to perform. This circuit is shown in Figure 20. The MOSFET used was the IRF520N supplied in the ECE2010 kit.

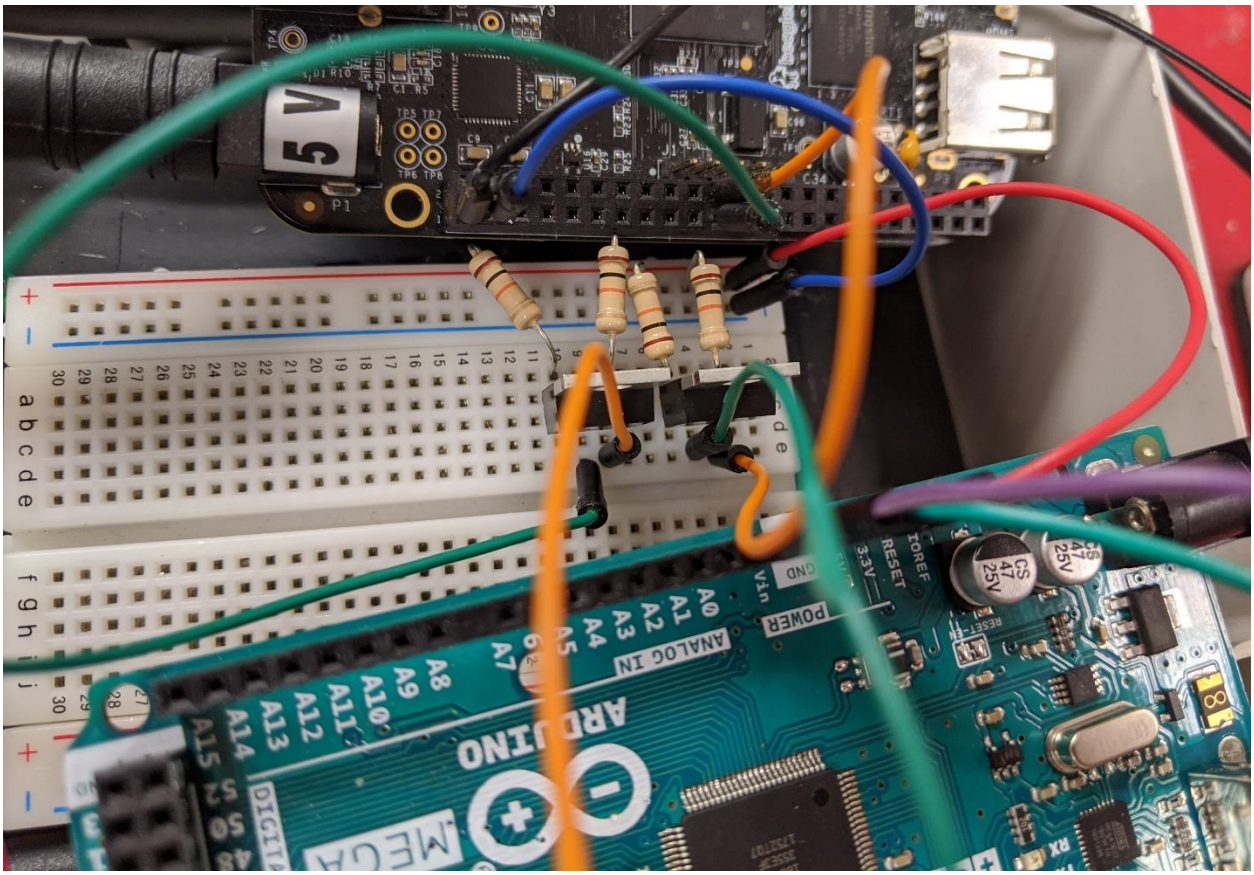

Figure 20: Prototyped logic level shifter.

While this circuit performed very well in small scale tests, a full system test revealed an unacceptably large number of messages being dropped between the two boards when using the UART protocol. This may have been caused by a variety of issues. One is the fact that the circuit was modified from what is typically used for this purpose. Eventually the reason that the standard circuit did not work was discovered. The data sheet for the IRF520N does not explicitly call out which pins represent which outputs of the MOSFET, so an incorrect assumption was made that the pins reflected the component schematic pictured in the data sheet. In practice, however, the pins represent gate, drain, and source from left to right as pictured in Figure 20. Another reason for the dropped messages may have been that the full system tests left less processing time to communication than the small scale tests, leading to packets getting cut off as they were being transferred over the UART protocol. Because of these results, the communication protocol was shifted to SPI and a COTS logic level shifter was purchased as too few MOSFETs were on hand to extend the prototyped circuit to 4 lines.

Since SPI can be implemented as a full-duplex communication protocol and the pinout confusion had not been resolved at that point, SPI seemed like a promising solution to the packet drops. However, due to an issue encountered when programming the board to use the protocol, it was decided to switch back to UART and instead resort to unidirectional control. Due to the time constraints of the project, this seemed like the best solution because UART was demonstrated to work in the small scale and the BBB is capable of outputting PWM, though only on 2 ports which is just enough to control the rudders and the movable ballast. Given that the overall design of the boat is not expected to change in the coming years, using all of the PWM output ports on the BBB is not expected to be as large of a concern as it was with the PWM input ports. The final IO utilization on the BBB is shown in Table 1.

Table 1: Description of used pins on BeagleBone Black.

Device	Port/pin number	Description
Maretron	USB port	Outputs NMEA0183 sentences, used to convert from NMEA2000 of the AirMar.
Ethernet router	Ethernet port	Outputs Protobuf messages to be sent over UDP to the Teensy 3.5 in the Trim Tab.
Arduino Mega TX	P9_22	UART pin receiving from Arduino's TX pin.
Arduino Mega RX	P9_21	UART pin transmitting to Arduino's RX pin.
Port Hall Effect Sensor	P9_23	GPIO pin used to read the state of the port hall effect sensor.
Starboard Hall Effect Sensor	P9_15	GPIO pin used to read the state of the starboard hall effect sensor.
TalonSRX	P9_14	PWM output pin controlling the TalonSRX.
Rudder Servo	P9_16	PWM output pin controlling the rudder servo.
Reserved for Movable Ballast Potentiometer	P9_27	Analog input pin reading the value of the potentiometer planned to replace the magnetic encoder on the movable ballast.

4.3.3. CTRE HERO

The movable ballast is controlled by a Talon SRX motor controller. The TalonSRX used to be connected over CAN bus to a Cross the Road Electronics HERO Board. Due to difficulties with software development for the HERO board and upon discovery that the Talon SRX motor controller can use PWM, the HERO board was phased out of the implementation and the motor controller was connected directly to the RC receiver as a test. Electrically, this was not a very significant change because the TalonSRX sources power from separate power input and the HERO was not instrumental in powering any other systems. The intention was to connect the TalonSRX to the BBB as one would connect a servo, but this was not tested due to the restricted access to campus implemented in D-term in response to the coronavirus outbreak. This was, however, tested successfully on the small scale with a small servo taking the place of the TalonSRX, as shown in Figure 21. Rudder control was tested in a similar fashion by changing which pin was generating PWM for the servo (this is the unconnected orange wire in Figure 21).

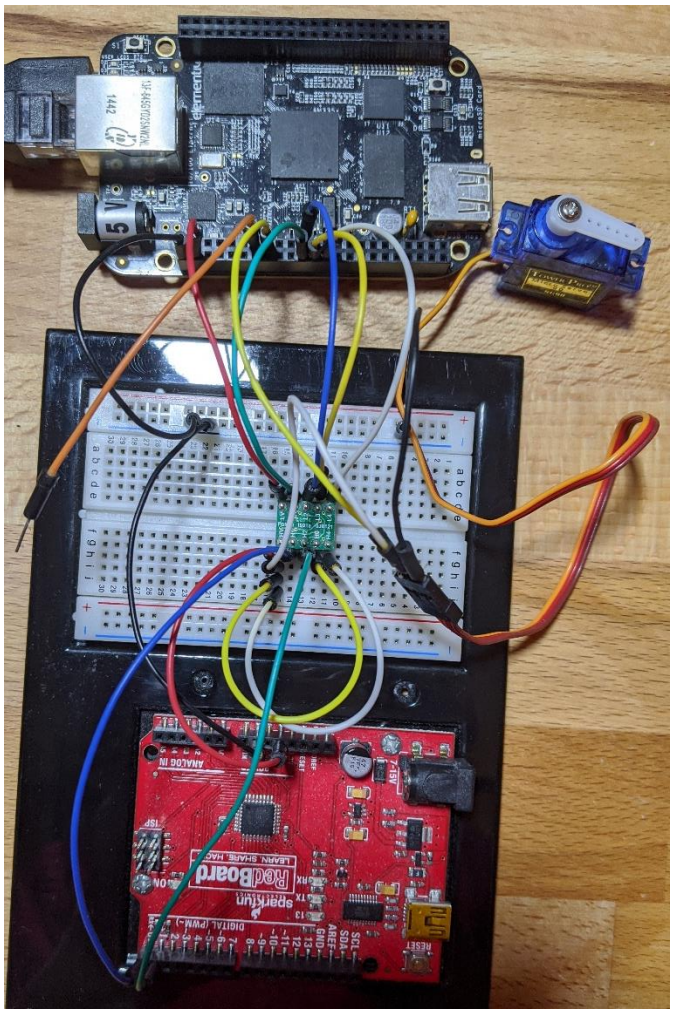

Figure 21: Small-scale testing setup.

4.3.4. Teensy 3.5

The Teensy 3.5 serves as the sole microcontroller for the Rigid Wing Sail's Trim Tab. The Teensy reads wind data from the encoder mounted on a pole extending fore of the sail, communicates with the BBB in the hull over Wi-Fi using an ESP8266 chip, and controls the Trim Tab servo. This system is largely unchanged from previous years with the exception of the connections for the wind-sensing encoder. The encoder sourced power from the vIn power pin, which changed its range of output depending on the charge of the battery. Moving the power to a stable 3.3V power pin resolved this issue. Additionally, the encoder's data pin returned spurious readings because it was floating. Pulling this pin to ground returned the readings to normal. Table 2 shows the pins connected to various devices.

Table 2: Description of used pins on Teensy 3.5.

Device	Pin number	Description
Encoder	38	Analog connection reading wind direction.
Servo	6	Controls servo position.
LED1	7	Direct connection showing mode. Works in conjunction with LED2.
LED2	8	Direct connection showing mode. Works in conjunction with LED1.

Wi-Fi LED	5	Direct connection showing the state of the Wi-Fi connection.
Power LED	4	Direct connection showing the power state.
Onboard LED	13	Direct connection to onboard LED. Not used.
ESP8266	9	RX pin for Serial communication.
ESP8266	10	TX pin for Serial communication.
vIn	16	Direct connection reading battery charge.

