SailGoat: Autonomous Sailing System (2021-2022)

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:

Andrew Del Vecchio - Robotics Engineering
Renée Gruner-Mitchell - Mechanical Engineering
Deep Kumar - Electrical Engineering
Tom Nurse - Robotics Engineering and Electrical Engineering
Jieyuan Song - Robotics Engineering and Mechanical Engineering
Molly Sykes - Mechanical Engineering
Tony Tesoriero - Robotics Engineering
Jarius Thomas - Robotics Engineering and Computer Science

Project Advisors:
Professor William Michalson & Professor Kenneth Stafford

Submitted: April 2022

Abstract

The goal of this project is to improve upon prior iterations of SailBot, an autonomous robotic sailboat. This was accomplished by improving the mechanical, electrical, and software systems already in place, as well as adding new systems. Our overall objective for this iteration is to increase the rigidity and reliability of the vehicle.

Acknowledgments

The 2021 - 2022 SailBot team would like to extend immense thanks to the following individuals for their contributions to the success of the project. Thank you to Scott Reid for donating the RDM carbon fiber mast used in the manufacturing of the wingsail, and to Samuel Ott and Douglas Moore for helping us assemble the boat. We would also like to thank our advisors, Professor William Michalson and Professor Kenneth Stafford, for all the help and guidance they provided over the course of this project. Thank you!

Authorship

Section	Author(s)
Abstract	R. Gruner-Mitchell
1.0 Introduction	R. Gruner-Mitchell
2.0 Background	–
2.1 Sailing Basics	M. Sykes
2.2 Sailing a Robotic Sailboat	J. Thomas
2.3 International Robotic Sailing Regatta	J. Song
2.4 Previous Sailboats	–
2.4.1 Mechanical Systems	R. Gruner-Mitchell
2.4.2 Electrical Systems	T. Nurse
2.4.3 Software Systems	A. Del Vecchio
2.5 Design Challenges	R. Gruner-Mitchell, M. Sykes, T. Nurse, A. Del Vecchio, D. Kumar, & J. Song
3.0 Project Goals	All
4.0 Electromechanical Design and Implementation	–
4.1 Wingsail Assembly	M. Sykes & T. Nurse
4.2 Ballast Assembly	R. Gruner-Mitchell
4.3 Electrical System	D. Kumar & T. Nurse
5.0 Software Development	–
5.1 Navigation Algorithms	A. Del Vecchio
5.2 Computer Vision	J. Thomas & T. Tesoriero
6.0 Testing and Validation	T. Nurse & A. Del Vecchio
7.0 Future Work and Recommendations	All
8.0 Conclusion	A. Del Vecchio

Contents

List of Figures

List of Tables

1 Introduction

The 2021-2022 SailBot Major Qualifying Project (MQP) is a continuation of five previous MQPs. The goal of the project is to improve the design of an autonomous sailboat by increasing the reliability of the mechanical, electrical, and software systems. The system improvements are influenced by the International Robotics Sailing Regatta (IRSR) rules, to ensure that the vessel will be a formidable competitor in the SailBot regatta hosted in June of 2022. The improvements to these systems aim to ensure SailBot's successful completion of the following IRSR events by the end of the MQP:

- Remote-controlled fleet race
- Autonomous endurance/long distance
- Collision avoidance
- Precision navigation
- Station keeping
- Object search and location

At the end of this project, we have successfully improved the wingsail assembly, the electrical systems in the hull, the ballast assembly, the autonomous sailing algorithms, and implemented a new computer vision system.

2 Background

In this section we introduce the topics critical to understanding the SailBot project. SailBot is a robotic sailboat, and thus it is important to have a basic understanding of both sailing a traditional sailboat and a robotic one. Additionally, this section introduces the International Robotic Sailing Regatta (IRSR) rules and challenges. Finally, the states of the mechanical, electrical, and software systems of the vessel as left by the 2020-2021 SailBot team are discussed.

2.1 Sailing Basics

A general understanding of the terminologies and concepts of traditional sailing are crucial to the development and optimization of a robotic sailboat. The four areas of importance are: general terminology of sailboats, points of sail, tacking and jibing, and rights of way.

2.1.1 General Terminology

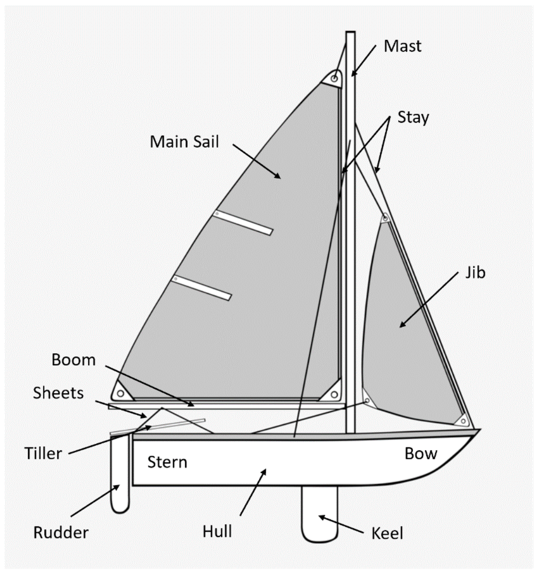


Figure 1: Anatomy of a typical sailboat

The terminology and function of each part/system in the boat are valuable to understanding how to develop and control a boat. A typical sloop rig sailboat is shown in Figure 1 with all the major components and systems labeled. Figure 2 illustrates the preexisting components and systems on SailBot as left by the previous team. The primary structural body of a sailboat is the hull, to which all other systems are attached. Attached to the bottom of the hull is the keel, a fixed underwater wing used to prevent sideways drift and provide stability. Note the difference in keel shape from the traditional sailboat (Figure 1) to SailBot (Figure 2); SailBot's keel has a bulb at the end. The rudder is the sailboat's moveable underwater steering mechanism. In a traditional sailboat, the tiller, a wooden or metal beam, is used to turn the rudder, as seen in Figure 1. Typically, sailboats