The biggest changes to this controller were in its software. Section **Error! Reference source not found.** discusses the software changes in more detail.

4.3.5. Sensors

The boat has multiple sensors that help it to determine its state. These sensors are distributed across systems and each system's microcontroller distributes sensor readings using a Protobuf message structure.

Apparent wind direction sensor

There is a single sensor on the Rigid Wing Sail in the form of an encoder located fore of the leading edge on a pole, shown in Figure 22. This sensor measures the apparent wind direction, but not its magnitude.

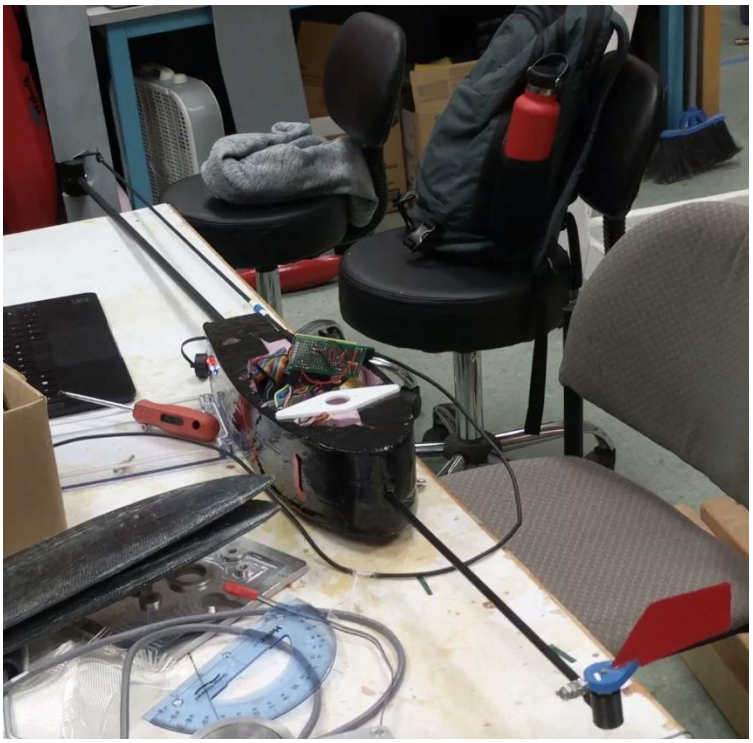

Figure 22: Trim tab with its wind sensor mounted on a pole opposite the tab itself.

Airmar Marine Weather Station

The magnitude of the wind, as well as its direction relative to the hull heading, is measured by the Airmar Marine Weather Station, mounted aft of the Rigid Sail on the body of the hull. Figure 25 shows this sensor. There is some concern that this sensor sees air disturbed by the sail thereby returning wind data not representative of that seen by the sail. To address these concerns, a plan was made to test

whether the level of air disturbed was significant enough to warrant moving the Airmar to another location on the boat. The first step was to estimate where the best locations to put the Airmar would be, prioritizing undisturbed access to the surrounding air and wind. The ideal locations were determined to be at the very front of the boat or at the top of the rigid sail. However, the latter location was quickly dismissed because attaching the Airmar at the top of the rigid sail would add a significant amount of weight to the lightest system on the boat, defeating the purpose of weight reduction of the sail. The second step was to test the Airmar's readings while outside, temporarily attaching it to pre-determined locations on the boat, then to compare with the data from the original location. The next step depended on the results from this test; if the disturbance of air was not deemed significant, then the Airmar would remain in its current position with sound reasoning as to why it is there. If it was significant, then the Airmar's best location would be determined from the experimental data and a new mount would be designed and manufactured for re-locating the Airmar.

However, much more time than expected was required to establish sound control over the boat. In addition, with the campus closures, this plan was unable to be executed. To work around this, the plan was altered to simulate air flow around the rigid wing in Ansys. Figure 23 shows the model of the rigid wing, a Joukowski airfoil, in Ansys. Significant progress was made to get the simulation complete, but due to lack of experience using Ansys and other required software and some technical difficulties, a robust air flow simulation was unable to be completed.

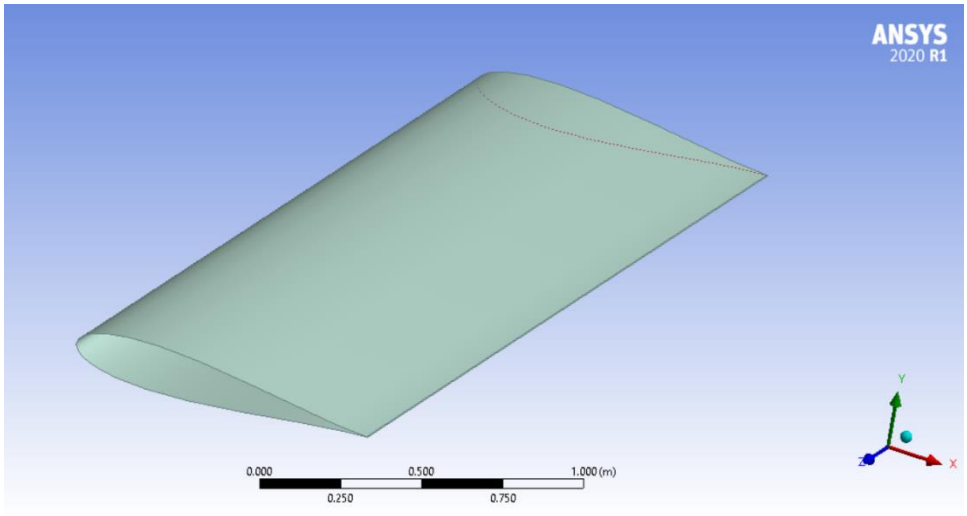

Figure 23: Rigid wing model in Ansys.

Figure 24: Boat with rudders and Airmar mounted. Note the lack of a winch provision in the hull.

Figure 25: Boat with rudders and Airmar mounted. Note the lack of a winch provision in the hull.

A Maretron NMEA 2000 to USB converter translates NMEA 2000 messages from the Airmar into USB messages because NMEA 2000 is not a readily available protocol to any of the microcontrollers onboard. Because wind speed and direction is not strictly required for manual sailing, control software did not initially include this sensor. However, it returns enough information to implement inertial navigation algorithms in the future, which would be required for autonomous operation. Some of the concepts and math behind these algorithms are discussed in section 7.

Magnetic digital limit switches

To sense whether the movable ballast has reached either limit of its travel, hall effect sensors were mounted on the inside of the hull. These are pictured in Figure 9. Hall effect sensors output an electrical signal in response to sensed magnetic fields. As such, they allow contactless sensing at short distances. These were the highest priority categories for sensing the moveable ballast because this ensures that the movable ballast cannot break the sensor and there are minimal risks to waterproofing as a result of installing these sensors.

The circuit for these sensors is shown in Figure 26 and in practice, two of these circuits were implemented side by side, as shown in Figure 27, to accommodate port and starboard. The hall effect

sensor sourced for this purpose was an Allegro A3144. This particular component is popular in the hobbyist community because its power and output characteristics require minimal work to set up.

These sensors trip when the South side of a magnet moves across their effective range. The effective range is approximately ¾" directly above the sensing side of the sensor (this is chiseled side on the A3144). In order to provide a visual indication of the sensor's state, LEDs were wired between the output and ground pins of the A3144. The resistor used to limit the current going to the LED is 330Ω because this was on hand in the lab and testing showed that this performed adequately with the LEDs found in the lab. The 10kΩ resistor is a pull up resistor used to prevent floating. While this is not called for in the schematic for this sensor and noise did not appear to be an issue when testing this setup, a 0.1uF smoothing capacitor can be implemented between the output and ground pins of the sensor to filter out any noise coupled with the digital output.

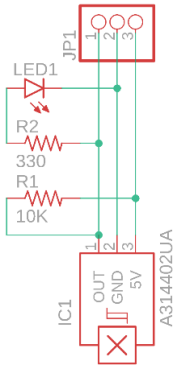

Figure 26: Magnetic sensor circuit based on the A3144 Latching Hall Effect sensor.

Figure 27: Practical implementation of the hall effect sensors. Red indicates 5V pins, blue indicated ground pins, and yellow indicates output pins. The two circuits are side by side and run top to bottom. The top is sensor side, and the bottom is microcontroller side.

4.3.6. Battery

At the start of this year, the batteries used on the boat were two Lead-Acid batteries in series, pictured in Figure 28, resulting in a 12V, 9Ah supply. However, these batteries have degraded over time and no longer provide the necessary voltage to run all systems. This issue was mentioned in this project's report

for last year. However, the battery health has degraded so much that the boat is no longer operational with these batteries.

Figure 28: Lead-Acid battery used previously to power the system on the boat.

Research into the types of batteries that the RC boating community uses showed that the most common battery chemistry used for this purpose is the Lithium Polymer (LiPo). This is ideal because it shows that LiPos are suitable for the marine environment and have been proven capable in robotic applications in the past. Using a lithium-based chemistry brings another advantage in that it has a large nominal area, where the output voltage drops insignificantly with discharge. This is shown in Figure 29 by running simulations using the generic battery models in Simulink.

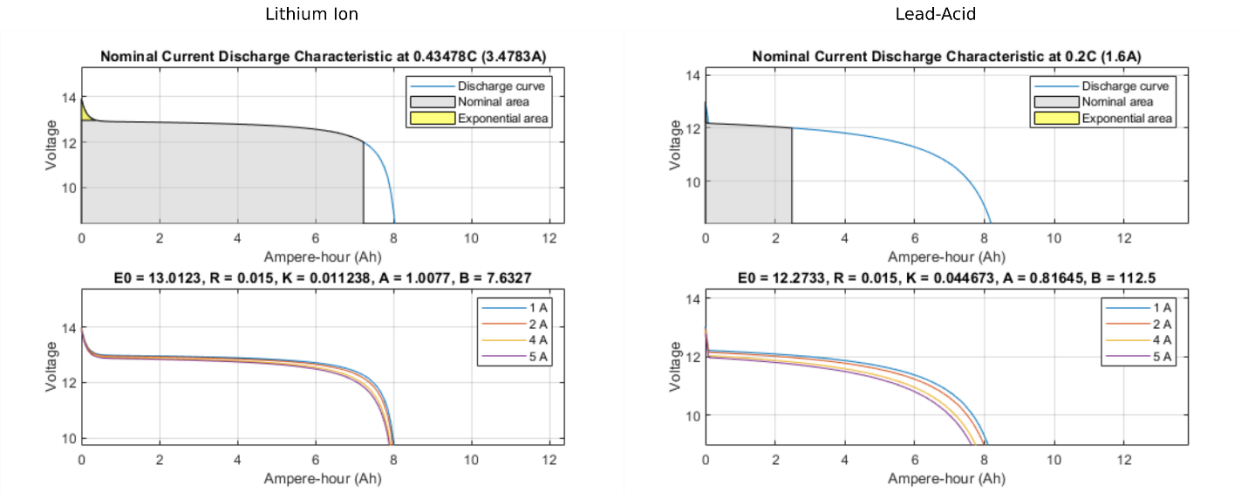

Figure 29: Simulink simulation plots showing discharge characteristics of a generic lithium ion (left) and lead-acid (right) battery with all parameters held the same between models.

In order to maintain a similar voltage to the lead acid battery and avoid overhauling the existing electrical systems, a 4S battery was chosen. The S number on a LiPo battery indicates the number of cells that are in series, while the P number designates the number of cells that are in parallel. However, since most batteries are 1P, the P number is usually omitted from most battery specifications. A LiPo cell's nominal voltage is 3.7V (4.2V fully-charged), making a 4S battery's nominal voltage 14.8V [8].

For testing purposes, a battery found in the lab was used, pictured in Figure 30. The battery was 4S1P, which means that it outputs 14.8V when fully charged. Since this is within the range of the voltage regulator on board, it a close substitute for the lead acid batteries used previously with additional capacity. Initial testing indicated that this is a significant improvement over the lead acid batteries. The maximum capacity of this type of battery available at a reasonable price is 20Ah [9], which would last approximately 6hrs assuming a very conservative peak current draw values of 11A and average current draw 30% of peak, as shown in Table 3.

Table 3: Battery endurance calculation.

Component	**Draw (A)**	**Battery Capacity (Ah)**	20
Movable Ballast motor	5.00	Avg % power draw	30%
Router	3.00	**Duration (hrs)**	**6.11**
Arduino	1.12		
BBB	1.20		
Airmar	0.09		
Maretron	0.15		
Rudder servo	0.25		
Hall effect	0.01		
Transceiver	0.10		

Total	**10.92**		

Most batteries of this type and class come pre-crimped with XT90 connectors, while the boat's battery leads are Anderson PowerPole 35Amp connectors. A converter was created by crimping Anderson PowerPole 45Amp connectors to the leads of an XT90 female connector with tails. The 45Amp connectors were chosen because they are more appropriate for the gauge of the tail wires and they use the same housings as the 35Amp kind, making them interoperable.

Figure 30: LiPo battery found in the lab and used for testing purposes.

4.4.Software Systems

Figure 31 shows the high-level system architecture that was initially targeted. This architecture attempted to make full use of the microcontrollers that were on the boat from previous years while implementing the sensors that last year's team did not address and the sensors that were added this year. This structure seemed advantageous because it built upon the work of previous years, potentially saving this year's team time in reinventing the basics and expediting the process for getting the boat water ready.

Initial high-level software architecture

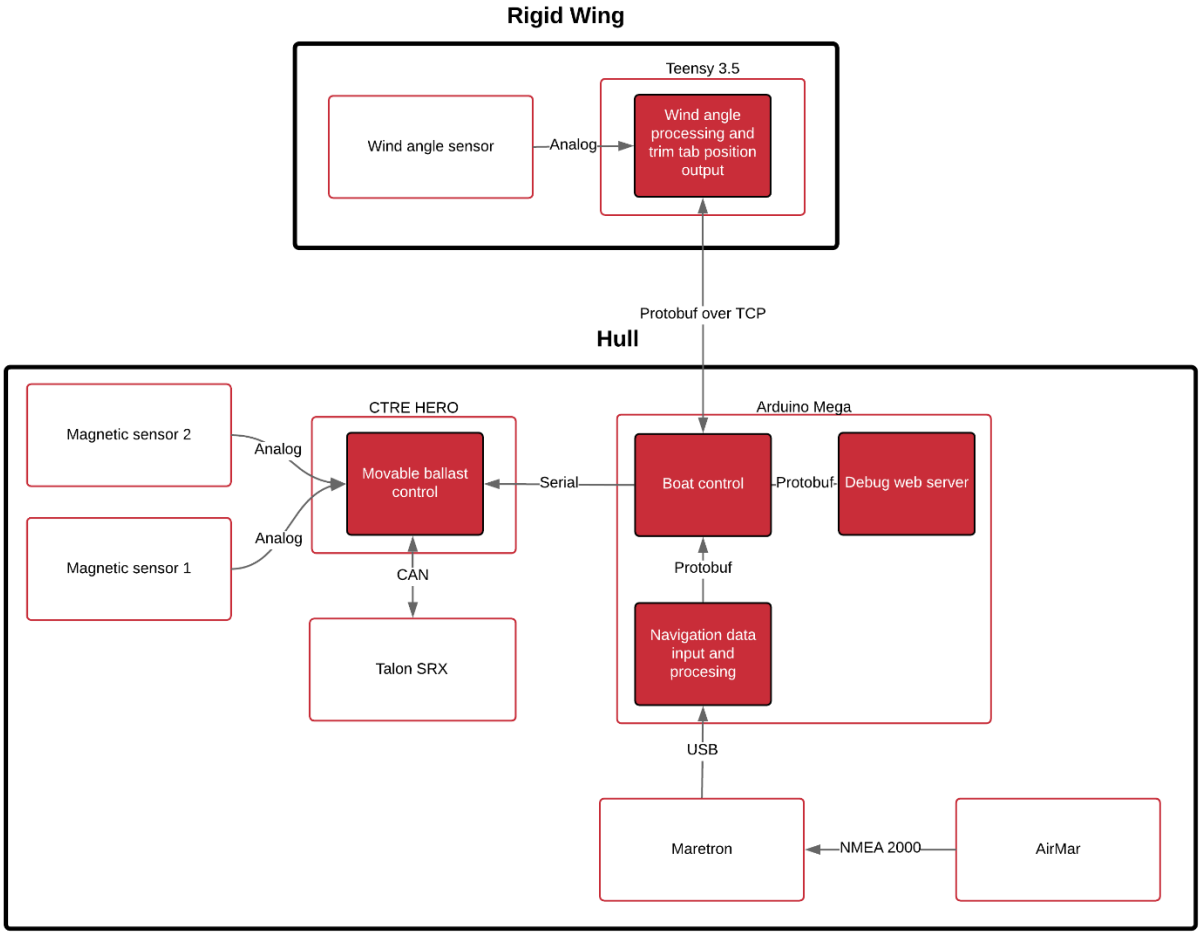

Figure 31: System architecture diagram showing the initial intended design.

Basic items that were required to become water-ready were:

- Reliable communication between the trim tab and the hull's microcontroller
- Reliable communication between the RC controller and the hull's microcontroller
- Reliable control of the rudders

Additional items required for full water-ready functionality were:

- Reliable control over the moveable ballast
- Reliable wind-sensing from trim tab apparent wind direction sensor

OBJ

Of the items listed above, the basic items received top priority in terms of software development. Still, a small portion of time was dedicated to addressing the additional items. During the development process, a number of discrepancies were discovered between the water-readiness goals, which were

continued from previous years, and the capability of the on-board hardware. This necessitated a change to the hardware that was very disruptive to software development and did not allow for software reuse wherever the change occurred. The system architecture that precipitated out of the hardware change is shown in Figure 32. Following is a walkthrough of the system-by-system software design process.

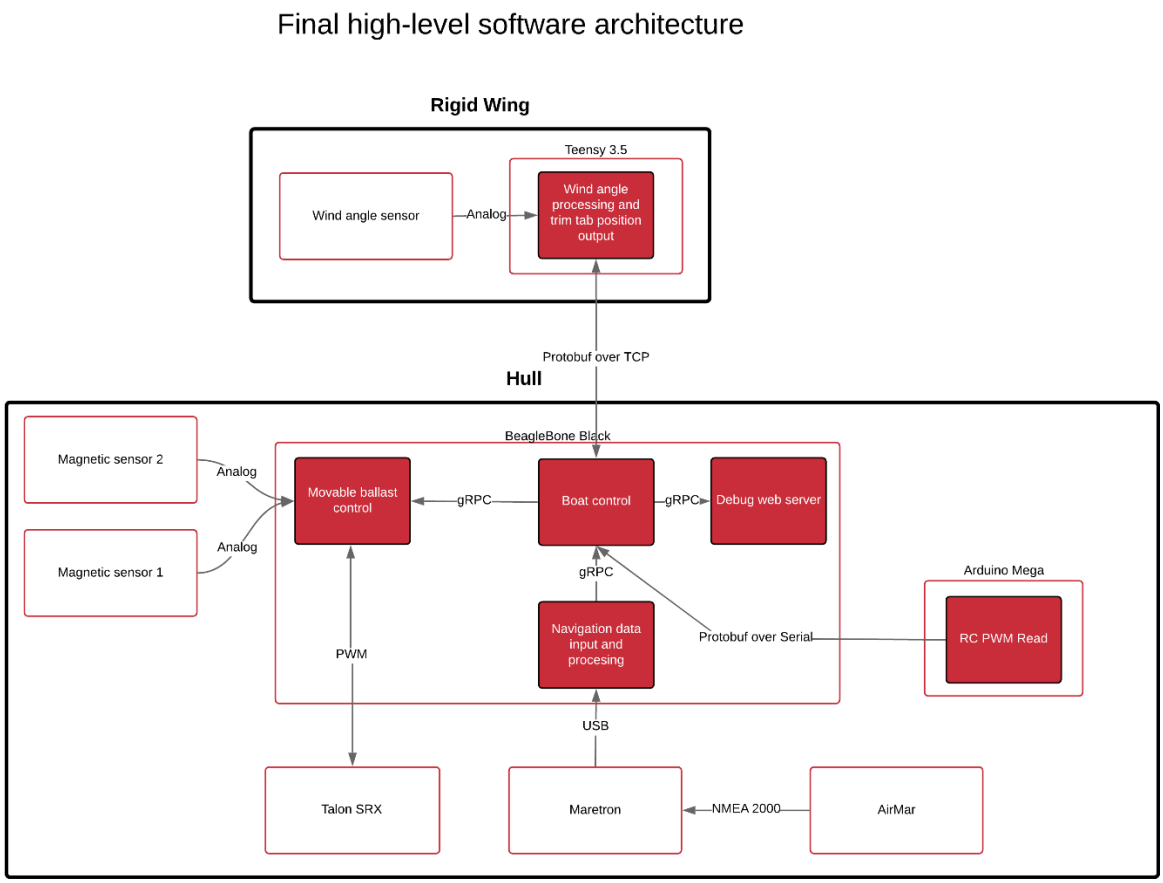

Figure 32: System architecture diagram showing the changes made this year.

For communication, the intention is to use gRPC [10] for much of it, as it is language-agnostic, fast, easy to implement, and defined using Protobufs, which can be used without gRPC [10]. The only downside of gRPC is its relatively large library, which makes it unsuitable for use with microcontrollers, such as the Arduino and Teensy. However, there exists a library called Nanopb [11], which is a stripped down version of Protobufs implemented for microcontrollers such as these. Because gRPC is based on the Protobuf serialization protocol, messages received using gRPC can be forwarded to these microcontrollers without any extra processing.