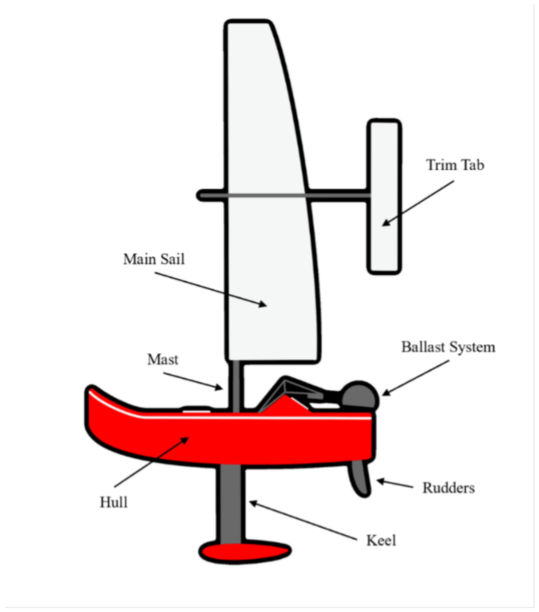


Figure 2: Anatomy of Sailbot 20-21

only have one rudder, however, SailBot has two rudders controlled by a single servo motor. SailBot's rudders are designed such that one will be vertical when the boat is heeled to the desired angle of 20 degrees.

The next system is the sail assembly. The mast is a vertical pole used to support the sails. In traditional boats, the mast is held upright and supported using wires called shrouds (or stays). Attached to the mast is the sail, which is extended using the boom, a horizontal pole that attaches to the mast. The position of the sail is controlled by the mainsheet, see Figure 1. SailBot utilizes a rigid sail assembly instead of a traditional cloth sail. The rigid sail assembly consists of three main components: the mast, the mainsail, and a trim tab (Figure 2). Unlike the mast of a traditional sailboat, the mast for a rigid sail is freely rotating and does not have stays or additional supports. Additionally, the position of SailBot's rigid sail is controlled using a trimtab instead of a mainsheet.

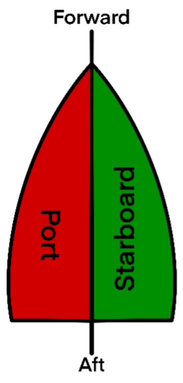


Figure 3: Diagram of port and starboard sides of a boat

The comprehension of the proper terminology of location and direction in a boat is critical in addition to understanding each part/system and its function within a sailboat, see Figure 3. The bow of the boat is the front of the vessel, and the stern is the back. To explain relative positions in a boat, the terms forward and aft are used, instead of in front of or behind, respectively. Forward refers to anything near the bow within the reference frame of the boat, and aft refers to anything near the stern. While in open water, using the terms right and left can be arbitrary, and therefore are not detailed descriptors of position. Instead, the terms starboard and port are used to describe the position. Starboard refers to the right-hand side of the boat (when facing the bow), and port refers the left-hand side.

2.1.2 Points of Sail

To successfully operate a sailboat, the position of the boat relative to the direction of the wind and the corresponding optimal sail trim must be understood. These relations are summarized by the six points of sail. The first point of sail is the no-go zone also referred to as being in irons or the no-sail zone (Figure 4). The no-go zone is too close to the wind to sail effectively because the sail cannot generate enough lift in the desired direction. To sail to a point that is directly into the wind, a method of zigzagging across the no-go zone must be utilized. Sailing towards the direction in which the wind is blowing is referred to as upwind sailing. Upwind sailing includes two points of sail: Close Hauled and Close Reach (Figure 4). While sailing on these two points of sail the sails are trimmed in close to generate maximum lift. The next point of sail is a Beam Reach, this refers to the position when sailing across the wind (Figure 4). In this position the sails are trimmed “half-in half-out". The final two points of sail are used when sailing away from the wind, referred to as downwind sailing. Downwind sailing is in the direction in which the wind is blowing. The two points of sail included in downwind sailing are Broad Reach and Running (Figure 4). The sails are trimmed most or all the way out when on these points of sail.

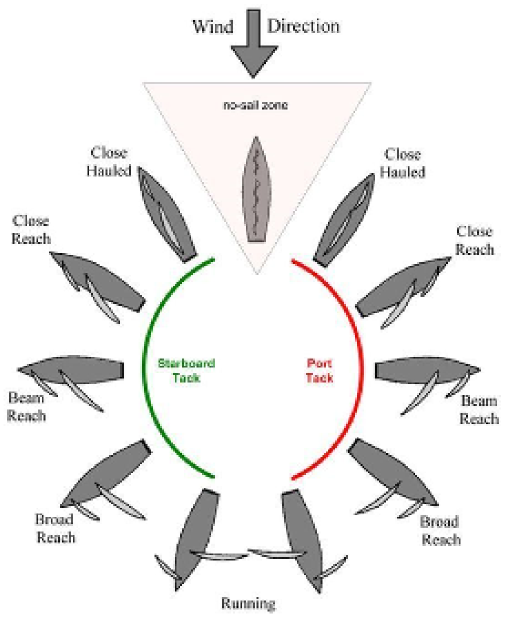


Figure 4: Points of sail

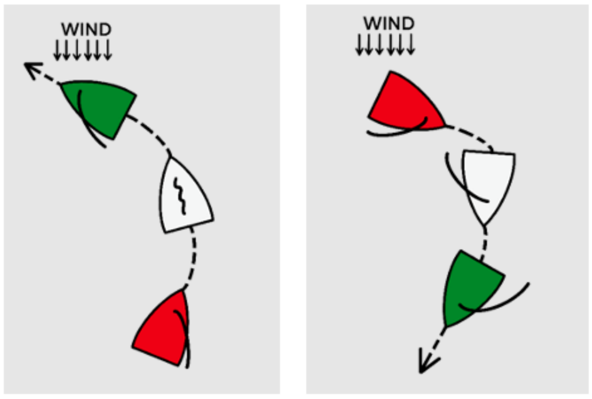
The points of sail are symmetrical across the wind, regardless of what side of the boat the wind is coming across. The side of the boat over which the wind is coming from is referred to as a tack. A port tack is when the wind is coming over the port (left) side of the boat, and a starboard tack is when the wind is coming over the starboard (right) side of the boat. The port tack is shown with a red semi-circle in Figure 4 and the starboard tack is shown with a green semi-circle in Figure 4.