For more detailed information on topics discussed in this section, please refer to the GitHub repository at https://github.com/wpisailbot/sailbot19-20.

4.4.1. Hull

As Figure 32 above shows, the hull contains two microcontrollers. The Arduino Mega has stayed from the previous year, but its purpose has changed, as will be discussed. The CTRE Hero board was removed,

while a BeagleBone Black (BBB) was introduced. These changes are the result of the discrepancies mentioned earlier in this section.

The initial architecture assumed that the Teensy would communicate with the Arduino Mega, which would act as the computational hub for the whole system. This would have allowed for easier implementation of changes to the subsystems, and it would have centralized the sensor inputs to a single controller. However, there was no simple way to read the Airmar weather station using this microcontroller. This sensor communicates using either the NMEA2000 or NMEA0183 over USB via the Maretron. A USB shield was acquired by the previous team in anticipation of using the latter method, but when this team attempted to implement this method, it was discovered that the USB shield requires drivers for the device it is reading [12]. This was an insurmountable hurdle because the Maretron is a proprietary device without free access to drivers and no drivers exist for interfacing the Maretron with the USB shield. Writing the drivers would have set the team farther back in the long run than finding an alternate solution. This is the primary reason for introducing the BBB into the system because it runs an embedded Linux system and as such ships with drivers. This bypasses the driver issue with the Maretron. The BBB was chosen over similar embedded Linux platforms such as the Raspberry PI because there were multiples of this board in the lab from previous boat iterations, there was a potential codebase to draw from written several years ago, and the other boards did not appear to offer any significant benefits.

As the Arduino functionality was transferred to the BBB, several discoveries were made. One was that the switch to the BBB and Python had an unintended consequence in that it improved the quality of the connection between the Teensy and the hull. The Python socket library is slightly better written than the Arduino Ethernet library, with better error handling and a friendlier API. Another discovery was that the BBB has very limited PWM capability. While there are six PWM outputs on the RC controller, the BBB only has two PWM inputs not used by the system and four total. Furthermore, implementing these ports required an outdated library. As such, the Arduino Mega could not be completely phased out of the system as it is still used to read PWM inputs from the RC receiver and transfer these readings over UART to the BBB. While this works, it is clunky and could be replaced with a dedicated solution built using the BeagleBone Proto Cape. The BBB also only has two PWM output ports. Luckily, only two hull subsystems require PWM outputs and the BeagleBone Servo Cape could extend this to 16 outputs should the design change to require more PWM outputs.ython with this system.

In order to enable modularity, the gRPC library was used to implement Inter-Process Communication (IPC). gRPC was chosen for several reasons. The primary and most important reason is it is based on the Protobuf serialization protocol. Before the switch to BBB, a library called NanoPB was used to enable data serialization in the communication between the Arduino Mega and Teensy. This library is a slimmed down version of the Protobuf protocol, which means that gRPC messages can be forwarded to these microcontrollers without any additional work. Secondly, gRPC is cross-platform and multi-language. This enables future teams to build tools like simulations directly on top of the code written for the boat and does not lock them into Python. Lastly, gRPC is easy to use and pick up. The API is simple, and it has excellent documentation, examples, and community support.

As the Arduino functionality was transferred to the BBB, several discoveries were made. One was that the switch to the BBB and Python had an unintended consequence in that it improved the quality of the connection between the Teensy and the hull. The Python socket library is slightly better written than the

Arduino Ethernet library, with better error handling and a friendlier API. Another discovery was that the BBB has very limited PWM capability. While there are six PWM outputs on the RC controller, the BBB only has two PWM inputs not used by the system and four total. Furthermore, implementing these ports required an outdated library. As such, the Arduino Mega could not be completely phased out of the system as it is still used to read PWM inputs from the RC receiver and transfer these readings over UART to the BBB. While this works, it is clunky and could be replaced with a dedicated solution using the BeagleBone Proto Cape. The BBB also only has two PWM output ports. Luckily, only two hull subsystems require PWM outputs and the BeagleBone Servo Cape could extend this to 16 outputs should the design change to require more PWM outputs.

As mentioned before, the CTRE Hero board was removed from the system. This board was initially introduced last year because the TalonSRX motor controller uses the CAN bus. However, this was not necessary as the TalonSRX has a PWM mode, which is accessible to both the Arduino Mega and the BBB. This proved the Hero Board redundant and the control of the TalonSRX was transferred to the BBB.

The process architecture that resulted from these changes is shown below in Figure 33. While nothing relating to autonomous control was implemented, what was implemented adhered to this plan. This architecture separates out the tasks into individual processes running in parallel, which will allow more tasks to be added easily using existing processes as examples. As can be seen by the key, all processes communicate using Protobufs over one of three communication protocols. The methodology for these is discussed in more detail later in subsection Protobufs. Another main idea of this architecture is to make it agnostic of the source of input. In other words, the main process does not care whether the inputs came from the RC Receiver via the Arduino Mega or from the autonomous control process.

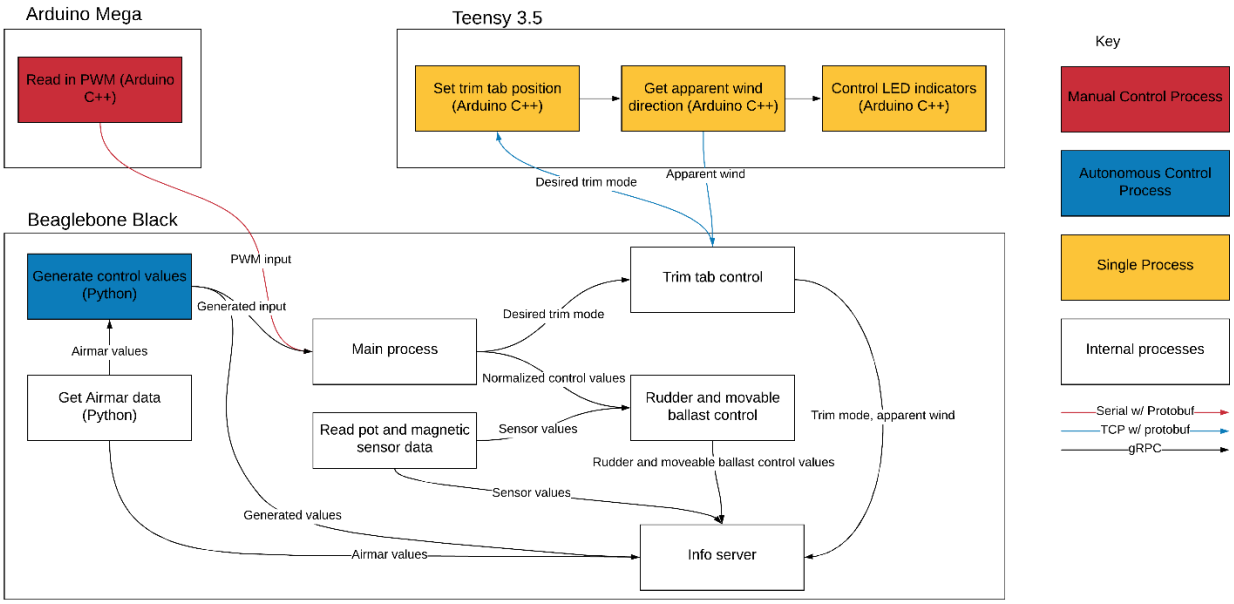

Figure 33: Low-level process diagram.

4.4.2. Trim tab

The Teensy controls the trim tab, which influences the angle of attack of the rigid wing sail and thus the direction and magnitude of the produced force vector. It communicates with the BeagleBone Black using the ESP8266 chip. The code inherited did not communicate with Arduino Mega in the hull. The reason for this difficulty was identified to be the cryptic sentence structure that the wireless chip uses. This was exasperated by the fact that most of the materials on the operation of this chip were either difficult to comprehend or incomplete. After some research, a library was located called WiFiEsp [13]. This library had examples for setting up the board as both a client and a server. Since the objective of the Teensy is to essentially receive instructions, respond with the wind encoder reading, and control the servo accordingly, the code running on the Teensy is essentially a modified version of the client example from the WiFiEsp library.

In order to communicate with the hull, the Teensy uses the Nanopb library. This is a library that cuts down the standard Protobuf libraries to a size that can be used on microcontrollers with limited memory [11]. This library was used primarily to serialize the structs that were used to describe the state of the boat and send the resulting bit stream over the LAN. Because gRPC is based on the Protobuf protocol, the Nanopb implementation did not have to be modified when the Arduino was replaced by the BBB.

The trim tab implements an LED state indicator. The indicator has four LEDs – two white, one yellow, and one red. Table 4 shows the meanings of the various LEDs.

Table 4: LED mode meanings.

STATE	LED	MODE
POWER ON	Red	Solid
WIFI CONNECTING	Yellow	Strobe
WIFI CONNECTED	Yellow	Solid
MAX LIFT – PORT	White1	Solid
	White2	Off
MAX LIFT – STARBOARD	White1	Strobe
	White2	Off
MAX DRAG – PORT	White1	Off
	White2	Solid
MAX DRAG – STARBOARD	White1	Off
	White2	Strobe
MIN LIFT	White1	Off
	White2	Off
MANUAL	White1	Solid
	White2	Solid

4.4.3. Arduino Mega

As mentioned previously, the majority of the functionality previously enabled by the Arduino Mega was reimplement using the BBB, but the Mega could not be phased out due to its usefulness in reading PWM signals. Reading PWM signals is done using interrupts on the PWM input pins. In essence, the interrupts time rising edges. Knowing the frequency of the PWM signal, one can deduce the period and thus the duty percentage of the measured signal. This information is then sent to the BBB. Sending the

information was more troublesome than initially expected because of electrical complications described in Electrical Systems and the choice of communications protocol. Initially, UART was chosen as it is straightforward on both platforms. At this point, all PWM signals were centralized on the Arduino, so the BBB would send desired positions over the UART to the Arduino, which would then generate the PWM signal on the appropriate pin. The Arduino would then return the readings it got of the six RC receiver channels and transfer that over UART as a response. This worked in the small scale, but proved unreliable when tested on the boat. Several reasons could have been the cause. First, UART is notoriously slow compared to SPI or I2C. Secondly, the logic level conversion was implemented incorrectly initially.

In order to address the speed of the connection, a conversion to SPI was attempted. However, this was ultimately unsuccessful because of issues with data representation. The SPI library on the BBB required that data is represented using integers, while the bytes are represented as strings in Python and thus Protobufs are serialized to strings. This was overcome using a series of conversions from string to hex to integer on the BBB. However, this conversion could not be performed on the Arduino. The conversion mixed ASCII codes with encoded strings. For example, one of test string encoded to "\x08[\x10\\". In this example, “\x08” and “\x10” are ASCII character codes, while “]” and “\\” are already encoded characters. The team could not find any library, method, or algorithm that could deal with this input because Arduino does not allow character arrays to mix these two methods and so reconstructing the strings using a table or other method is impossible.

After an unsuccessful attempt with SPI, the team decided to revert to UART since it has been demonstrated as operational, but reduce the traffic on the bus and acquire a COTS logic level shifter. The traffic was reduced by limiting it to one way packets with RC receiver channel information generated by the Arduino and offloading the PWM generation to the BBB. The Adafruit BBIO library was used once again to generate the PWM.

4.4.4. Info server

As shown in Figure 33, the info server aggregates all of the system state information. Its purpose is to show all of the information that the boat uses to make decisions. This webserver can be accessed by navigating to the IP address of the hull, port 5000. Since the boat’s router uses static IP allocation, this should be 192.168.0.21:5000. The server has two subpages: Subsystem_state and Airmar. Navigating to Airmar should look like 192.168.0.21:5000/Airmar. The information was segmented for clutter purposes and ease of use. The webpages require refreshing in order to update, so it was critical that they fit in their entirety on most computer screens. These pages can be unified in the future should that be necessary. The pages were created using the TOPOL.io service for creating HTML emails. Since the server was implemented using Flask [14], the variables can be input directly into the template using the syntax {{var_name}}.

Because the Flask run() method is blocking, the webserver was implemented as a separate process and uses gRPC to communicate with all of the processes to gather information. In order to simplify the startup process, a simple bash script can be written. It can either be run from an SSH session, or be added to the boot sequence, which will make sure that the processes start when power is applied to the board. When the processes are started using the bash script, it is important to realize that the keyboard interrupt signal will not be forwarded to the processes, so either a way of forwarding this signal must be devised or an alternate method of cleaning up at shutdown must be implemented in the future.

4.4.5. Protobufs

In order to simplify the generation of language specific Protobuf files, a script was written. This script runs the Google protoc program to generate Protobuf files for the BBB, the Nanopb protoc to generate the Protobuf files for the Arduino based boards, and the gRPC plugin for the gRPC-specific message files for the BBB. This script was written for used on Windows, but it uses variables that should be easily modified for use with different setups. This is noted in the comments of the file.

To summarize the development process with Protobufs, first a proto file is created that defines the message structure, size, and variable names. Protobuf structures are extremely flexible in the types, number of variables, nested variables and messages and so forth. Next, these messages are used to generate language specific resource files. These are then imported into the project the same way that a library would be and the variables and methods are then used to encode the data. Because Protobufs support a wide variety of languages, a single proto file can be used to define the message structure used by multiple microcontrollers using different languages. For details on the exact proto files implemented in this project, refer to the project's GitHub page [15] under Protos.

In order to simplify the generation of language specific Protobuf files, a script was written. This script runs the Google protoc program to generate Protobuf files for the BBB, the Nanopb protoc to generate the Protobuf files for the Arduino based boards, and the gRPC plugin for the gRPC-specific message files for the BBB. This script was written for used on Windows, but it uses variables that should be easily modified for use with different setups. This is noted in the comments of the file.

5. Results and Future Recommendations

Reflecting on the goals of this project proposed in Purpose and Project Goals, the team identified 12 definite goals and 6 stretch goals for this project in total. The definite goals were:

- Prevent movable ballast over-rotation
- Re-work keel insert area
- Redesign trim tab electronics housing
- Improve rudder efficiency and durability
- Mount additional sensors
- Change power source
- Document all new electrical components
- Add sensing to mechanical systems
- Write software for control systems
- Validate Airmar data and location optimization
- Make software architecture diagrams
- Enable communications between hull and trim tab

The stretch goals were:

- Reduce the weight of the boat
- Refinish the hull
- Redesign boat stand for portability
- Set up navigation based on Airmar and vision
- Introduce autonomy
- Set up navigation based on AirMar and vision

This section will cover how much progress was made towards each one of these goals and any future recommendations for improving the boat's systems in that area.

5.1. Definite Goals

5.1.1. Prevent Movable Ballast Over-Rotation

A major issue this year was dealing with the movable ballast system. The movable ballast itself had a tendency to rotate beyond its designed limit, which led to physical damage on some of the gears in the system and significant scratches on the hull. Additionally, the boat is much more difficult to control when this is happening. As it stands now, this goal is partially complete. A major mechanical upgrade to the boat was designed to physically prevent movable ballast from going too far; hard stops were designed, manufactured, and implemented onto the boat; sensors were added at the ends of travel, but they have not been implemented in software yet.

5.1.2. Re-work Keel Insert Area

The keel insert cavity was a problematic area that required a lot of work. Due to tight tolerances and high stresses, complications quickly arose when working with this section of the boat. Despite this, the keel insert cavity is much more robust now that at the start of the year. Fixes to both the inside and

outside of the keel insert cavity have increased the waterproofing in this area and reduced the chance of future leaks developing. The fit of the keel insert into this cavity has also seen improvements. Small changes in the shape of the insert reduce the chances of damaging the insert cavity during insertion. An improved fit inside the cavity and a longer bolt to hold the keel in place ensure a stronger connection between the keel and boat. However, there are still improvements that can be made in this area. When inserting the insert there is a fair amount of wiggling required to get the insert fully into the cavity. By improving the fit further, the amount of wiggling could be reduced, allowing for easier and quicker insertion of the keel into the boat.

5.1.3. Redesign Trim Tab Electronics Housing

Unfortunately, although significant process was made on the trim tab housing redesign, the team was unable to fully complete the project due to campus closing for D term. The housing was redesigned, and all the required parts were either taken off the old housing or manufactured by the team. While most of the housing was successfully assembled, there are a few last steps required to complete the trim tab housing. These steps have been outlined in a document for next year's MQP team so they can easily finish the redesign process. This updated design should help solve several issues that were present in the old trim tab housing. The new design will allow the trim tab to turn left or right by about the same amount, unlike the old design where it turned significantly more towards one side than the other. The new design also features redesigned part connections and additional support pieces which reduce the likelihood of internal parts falling out of their mounted positions. Lastly, changes to the shape of the foam layers reduce wear damage to the foam and allow for a greater range of motion for the trim tab servo. Overall, the redesigned trim tab housing will solve several issues present in the old design and help improve the durability and performance of the boat.

5.1.4. Improve Rudder Efficiency and Durability

The rudders at the beginning of the project had a rough finish on them and small epoxy bubbles lined the edges, which was decreasing the rudders' efficiency by creating more drag in the water. The team sanded the rudders down, re-epoxied them, and then sanded them down again to create a smooth, even finish across the surface of the rudders, which significantly increased their efficiency and durability in the water.

5.1.5. Mount Additional Sensors

Since the magnetic sensors were the only ones ready for mounting, they are the only ones that were semi-permanently attached to the boat. Although not an elegant solution, hot glue worked very well for mounting the Hall Effect sensors, while the small supporting board was moved away from the sensors using tails. Additionally, the accompanying magnet was mounted to the arm of the movable ballast via a 3D printed enclosure that can be fixtured with a thru-bolt, although the lab currently does not have the appropriate thru-bolt required. A more elegant solution to mounting the magnetic would have added to the boat's aesthetic but was ultimately not necessary and the prototyping was eating away at time the team could have used for other tasks.

5.1.6. Change Power Source

Changing the power source was a vital goal for the operation of the boat because the lead acid batteries could only hold sufficient charge for a few minutes before dropping too much voltage. The switch to a lithium-based chemistry and specifically the Lithium Polymer battery was a big improvement for this

year. Calculations using conservative numbers showed that this battery should last well into the endurance challenge. Because this challenge allows pit stops to change batteries, this is good enough. However, a stress test could not be conducted to verify these calculations and one should be conducted next year. A similar stress test should also be conducted on the trim tab. While the endurance of the LiPo on the trim tab appears to have adequate, this could not be empirically verified before the campus closure.

5.1.7. Document All New Electrical Components

Since this year focused on software primarily, there were very few custom electrical solutions. However, the few that were implemented were drawn up in Eagle and can be seen in this report. Furthermore, the communications methods were documented both in the code and in diagram form. There is still work to be done in documenting the electrical solutions from years before this one. This is especially true of the Teensy board as it is a fairly cluttered board that is difficult to troubleshoot.

5.1.8. Introduce Sensing to Mechanical Systems

Unfortunately, only some of the planned sensors could be implemented this year. The magnetic sensors were mounted and tested electrically, but could not be tested fully in software due to the closure of the campus. The logic was tested with a stand-in in a small scale, but it must still be verified on the boat before it can be ready to sail. Furthermore, the potentiometer that was supposed to replace the magnetic sensor on the moveable ballast shaft could not be implemented because this idea was introduced late into the year and the replacement shaft could not be machined in time before campus closure. As such, some kind of absolute or relative sensing must still be implemented on the moveable ballast in order for it to be controllable in software. This can be done by reimplementing the magnetic sensor, implementing the potentiometer plan devised this year, or implementing some other solution. A relative solution can work using the magnetic sensors as limits, but this solution will likely be less accurate as there is not guarantee that the magnetic sensors are exactly equidistant from the center of the moveable ballast's rotation arc.

5.1.9. Write Software for Control Systems

While significant progress was made towards writing software for the control systems, this goal was not fully achieved this year. This was because a large amount of time was wasted attempting to bring the incumbent system to life when it was not capable of meeting all requirements. Additionally, time had to be spent acquainting with new hardware that replaced the old inadequate systems. Even so, a basic structure was implemented which should set a solid foundation for future teams to build upon. Importantly, communication was established between all major subsystems, which has never been done before. While the existing codebase was tested in a small-scale setup, given more time on campus, it would have been tested on the boat and highly optimized for manual control if no issues were discovered. This would have tested all of the systems in concert, potentially showing errors in logic or strategy and eased the team into implementing autonomous functions by sequentially automating manual motions.

5.1.10. Validate Airmar Data and Location Optimization

The quality of the data returned by the Airmar could not be validated this year due to time and campus access restrictions. However, a plan was created that would simultaneously test the quality of the

Airmar's date and determine an ideal location on the boat for the Airmar. This plan can be used or improved upon by next year's team in order to achieve these goals.

5.1.11. Make Software Architecture Diagrams

Every effort was made to document the software written this year. Lack of documentation was very large hurdle last year and this team did not want to cause a repeat. Detailed comments accompany all of the relevant code, and miscellaneous files are clearly marked as such. Furthermore, diagrams were created to reflect the intention of the code so that it is clear to anyone taking this project over the direction in which the project was headed when it was handed off. Future teams should attempt to continue this endeavor as it will prevent future teams from abandoning previous work simply because they don't understand how it works.