The final concept to understand is windward versus leeward. Windward means upwind, or the direction from which the wind is blowing. A windward vessel refers to the vessel that is upwind of the other vessel, the leeward vessel. Leeward refers to downwind, or the direction opposite to the way that the wind is blowing from.

2.1.3 Tacking and Jibing

There are two basic turning maneuvers when operating sailboats: tacking and jibing. Understanding the difference between the two techniques requires an understanding of the wind direction and the points of sail stated above in section 2.1.2. Both tacking and jibing are used to change the tack of the boat, the side of the boat that the wind is coming over, which consequently switches the side of the boat the sail is on. Tacking refers to the maneuver in which the bow of the boat turns through the wind. Tacking is mainly used when sailing upwind while close-hauled or on a close reach. Jibing refers to the maneuver in which the stern of the boat passes through the wind. A jibe is used to maneuver the boat when sailing downwind, either on a broad reach or while on a run.

A tack is demonstrated in Figure 5a, in which the boat changes from a port tack (red) to a starboard tack (green) while sailing upwind. The boat is initially sailing on a port tack close-hauled (red), then the bow passes through the wind (white), and the boat continues sailing on a close-hauled starboard tack (green). A jibe is demonstrated in Figure 5b, in which the boat changes from a port tack (red) to a starboard tack downwind (green). The boat is initially sailing on a port tack on a broad reach (red), then the stern of the boat passes through the wind (white), and the boat continues sailing on a broad reach starboard tack (green).



a) Tacking b) Jibing

Figure 5: Tacking vs jibing

2.1.4 Rights of Way

While operating any vessel it is imperative to understand the boating right of way rules, comparable to “rules of the road” on land. Right-of-way rules are specifically designed maneuvering regulations for the purpose of avoiding collisions between vessels. The five main right of way rules are as follows (*Understanding Boating “Right of Way” Rules*, 2021):

1. Vessels under sail have the right of way over powerboats (with some exceptions).
2. When crossing, the boat on the right (approaching from starboard) has the right of way.
3. When meeting head-on, each vessel must alter course to starboard.
4. Any vessel overtaking another must keep clear of the stand-on vessel.
5. When approaching another vessel whose intentions aren’t clear, take evasive actions early.

The five rules stated above must be understood by all vessels. In addition to the general boating right of way rules, there are specific right of way rules that sailboats must also adhere to. During the IRSR vessels will be exposed to situations in which two sailboats meet, in which the following rules will apply (*Understanding Boating “Right of Way” Rules*, 2021):

1. The boat on the starboard tack (wind coming over the starboard side) has the right of way.
2. When two vessels are on the same tack (wind is coming from the same side), the leeward (downwind) boat has the right of way over the windward (upwind) boat.
3. When on the same tack in a passing situation, the vessel being overtaken has the right of way always.

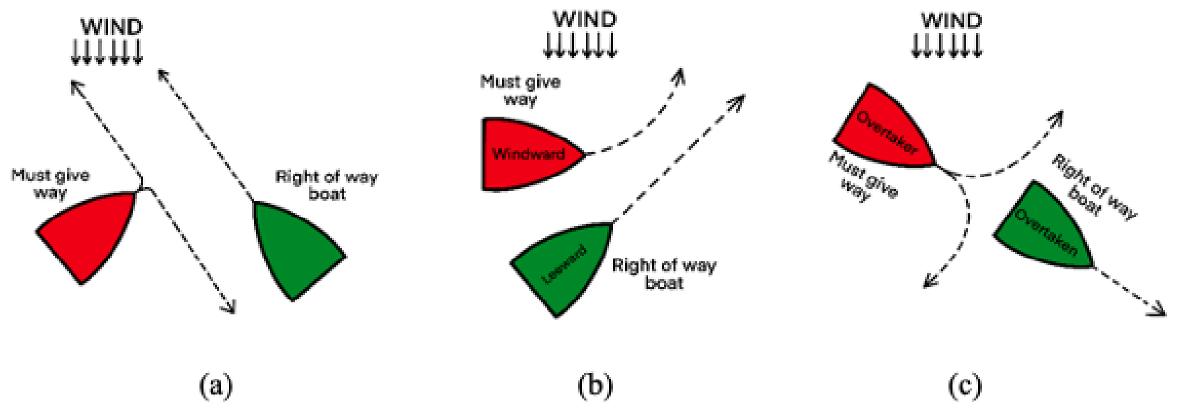


Figure 6: Examples of right of way scenarios

Figure 6a demonstrates sailing rights of way rule 1, in which the starboard tack boat (green) has right of way over the port tack boat (red) which must give way by heading off or tacking to avoid.

Figure 6b demonstrates sailing rights of way rule 2, in which two boats are on the same tack and the leeward boat (green) has the right of way over the windward boat (red). Therefore, the windward boat must head up or tack to avoid the leeward boat. Figure 6c demonstrates sailing rights of way rule 3, in which two boats are on the same tack and in a passing situation. The overtaking boat (red) must give way to the boat that is being overtaken (green) by heading up or down.

The comprehension of the eight rules stated above is crucial to maintain safe boating conditions for all vessels, especially during the IRSR.

2.2 Sailing a Robotic Sailboat with a Rigid Wingsail

This section presents the physics of sailing with a rigid wingsail. For most points of sail, our robotic sailboat is propelled through the water predominantly due to the lift generated by the rigid wingsail. The total aerodynamic force on a wingsail has two components – lift, which acts perpendicular to the wind direction, and drag, which acts in the same direction as the wind (Figure 7). Drag tends to push the sail downwind, which is sideways on most points of sail. However, when sailing on a run, drag is the predominant force and it pushes the sailboat forward since the direction of travel is directly downwind. It should be noted that SailBot's mast is designed to freely rotate, such that when the boat travels at an angle with the apparent wind, the wingsail will tend to rotate until the angle of attack (AoA), the angle between the apparent wind and the chord line, reaches zero degrees.

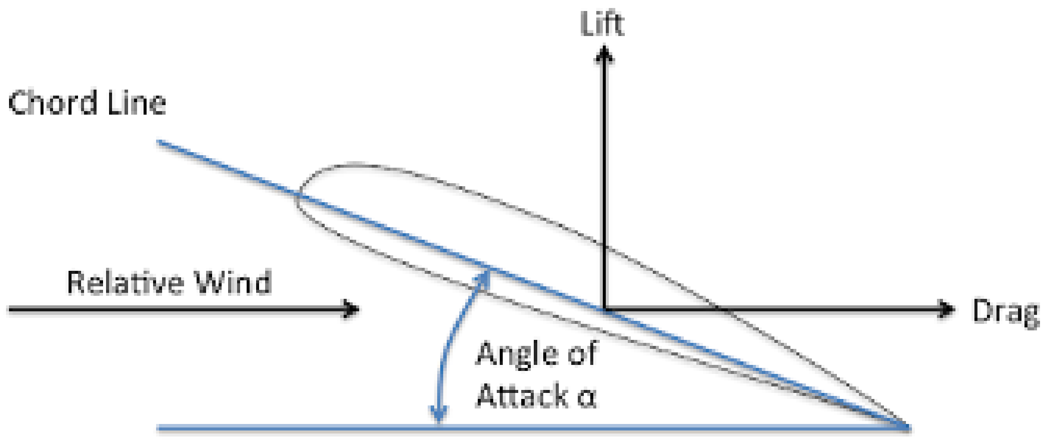


Figure 7: Diagram of angle of attack, lift vector, and drag vector on an airfoil

SailBot's wingsail is fitted with a trim tab, a smaller secondary symmetrical foil that is mounted midway up the mast at the end of two long booms that extends beyond the sail. This airfoil, like the airfoils in the wingsail, is also subject to lift and drag forces based on its angle of attack with the apparent wind. Its purpose is to help the sail maintain its desired AoA. The trim tab does this by creating a counteracting moment opposing the moment created by the wind on the wingsail when the wingsail develops any AoA. A static equilibrium between these moments balances out the wingsail, keeping it at the intended AoA.

2.3 International Robotics Sailing Regatta

The IRSR, also known as the SailBot Competition, is a regatta consisting of seven different events designed to evaluate a robotic sailboat's ability to sail. The participating groups are divided into classes based on the dimensions of the boat. As it has in previous years, the WPI SailBot team will participate in the 2-meter SailBot Class for the upcoming SailBot competition in June of 2022.

2.3.1 Challenges and Rules

Our team evaluated the seven different IRSI events and ranked the them based on the strengths and weaknesses of the current (2020-2021) SailBot design and the areas of expertise of our team. The events listed below are a result of this evaluation and are ranked from highest to lowest priority. The event descriptions are based on the documents available on the official IRSR website (International Robotic Sailing Regatta, 2014).

- **Fleet Race**: Demonstrate remote-control capabilities by sailing the boat around a triangular path rounding two buoys. The boat must also pass between two buoys located at the start/finish line. Remote control must be used for steering during the event.
- **Endurance / Long Distance**: Demonstrate the durability and capability to sail by rounding 4 buoys on a square pattern with remote control and/or autonomous sailing.
- **Collision Avoidance**: Demonstrate the boat's ability to autonomously navigate and avoid collisions with other watercraft while traveling between two buoys.
- **Precision Navigation**: Demonstrate the boat's ability to autonomously navigate a course within tight tolerances by starting and finishing the course between two buoys that are close together (3 meters apart).
- **Station keeping**: Demonstrate the ability to remain close to one position, then leave the position when necessary, by staying within a designated area until asked to leave. The boat must leave the area within 30 seconds.
- **Search**: Demonstrate the boat's ability to autonomously navigate in a search pattern to recognize and locate an object.
- **Payload**: Demonstrate the boat's ability to carry weight.

2.4 Past Iterations of SailBot

This section describes the state of the WPI SailBot as left by the 2020-2021 SailBot team, as well as provides a brief history of the boat's key systems. This is summarized by three sub-sections: Mechanical Systems, Electrical Systems, and Software Systems.

2.4.1 Mechanical Systems

As noted under Section 2.1.1, SailBot utilizes a rigid wingsail, designed by the 2016-2017 SailBot team and implemented by the 2017-2018 SailBot team. Prior iterations of the project implemented a winch-based cloth sail. A rigid wingsail has several advantages over a cloth sail, including: greater lifting forces, increased level of control, decreased weight, and increased durability (Regan et al., 2017). The current sail consists of three components: the lower rigid wingsail, the upper rigid wingsail, and the trim tab assembly.

The lower rigid wingsail is rigidly attached to the mast, which slides into the hull. The trim tab controller and upper rigid wingsail slide onto the mast protruding from the lower rigid wingsail. The mast allows for free and full rotation. It is important to note that the 2020-2021 SailBot mast broke during water testing. Several factors contributed to the mechanical failure, but from the 2020-2021 SailBot team's analysis, it was primarily due to the diameter of the mast being too small for the stresses endured during operation (Burri et al., 2021).