5.1.12. Enable Communications Between Hull and Trim Tab

The BBB in the hull and the Teensy controlling the trim tab successfully communicate useful information without significant disruptions or delays. This was a major milestone, but improvements can still be made. One of the most major improvements that can be made to this system is recovery from a lost connection. This feature used to exist when the Teensy communicated with the Arduino, but was lost in the process of converting to the BBB. The WifiESP library includes methods that allows the Teensy to detect a loss of connection event and initiate a search. Similarly, the Python Socket library has methods that allow the detection of connection loss.

5.2. Stretch Goals

5.2.1. Reduce Weight of Boat

Reducing the overall weight of the boat, while potentially beneficial, was deemed to be non-essential since many systems in the boat are already optimized to be very light-weight. Nonetheless, switching the battery from lead-acid to LiPo has shed approximately 5 pounds. Going forward, weight reduction should at the very least remain a design factor, if not a major goal involving redesigning existing systems in the boat.

5.2.2. Refinish the Hull

No progress was made towards refinishing the entire hull this year. Ultimately, this is mostly a cosmetic issue so it is not an essential task for the boat's operation. However, at some point exterior paint damage may affect the integrity of the hull or will make the boat look less impressive visually. But until one of these issues develop, this task mainly remains an unneeded but nice fix that can be done if a future MQP group has some extra time.

5.2.3. Redesign Boat Stand for Portability

While improvements were made to the stand for SailBot, they did not focus on improving the portability. The back two side support pieces were replaced with brand new supports that were twice as thick as the old ones. This was because one of the old supports broke off, leaving the boat vulnerable to sliding off the stand. However, alterations beyond this were deemed not necessary. Since SailBot rarely needs to be on a stand outside the lab, redesigning the whole stand to increase portability for something that happens only once or twice a year was not worth it.

5.2.4. Convert to NMEA2000

While reading NMEA sentences currently works with translating NMEA2000 into NMEA0183 using the Maretron, there are many advantages that can be leveraged by switching to just reading the NMEA2000 sentences. Firstly, removing the Maretron opens up a USB port on the BBB. Additionally, removing the Maretron would allow for the implementation of CAN in the case where USB cannot be used. And lastly, NMEA2000 is the current maritime standard, thus switching to it would allow for easier integration of other marine equipment in the future.

5.2.5. Introduce Autonomy

A signification portion of this year was spent establishing a teleoperated system that would lend itself to effortless conversion to autonomy and one the resources for which would be widely available. The software architecture for was structured such that it does not care when data is coming from so long as it is in the right format, as dictated by the Protobuf structures. As such, the intention is that the next team can write a set of BBB processes that can plug into the existing system. That said, given that the existing system could not be fully tested in time before the unexpected closure of campus, some changes may be required.

5.2.6. Setup Navigation Based on Airmar and Vision

Vision

Vision will greatly aid in most of the challenges that the SailBot competition poses because it will allow the boat to discern its location relative to the buoys and to detect obstacles in its path. Vision was one of the stretch goals for this year and it was approached to some degree with some simple Matlab masking scripts, but more development is required in order for this to be useful. Future teams should look to code from previous years (2015-16 and 2016-17) for inspiration on how to implement vision as they were able to do both buoy and obstacle detection, but rewriting will be required because of differences in implementation between the years.

Inertial Navigation

Autonomous navigation of any area requires accurate position and velocity state information in order to implement motion-planning algorithms. GPS-aided inertial navigation is capable of providing very accurate state information, even in the presence of sensor noise. This topic is complex and requires understanding of such concepts as coordinate transformations, kinematic modeling, ordinary differential equations, numerical integration methods, and Kalman Filtering.

5.3. Transition Documents

5.3.1. Software

Refer to the markdown files in the project's GitHub: https://github.com/wpisailbot/sailbot19-20. Wherever necessary, each subsystem has its own markdown file with instructions and explanations. Additionally, the code contains many comments to aid in transitioning.

5.3.2. Electrical

The majority of the electrical work this year took place with the battery and in support of the software work. This included switching the land connection to the RC controller with PWM output, changing

motor control to PWM, and switching the battery chemistries and accommodating the change to a different computational unit in the hull.

The RC receiver is powered by 5V from the box with the unused CTRE Hero board. It outputs PWM signal and is thus useful for testing motors that use this control signal protocol. This is also useful if the boat needs only to operate in tele-operated mode without autonomy, as the only system that cannot be directly controlled from the RC receiver is the Trim Tab.

The only major change that the battery change precipitated is the speed of the window motor that powers the movable ballast as that is the only system powered directly from the battery. All other systems on the boat use voltage regulated down to 5V. Because of the voltage regulation, nothing on the boat was changed to accommodate the new battery and the old lead-acid batteries are still compatible should that be necessary.

The LiPo and the lead-acid batteries use different chargers. Figure 34 shows the lead-acid charger and Figure 35 shows the LiPo charger. While charging the lead-acid battery is as straight forward as attaching the leads of the battery to the charger, charging the LiPo requires extra care by either selecting or setting up a profile for the battery and plugging in the battery monitoring circuit to the charger. The charger pictured in Figure 35 already has a profile set up for the LiPo used in this project. This video was particularly helpful when setting up this profile and may be helpful should another profile have to be programmed: https://youtu.be/5AIAomVTKHw.

Figure 34: Lead-acid battery charger.

Figure 35: LiPo battery charger.

5.3.3. Mechanical

Current state of the trim tab housing:

- All foam layers have been cut to the appropriate shapes and glued together
- The trim tab, the wind encoder rod, the USB plug, and the circuit board with lights have all been glued to the housing
- All other remaining parts needed have been made and are waiting to be attached.

Steps to finish trim tab housing:

1. One end of the trim tab rod needs to be reattached. The part that needs to be attached is a small rubber piece that is very similar to the piece on the other end except it's slightly thinner. Make sure these pieces on either side have their holes aligned in the same direction and then glue the piece onto the rod.
2. The trim tab rod needs to be reattached to the servo crank. There is a small bolt that fits through the servo crank and the end of the rod and it is secured in place using a nut. Make sure the bolt is pointing up so that the side with the nut is on the top. This will prevent the bolt from hitting the foam below.
3. Screw the servo crank back on the servo.
4. The housing now needs to be covered on the outside. I was planning on using tape like the old design so refer to pictures of the old housing for how it should look. Cover the outside of the housing with the tape, ensuring that all parts exposed during sailing are covered.
5. Glue the 3D printed part that protects the top hole in place. This part is identical to the old design so refer to old pictures of the trim tab if needed.
6. Lastly, place all the electronics back in the electronics box and put the box back into the housing. Reconnect all wires as necessary.

6. References

[1] A. c, Artist, *Points of Sail.* [Art]. Wikipedia, 2007.

[2] K. Quant, "The Basic Sailboat Racing Rules All Racers Should Know," 2009. [Online]. Available: https://www.bcya.com/Misc/BasicRacingRules.pdf. [Accessed 14 December 2019].

[3] United States Coast Guard, "Navigation Rules: International-Inland," [Online]. Available: https://www.navcen.uscg.gov/pdf/navRules/navrules.pdf. [Accessed 14 December 2020].

[4] A. Pooler, "Amesbury, MA," 16 May 2013. [Online]. Available: https://www.amesburyma.gov/sites/amesburyma/files/file/file/assessment130517.pdf. [Accessed 11 October 2019].

[5] "Division of Fisheries and Wildlife," 2018. [Online]. Available: https://www.mass.gov/files/documents/2018/03/06/quinsigamond.pdf. [Accessed 11 October 2019].

[6] "Portmouth, NH History," 2019. [Online]. Available: https://www.wunderground.com/history/monthly/us/nh/portsmouth/KPSM/date/2017-6.

[7] "RC Groups," [Online]. Available: https://www.rcgroups.com/forums/showthread.php?2457851-TARANIS-X9D-Range. [Accessed 14 December 2019].

[8] T. Dunn, "RC Battery Guide: The Basics of Lithium-Polymer Batteries," Adam Savage's Tested, 15 March 2015. [Online]. Available: https://www.tested.com/tech/502351-rc-battery-guide-basics-lithium-polymer-batteries/. [Accessed 14 December 2019].

[9] "Turnigy High Capacity 20000mAh 4S 12C Lipo Pack w/XT90," HobbyKing, [Online]. Available: https://hobbyking.com/en_us/turnigy-high-capacity-battery-20000mah-4s-12c-drone-lipo-pack-xt90.html. [Accessed 14 December 2019].

[10] Google, "gRPC," Google, [Online]. Available: https://grpc.io/. [Accessed 14 December 2019].

[11] P. Aimonen, "Nanopb - Protocol Buffers for Embedded Systems," GitHub, 12 December 2019. [Online]. Available: https://github.com/nanopb/nanopb. [Accessed 14 December 2019].

[12] A. Mazurov, "USB Host Library Rev.2.0," Circuits@Home, 13 September 2019. [Online]. Available: https://github.com/felis/USB_Host_Shield_2.0. [Accessed 14 December 2019].

[13] B. Portaluri, "WiFiEsp," GitHub, 03 March 2019. [Online]. Available: https://github.com/bportaluri/WiFiEsp. [Accessed 14 December 2019].

[14] "Flask," The Pallets Projects, 24 November 2019. [Online]. Available: http://flask.palletsprojects.com/en/1.1.x/. [Accessed 14 December 2019].

[15] WPI Sailbot, "Sailbot 19-20," 27 February 2020. [Online]. Available: https://github.com/wpisailbot/sailbot19-20. [Accessed 17 March 2020].

[16] H. Serrano, "Visualizing the Runge-Kutta Method," 4 May 2016. [Online]. Available: https://www.haroldserrano.com/blog/visualizing-the-runge-kutta-method. [Accessed 13 October 2019].

[17] "Adafruit Beaglebone I/O Python API," Adafruit, 02 April 2019. [Online]. Available: https://github.com/adafruit/adafruit-beaglebone-io-python. [Accessed 14 December 2019].

[18] "EthernetServer()," Arduino, [Online]. Available: https://www.arduino.cc/en/Reference/EthernetServer. [Accessed 14 December 2019].

[19] O. Halytskyi, "PJON-gRPC," GitHub, 30 November 2018. [Online]. Available: https://github.com/Halytskyi/PJON-gRPC. [Accessed 14 December 2019].

[20] G. B. Mitolo, "PJON 12.0," GitHub, 09 December 2019. [Online]. Available: https://github.com/gioblu/PJON. [Accessed 14 December 2019].

[21] B. Ripley, "Three Ways To Read A PWM Signal With Arduino," 06 June 2014. [Online]. Available: https://www.benripley.com/diy/arduino/three-ways-to-read-a-pwm-signal-with-arduino/. [Accessed 14 December 2020].

7. Appendices

7.1.Inertial Navigation

Implementing inertial navigation will allow for the advanced motion planning algorithms required for autonomous operation. Figure 36 shows the basic information input and output of an INS system.