The control box for the trim tab contained the mechanical and electrical systems necessary for operation. This system contained the following components: a splash proof servo which controlled the trim tab angle via a carbon fiber pushrod, an Arduino microcontroller with wireless connectivity to receive and execute commands from the hull, an encoder to read wind direction, a LiPo battery, and external LEDs for reading the status from the shore. Detailed information on the past electrical systems of the trim tab can be found in section 2.4.2.

SailBot's hull was constructed out of fiberglass by the 2016-2017 SailBot team. Three main mechanical systems are mounted to the hull: the ballast assembly, the keel, and the rudders (Burri et al., 2021). The ballast is an approximately 8-kilogram block of lead attached to a carbon fiber arm, and driven by an automotive window motor. The purpose of the ballast is to control the heel angle of the boat. The ballast assembly was first implemented during the 2017-2018 iteration of the boat, and since its implementation has been a source of trouble, namely due to over-rotation. In the past this caused the gears to disengage with each other, preventing the motor from being able to move the ballast back to operating range (Guzman et al., 2020). To prevent this, hard stops were implemented, but contact with these stops still damaged the gearbox due to over-torquing of the gears (Guzman et al., 2020). The 2019-2020 team made plans to improve this system by re-machining certain parts, attaching a potentiometer for control, and integrating hall-effect magnetic sensors to prevent hard stops, but these plans were not implemented due to the COVID-19 Pandemic (Guzman et al., 2020).

The keel is used for counterbalancing the forces from the mast and the heel of the vessel. The 2019-2020 team redesigned the keel to be stronger, more structurally sound, and more hydrodynamic. However, with this redesign the keel cavity of the hull had to be modified to fit the new keel. SailBot utilizes a dual rudder system controlled by a single servo. The dual rudders are positioned at an angle to increase the effectiveness of the boat's steering such that when SailBot is at an optimal trim, one of the rudders is near perpendicular to the water's surface. Overall, the hull's mechanical systems keep the boat in good trim.

2.4.2 Electrical Systems

The electrical system on SailBot is essentially split into two distinct subsystems: the trim tab control system and the hull control system, see Figure 8 for an electrical schematic. The two systems previously communicated with each other via a WiFi connection. By keeping these components physically separated the mast can rotate freely without running the risk of twisting or damaging any wires. The trim tab system utilizes a small, splash-proof servo to move the trim tab and therefore control the AoA of the rigid wingsail. It is powered by a 3.7V Li-Po battery and is controlled using an Arduino MKR 1010 which was installed by the 2020-2021 SailBot team (Burri et al., 2021). The Arduino replaced the Teensy microcontroller which had been used prior. In addition to controlling the trim tab, the trim tab system is also wired to a wind vane mounted in front of the mast, which uses an encoder to sense the current AoA.

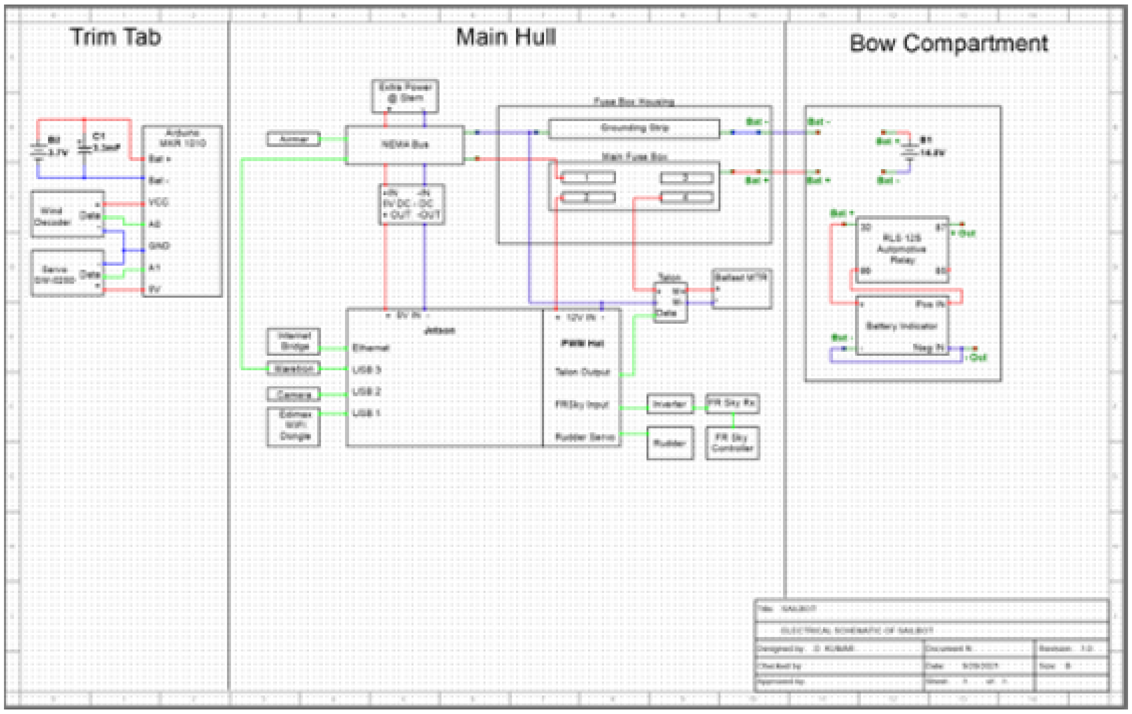


Figure 8: Schematic of SailBot's electrical systems (2020-2021)

The Arduino controlling the trim tab sends telemetry data to an Edimax wireless dongle which is plugged into a Jetson Nano. The Jetson serves as the boat's primary controller. The wireless link between the Jetson and the Arduino is also used in the reverse direction to send commands for updating the state of the trim tab. For instance, to reduce lift on the wingsail, the state could be set to the minimum lift condition. To determine the direction and speed of the apparent wind, as well as the boat's current pitch, roll, and yaw, a Maretron NMEA 2000 / USB Gateway converts NMEA 2000 messages sent by an Airmar WeatherStation 220WX (mounted on the stern of the boat) to readable messages that can be sent over a USB connection to the Jetson.

For communications with the shore, particularly for RC control, the boat has been using a 2.4 GHz FrSky X6R receiver to communicate with a FrSky X9D controller on land. Signals from the receiver pass through an inverter before going into the Jetson. The RC connection is reported to have an approximate range of about 1.5 nautical miles (Guzman et al., 2020). There is also a Digi Xpress 900 MHz Ethernet bridge on board that connects with a second bridge on land to enable wireless remote access to the Jetson. This is used to upload new code to the boat as well as to send back telemetry data. The connection has a reported maximum range (outdoor line of sight) of about 3.2 km when using a 2.1 dBi dipole antenna (Digi International, 2016). All these systems are powered by a 14.8 V LiPo battery located in the bow of the boat. The battery is fed through a magnetic on/off switch, then a fuse box, to provide steady and reliable power to all the electronics within the hull.

In addition to the electronics mentioned above, there are a few additional systems installed on the boat that are not currently in use. There is an e-paper screen installed by the 2017-2018 SailBot team for debugging purposes (Burklund et al., 2018) as well as LEDs located in the trim tab housing that are no longer connected. Various cameras, for machine vision, have also been used previously but are also currently disconnected.

2.4.3 Software Systems

ROS2

Several major changes have been made to the software of the robot in recent years, the largest of these changes was made last year, during the 2020-2021 SailBot iteration. The entire software environment was migrated over to ROS2, a framework for managing complex robotic software. Previously, the robot's software was divided broadly by whether it was designed for remote control or autonomous control. This design was useful, but after multiple years with different teams contributing to and altering it, the codebase began to become bloated and difficult to understand. This necessitated the move to a framework that allowed large projects to be easily maintained. ROS2 provides a large collection of tools to streamline communication between different software processes and ensure that all parts of large software projects receive the data they need. For example, using ROS2 would make it extremely easy to send telemetry data to both the debugging interface and the navigation algorithms, while ensuring that both systems are able to process that data at a compatible rate. To do this, the ROS2 architecture divides discrete pieces of code into "nodes" which can both publish certain kinds of data and listen to other nodes to react when a certain kind of data has been published. The previous year's team divided the code into the following seven nodes:

- Control System
- Debugging Interface
- Airmar Reader
- PWM Control
- Serial RC Receiver
- Trim Tab Comms

- High-Level Pathing

Each of the above listed nodes are described in greater detail in the succeeding paragraphs.

The Control System node is responsible for making decisions for the boat. This node first determines if the robot is in Full RC, Semi-Autonomous, or Autonomous mode based on input from the RC controller. If in Full RC mode, the node takes inputs from the RC controller and sends the appropriate states to the rudder, trim tab controller, and ballast. If in Semi-Autonomous mode, the Control System node takes control of the ballast and trim tab but leaves the rudders to be steered by RC control. In Autonomous mode the node takes full control of the robot and navigates using autonomous control algorithms.

The Debugging Interface node simply relays information about the different systems on the boat to a central dashboard. This gives users a better sense of what the current status of the boat is and helps debug any issues that might arise during testing of the boat.

The Airmar Reader node interprets data coming from the Airmar and distributes that data to other nodes. The node simply parses the data into a readable format and publishes it so that any node that needs the data can easily access it.

The PWM Control node listens for messages from the Control System node and handles setting the rudders and ballast to the appropriate state. It does this by controlling the servo driver that is responsible for moving the servo attached to the rudders and sending a PWM signal to the Talon SRX motor controller that controls the ballast.

The Serial RC Receiver node manages communication between the remote controller and the boat. This node parses the information sent from the controller over six channels, four of which represent the position of the joysticks and two of which represent the state switches. The joysticks are used to control the rudder, trim tab, and ballast while in full RC mode, and the state switches determine which mode the boat should be in. All this information is decoded from the RC controller and sent to the Control System node.

The Trim Tab Comms node, as the name implies, communicates with the Arduino controller in the wingsail. The appropriate trim tab state is received by the Control System Node, encoded, and sent to the Arduino wirelessly.

The High-Level Pathing node had not been fully implemented and tested but was meant to be responsible for interpreting telemetry data and finding the optimal path for the robot to take. This path was then to be used to determine the optimal position of the rudder, trim tab, and ballast while in Autonomous mode.

Telemetry Dashboard

Separate from the architecture for internally controlling the boat is the telemetry dashboard, which receives information from the boat while sailing and displays it in a readable format. The robot uses the 900 MHz Ethernet bridge to send telemetry information to a server hosted on land. The dashboard itself is a webpage that overlays the robot’s position onto a map of the area taken from

Google Maps. The dashboard can also be used to view the map for the robot's high-level pathfinding, and even save a map onto which future pathfinding tests can be performed.

Autonomous Sailing Algorithms

The previous years' autonomous sailing algorithms on the boat could be divided into high-level navigation, low-level navigation, and a ballast control algorithm. The high-level navigation algorithm began by taking a map of the sailing area, specifically Lake Attitash, and dividing it into a navigable grid, filtering out any areas that are unreachable due to obstacles such as shallow water, large rocks, etc. The algorithm then determined the current true wind direction and current robot orientation, which it used to determine the difficulty of traveling in each direction. The robot then used an A* algorithm to determine the best path to take between given targets. By taking the wind direction and current momentum of the boat into account, the algorithm produced a path that planned tacks in advance. Several key factors, such as taking into account wind direction, had not been implemented and the system as a whole had not been tested in a real-world capacity.