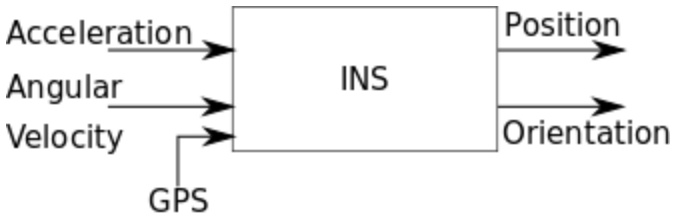

Figure 36: INS block diagram.

Inertial navigation is a complex topic that takes advantage of rotation matrices for coordinate transformation, ordinary differential equations, Newtonian physics, numerical integration methods, and predictive filtering techniques.

7.1.1. Coordinate transformations

Inertial navigation deals in four different coordinate systems, pictured in Figure 37. Following is a description of each coordinate system and its significance to implementing inertial navigation.

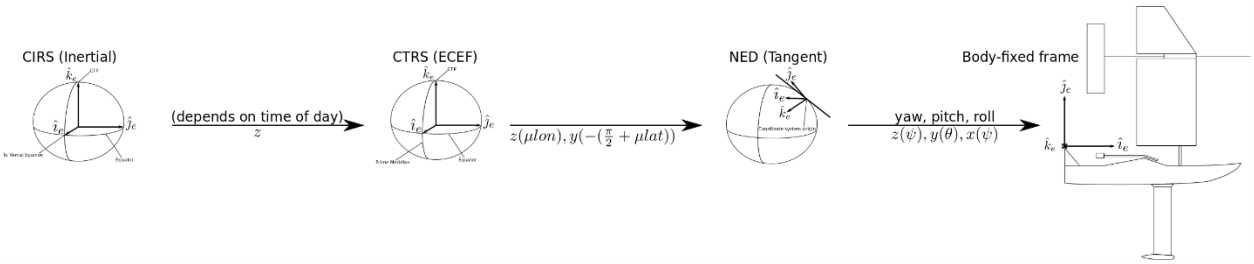

Figure 37: Coordinate systems required for Inertial Navigation.

Conventional Inertial Reference System (CIRS)

Figure 38 below shows the definition of the Conventional Inertial Reference System (CIRS). This system has its z-axis pointing at the instantaneous North Pole (aka Conventional Terrestrial Pole or CTP), x-axis toward the Vernal Equinox, and the y-axis completes the right-handed triplet. This system is required for inertial navigation because physical laws and equations are defined only in the inertial frame. Therefore, all calculations are performed in this coordinate frame.

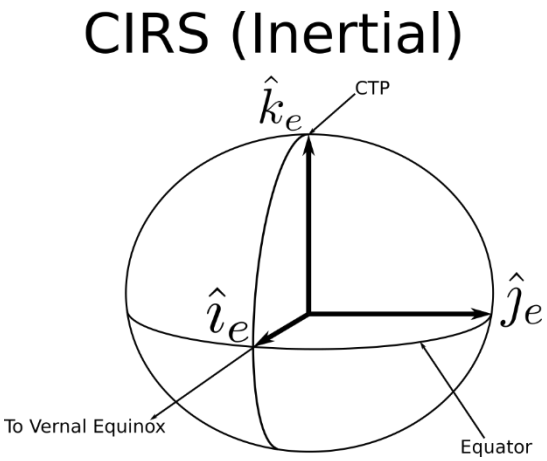

Figure 38: Illustration of the Conventional Inertial Reference System definition.

Conventional Terrestrial Reference System (CTRS)

Figure 39 below shows the Conventional Terrestrial Reference System (CTRS). This system uses a definition very similar to that of the CIRS. The only difference between the CIRS and CTRS is that the x-axis points at the Prime Meridian (aka Greenwich Meridian). Like the CIRS, this system is Earth-centered and Cartesian. Unlike the CIRS, it is also Earth-Fixed. This is an intermediary system in the inertial navigation calculations useful because terrestrial systems typically like to reference landmarks fixed on the planet. This system allows the landmark to maintain constant coordinates with time, simplifying calculations.

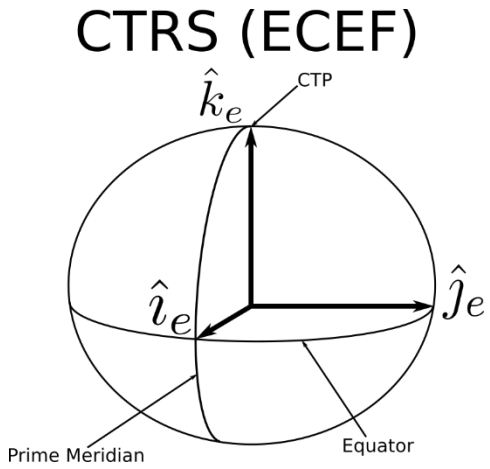

Figure 39: Illustration of the Conventional Terrestrial Reference System definition.

Converting from CIRS to CTRS requires only the time of day. Figure 40 shows how to do this conversion.

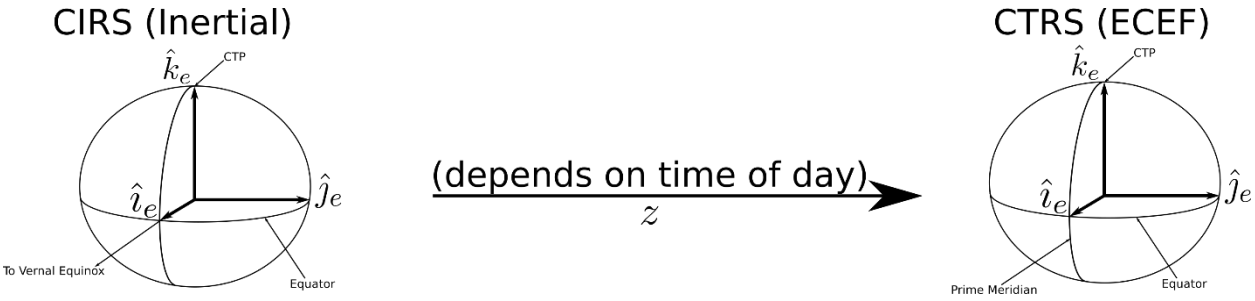

Figure 40: Diagram showing how to convert from CIRS to CTRS.

Tangent Axes System

Figure 41 below shows a type of Tangent Axes system called North-East-Down (NED). This system defines the y-axis pointing north, x-axis point East, and the z-axis point down and perpendicular to the tangent line at the origin of the system. Unlike the two systems above, this system is not Earth-Centered. Instead, this system centers on a point on the Earth's surface at a chosen point, typically origin of travel or other known reference. This is an Earth-Fixed system. This system is useful mainly for interpretation of results because navigation is typically relative to a known landmark nearby. This system is a flat-Earth approximation and not appropriate for long-distance navigation without moving the origin. This is unlikely to be an issue for SailBot at this stage in its development.

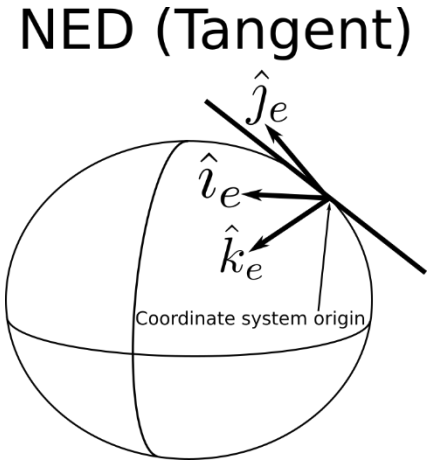

Figure 41: Illustration of the Tangent Axes System definition.

Converting from CTRS to NED requires the longitude and latitude coordinates of the point on the surface of the planet where the NED system should have its origin. Figure 42 shows the formulae for the angles and order of the rotations required to complete this transformation.

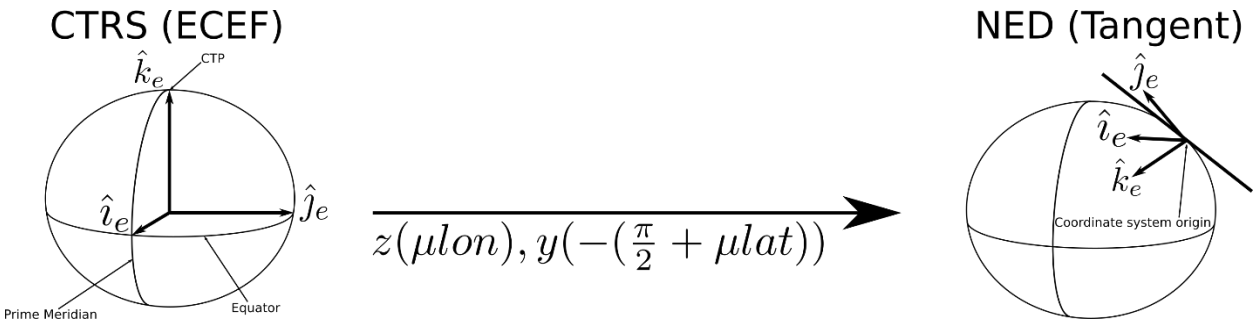

Figure 42: Diagram showing how to convert from CTRS to NED.

Body-Fixed System

Figure 43 below shows the Body-fixed frame. This system is crucial to inertial navigation when the system uses modern sensors. Strap-down sensors (as opposed to gimballed sensors) report inertial measurements in the body-fixed frame. The definition of this system will depend on the system that the sensor reports its measurements in, as well as the mounting location and orientation of the sensor. It may not necessarily match between different kinds of sensors or different manufacturers.

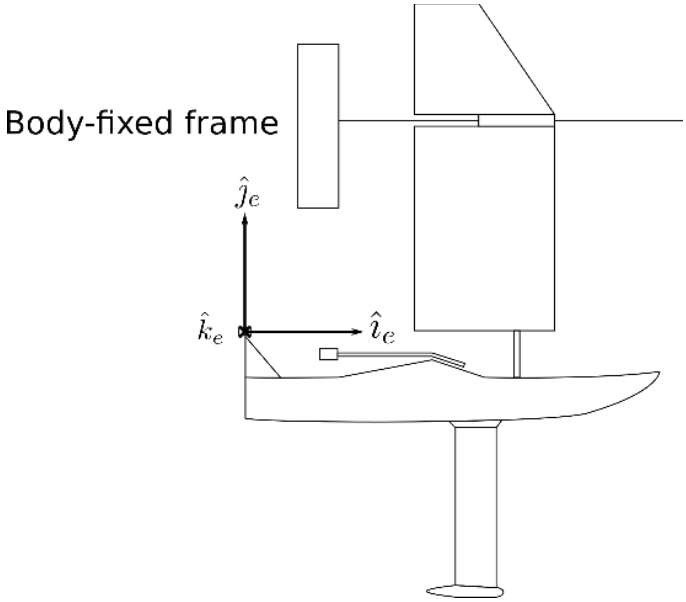

Figure 43: Illustration of the body-fixed system definition.

Converting from NED to the body-fixed system requires knowledge of the body's orientation. The transformation is a series of three rotations about three different axes. The order of the rotations matter and each rotation is about a different state angle.

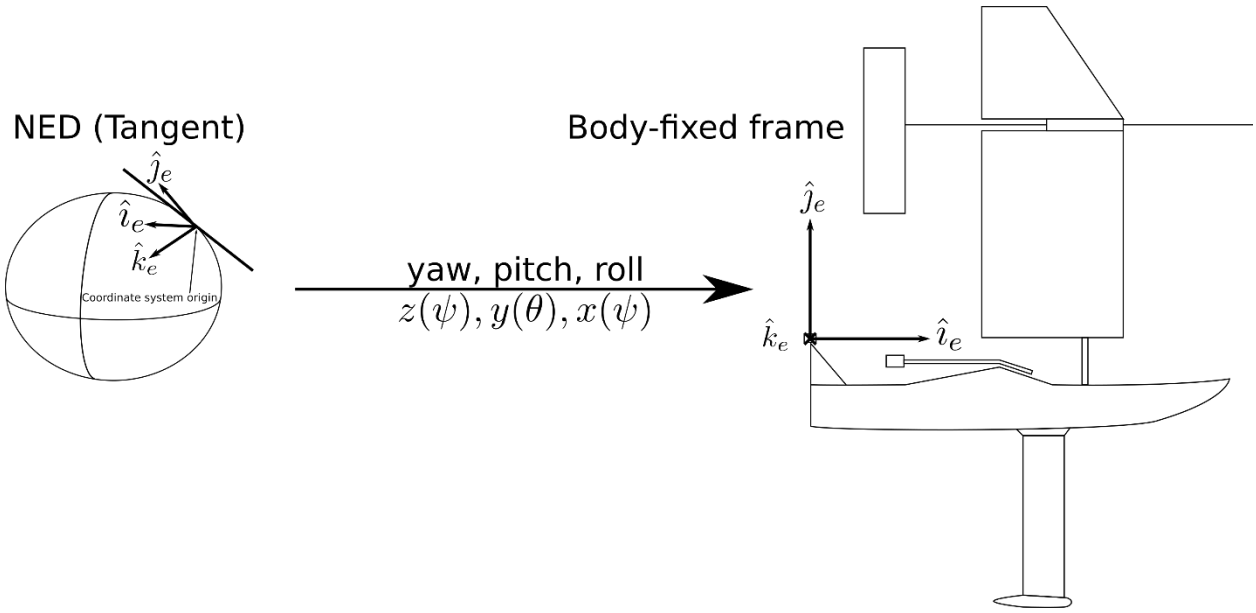

Figure 44: Diagram showing how to convert from NED to Body.

7.1.2. Numerical Integration

The most common numerical integration method used for this purpose is Runge-Kutta 4 (RK4). This method integrates first-order ODEs numerically by taking a weighted average of a linear approximation of the slope of the data. It works like the Euler Method of numerical integration, but adds the weighted average of four different slopes [16]. The equations below show RK4 with a first-order ODE.

$f(t, x(t))$ below describes the ODE that RK4 will numerically integrate:

$$\mathrm{f}\big(\mathrm{t}, \mathrm{x}(\mathrm{t})\big) = 4 + 3\mathrm{x}(\mathrm{t}) \tag{1}$$

RK4 is a discrete-time algorithm, which means that it has a time-step, defined as h in this example. The smaller the time-step, the closer the algorithm will approximate to the true value, but it will also increase the computational cost.

$$h = 0.1$$

Equations (2) through (5) show the weighted slopes of this integration method:

$$\mathrm{k1} = \mathrm{h} * \mathrm{f}\big(\mathrm{t}, \mathrm{x}(\mathrm{t})\big) \tag{2}$$

$$\mathrm{k2} = \mathrm{h} * \mathrm{f}\left(\mathrm{t} + \frac{\mathrm{h}}{2}, \mathrm{x}(\mathrm{t}) + \frac{\mathrm{k1}}{2}\right) \tag{3}$$

$$\mathrm{k3} = \mathrm{h} * \mathrm{f}\left(\mathrm{t} + \frac{\mathrm{h}}{2}, \mathrm{x}(\mathrm{t}) + \frac{\mathrm{k2}}{2}\right) \tag{4}$$

$$\mathrm{k4} = \mathrm{h} * \mathrm{f}(\mathrm{t} + \mathrm{h}, \mathrm{x}(\mathrm{t}) + \mathrm{k3}) \tag{5}$$

Finally, the weighted slopes come together to approximate the integration result x:

$$\mathrm{x} = \mathrm{x}(\mathrm{t}) + \frac{1}{6}(\mathrm{k1} + 2\mathrm{k2} + 2\mathrm{k3} + \mathrm{k4}) \tag{6}$$

The method is extendable to second-order ODEs and beyond by breaking them down into series of first-order ODEs and iteratively computing the slope approximations for each. Following is an example of a second-order ODE RK4.

$f\big(t, x(t), v(t)\big), g(t, x(t), v(t))$ below describe the ODE $\ddot{x} = 4\dot{x} + 3x$ in terms of two first-order ODEs. These equations are what RK4 will numerically integrate:

$$f(t, x(t), v(t)) = \mathrm{v}(\mathrm{t}) \tag{7}$$

$$\mathrm{g}(\mathrm{t}, \mathrm{x}(\mathrm{t}), v(t)) = 4\mathrm{v}(\mathrm{t}) + 3\mathrm{x}(\mathrm{t}) \tag{8}$$

As before, the algorithm requires a time step, which will carry a tradeoff between accuracy and computational cost:

$$h = 0.1$$

Equations (9) through (16) show the weighted slopes of this integration method. Notice the addition of an additional parameter in the integrated functions. Also, notice the iterative computation of k and l terms.

$$\mathrm{k1} = \mathrm{h} * \mathrm{f}(\mathrm{t}, \mathrm{x}(\mathrm{t}), v(t)) \tag{9}$$

$$\mathrm{l1} = \mathrm{h} * \mathrm{g}(\mathrm{t}, \mathrm{x}(\mathrm{t}), v(t)) \tag{10}$$

$$k2 = h * f\left(t + \frac{h}{2}, x(t) + \frac{k1}{2}, t + \frac{h}{2}, \mathrm{v}(t) + \frac{\mathrm{l1}}{2}\right) \tag{11}$$

$$l2 = h * g\left(t + \frac{h}{2}, x(t) + \frac{k1}{2}, t + \frac{h}{2}, \mathrm{v}(t) + \frac{\mathrm{l1}}{2}\right) \tag{12}$$

$$k3 = h * f\left(t + \frac{h}{2}, x(t) + \frac{k2}{2}, \mathrm{v}(t) + \frac{\mathrm{l2}}{2}\right) \tag{13}$$

$$\mathrm{l3} = h * g\left(t + \frac{h}{2}, x(t) + \frac{k2}{2}, \mathrm{v}(t) + \frac{\mathrm{l2}}{2}\right) \tag{14}$$

$$k4 = h * f(t + h, x(t) + k3, v(t) + l3) \tag{15}$$

$$l4 = h * g(t + h, x(t) + k3, v(t) + l3) \tag{16}$$

Finally, the weighted slopes come together to approximate the integration results both x and v:

$$x = x(t) + \frac{1}{6}(k1 + 2k2 + 2k3 + k4) \tag{17}$$

$$v = v(t) + \frac{1}{6}(l1 + 2l2 + 2l3 + l4) \tag{18}$$

7.1.3. Kalman Filtering

Kalman filtering is the industry standard for dealing with sensor noise. The basic mathematical processes required for implementation are shown in Figure 45.

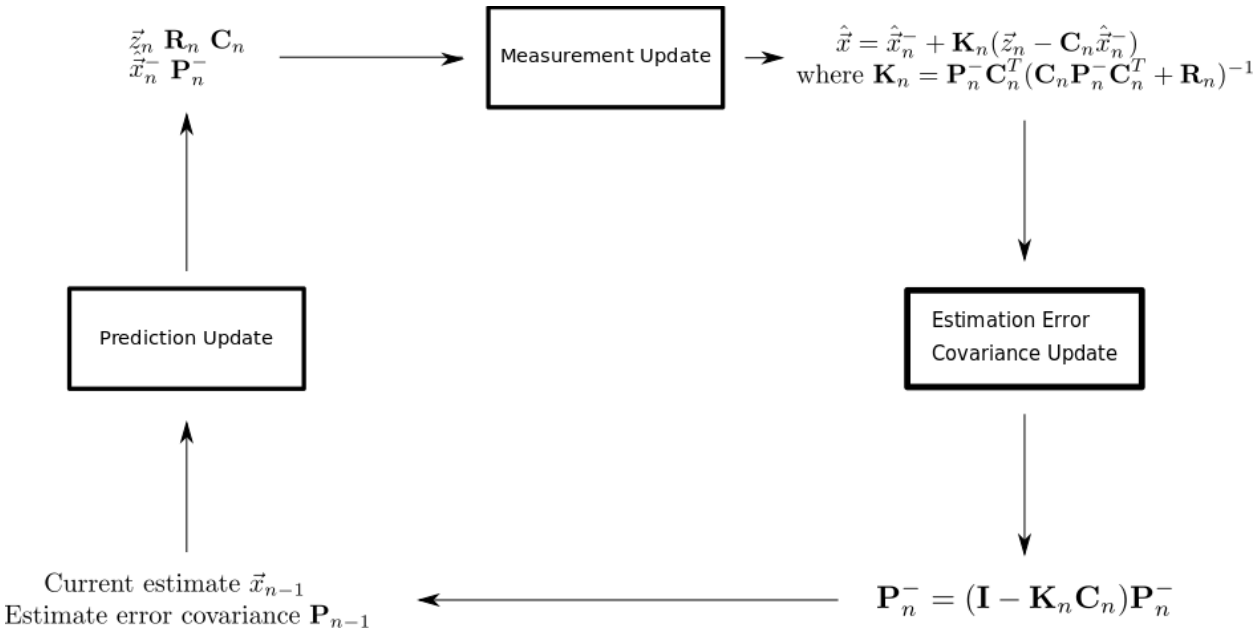

Figure 45: Diagram of the order of calculations for the Kalman Filter.

At the core of the filter is an iterative least squares estimation. Least squares estimation is a weighted average that uses confidence in the sensors' measurements as a basis for the weights of the corresponding sensor measurements. The Kalman Filter combines the least squares estimation with a prediction from the state model using some of the concepts presented earlier in this section.