The low-level navigation algorithm was designed to navigate the robot between the intermediate points generated by the high-level navigation algorithm. This algorithm attempted to move from point to point by controlling the rudders and trim tab. The algorithm itself was a simple decision tree that checked for any changes in wind direction and the relative position of the target point and incrementally adjusted the rudder and trim tab accordingly. This algorithm did not attempt to calculate an optimal position for the rudders, and would instead move them incrementally until the boat was oriented correctly. This resulted in very slow turns which made it difficult to maintain a heading under strong wind or waves.

The ballast control algorithm was designed to keep the boat in proper trim during autonomous sailing. This algorithm worked by incrementally adjusting the position of the ballast as the robot's measured roll changed. If the ballast reached its maximum position and was still unable to control the roll of the boat, the trim tab was told to reduce lift until the boat could be stabilized.

2.5 Design Challenges

In this section we discuss issues that existed in the systems on the SailBot 2020-2021 boat.

2.5.1 Wingsail Assembly

At the start of this project, SailBot did not have a functional wingsail. The previous mast broke due to several factors including an unsecure mounting mechanism and the physical design of the mast. The failure was primarily due to the diameter of the mast being too small for the stress endured during typical operation. Improvements to the mast require improvements to the whole wingsail assembly, and therefore the design of the wingsail as a whole must be reworked.

In addition, the trim tab was an unreliable system that had presented many issues for past SailBot teams. There were issues with the hull-to-wingsail communications, the current draw of the servo, and the lack of stability of the trim tab pushrod.

2.5.2 Ballast Assembly

The ballast had been a continual source of frustration since its implementation. The main issue was its tendency to over-rotate. The over-rotation of the gearbox damaged both the gear box and the motor, increasingly lowering the system's reliability over time. As of three years ago, hard stops prevent over rotation, but damage to the system still occurred. It was necessary to improve the ballast mechanism both mechanically and through software updates before serious damage occurred.

2.5.3 Hull Electrical Systems

The housing units for electrical components were unsecured within the hull of the boat. Therefore, items were free to shift around inside, potentially impacting the boat's weight distribution and damaging the electrical systems and wiring. The electrical components and wires had visible wear that could cause problems during their operation. In addition, the electrical schematics of the current state of SailBot were missing. Lastly, there were electrical components in the hull that are not being utilized and thus could be removed.

The only indication of the battery's charge was when the boat stoped working or when the LED on the old magnetic switch, previously removed, was off. A battery indicator would be better suited to give a visual representation of the main battery capacity by sending a reading through the Jetson to the land-based team.

In previous years, an e-paper screen had been used for quick debugging and to check the overall status of SailBot. It was connected to the primary microcontroller and used to display debugging information which partially eliminated the need to connect directly to the controller to get information. The SailBot team had run without the screen for a few years, however it was a useful way to check in on the condition of SailBot quickly.

2.5.4 Navigation Algorithms

At the start of this project, the autonomous navigation algorithms had not been tested and most were only partially completed. The low-level controller was only able to move the rudders in set increments and could not react to large changes in desired heading. The A* algorithm was missing the ability to weigh headings based on wind direction, which significantly hindered its ability to generate sailable paths. The point-to-point navigation system was very straightforward and assumed that the path created by the A* algorithm was sailable. If the boat drifted off course or the wind direction changed significantly, the point-to-point system would rerun the A* algorithm, meaning the entire path between targets would be repeatedly recalculated.

2.5.5 Computer Vision

Computer vision has been an idea proposed by previous teams for several years. The appeal of a computer vision system is the ability to detect and precisely navigate around buoys during competition rather than relying on their approximate GPS coordinates and dead reckoning. Previous teams have attempted to implement a computer vision system but have not been successful. As such, previously existing computer vision systems have not had robust buoy detection software and were not able to act upon the detection of an object.

3 Purpose and Project Goals

The purpose of the SailBot 2021-2022 MQP is to continue the development of the autonomous sailboat started through the efforts of five previous MQP projects. We established a series of overarching goals to set standards for SailBot's performance. Below are our four primary goals for the 2021 - 2022 academic year.

- The vessel can maintain control of its basic sailing functions for at least 7 hours.
- A human operator can assume control of the vessel within 1 nautical mile at any time during the vessel's operations.
- The vessel can plan a path and maintain course in winds below 15 knots.
- The vessel can autonomously detect and maneuver around buoys and obstacles.

Completion of these goals will demonstrate SailBot's ability to handle the unpredictable environment on the water, while contributing to our completion of IRSR challenge objectives.

4 Electromechanical Design and Implementation

To achieve our project goals, we have focused our design and implementation efforts on three key electromechanical systems: the wingsail assembly, the ballast assembly, and the hull electrical systems.

4.1 Wingsail Assembly

The mast of the previous SailBot 2020-2021 broke during water testing, demonstrating that the design was venerable to the forces experienced during normal operation. The mast is critical to the function of the wingsail, and the wingsail is the main propulsion system for SailBot. Therefore, without a structurally sound mast, SailBot was inoperable, and it was imperative that we replace the mast. Implementing a new mast led to a complete redesign of the wingsail assembly. The updated wingsail assembly includes a new mast, an updated trim tab assembly, and updated trim tab electronics.

4.1.1 Mast Replacement

The first step in the wingsail redesign was the replacement of the wingsail mast. To calculate the loading conditions of the mast during operation, we first determined the conditions in which the mast would experience maximum loading. This was determined to be when the boat's heel angle is high, resulting in greater forces from the sail than the system's counteracting forces. The forces on the sail are countered by the keel, the ballast, and the hull of the boat as seen in Figure 9. The hull, ballast, and keel attempt to keep the mast upright as the wind blows against the sail. All these forces are focused at the point where the mast connects to the hull and act as a large lever arm.

To determine the maximum loading condition the mast experiences, the moments on the vessel were calculated and then verified through testing. Using a 3D model of the hull, the displacements of the hull and the center of mass at different heel angles were calculated using the SolidWorks analysis tools. Using the data from the SolidWorks analysis, the righting moment on the hull was determined for a range of heel angles from 10° to 70°, with the maximum righting moment occurring between 30° and 40°. The moments caused by the keel and the ballast were determined using the weight of each respective system at each heel angle. To determine the maximum moment the mast would experience, the moments of the keel, hull and ballast were summed at heel angles of 20°, 30°, 40°, and 50°. The results are tabulated in Table 1.

From these calculations, the maximum moment experienced by the mast was determined to be 98.37 ft-lbs at a heel angle of 40°. To validate these results, an in-water test was conducted in WPI's Sports and Recreation Center pool. During the test, we gathered data on how much force against the mast was required to hold the boat in static equilibrium and observed approximately where the waterline was on the boat at different heel angles. The results of the waterline test were used to validate the estimated waterline in the CAD model of the hull. The results from the

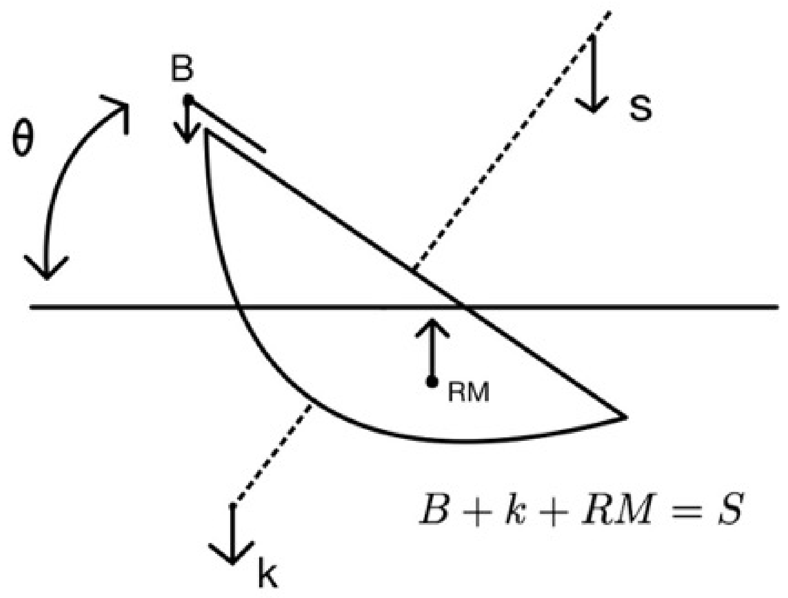


Figure 9: Diagram of the moments on the boat at a heel angle θ

Table 1: Sum of Moments at Different Heel Angles

Heel Angle	**Sum of Moments**
20	77.14 ft-lbs
30	95.56 ft-lbs
40	98.37 ft-lbs
50	95.42 ft-lbs

testing of the forces required to hold the boat in static equilibrium are tabulated in Table 2. The force reading was calculated using a spring scale positioned on the mast 93 inches from the deck of the boat. Data from this testing was also utilized in calculations for the ballast system, see section 4.2.

Table 2: Force Measured to Maintain Static Equilibrium in Different Loading Conditions

Loading Condition	**Measured Force**
$\sim$ 25-degree heel angle	7 lbs
$\sim$ 25-degree heel angle with 15-pound ballast	10 lbs
Max recorded force ($\sim$ 40-degree heel angle)	11.5 lbs

The maximum recorded force of 89.13 ft-lbs was then used to determine the maximum moment that the mast would experience during operation. This value is a bit less than the calculated maximum moment of 98.37 ft-lbs, but still ended up being fairly close. Therefore, we concluded that our water testing validated the calculated results. The maximum moment was then used to determine the required dimensions of the mast to ensure safe operation. The new mast segment was selected from a varying diameter 80% carbon fiber composite mast with a minimum diameter of 1.15 in, maximum diameter of 2 in, length of 8 ft, and an overall wall thickness of 0.12 in.

To determine which segment of the new reduced diameter mast should be selected to withstand the calculated maximum loading conditions, a bending stress calculation was conducted. A conservative estimate of 200 MPa (29,007.5 psi) for the yield strength of the 80% carbon fiber composite mast was utilized for the calculations. The desired factor of safety for this application should be at least 2, and any value less than 1 is an unacceptable safety factor for the new mast. The first calculation was done using the maximum diameter of the new mast, which results in a factor of safety of 19.3. This safety factor value greatly exceeded our goal of 2. Therefore, another calculation using the minimum diameter of the new mast was done to determine if any section of the mast could be selected. The resulting factor of safety using the minimum diameter was 3.7. We therefore concluded that any segment of the new tapered carbon fiber mast would adequately withstand the maximum loading conditions with a factor of safety greater than 2. In comparison, the previous team's straight mast had a diameter of 0.73 in and a wall thickness of 0.12 in, which results in a factor of safety of 0.9. The result of a safety factor less than 1 for the old mast confirms the hypothesis that the diameter was too small for the loading conditions and contributed to the mechanical failure.

4.1.2 Trim Tab Mechanical Redesign

The preexisting wingsail assembly was assembled by sliding together three sections: the lower rigid wingsail, the trim tab assembly, and the upper rigid wingsail. The trim tab assembly was a single unit consisting of an electronics housing, a splash-proof servo, a trim tab pushrod, one boom and the trim tab airfoil. It was mounted to the wingsail by sandwiching the electronics housing between the lower rigid wingsail and the upper rigid wingsail, see Figure 10. The entire assembly was unstable during operation due to the aforementioned mounting strategy, single trim tab boom, and thin pushrod. This instability introduced critical amounts of torsional stress and linear oscillation to the entire wingsail assembly, making a redesign critical. The new trim tab mechanical system includes the redesign of both the trim tab driving mechanism and the mounting system.

Trim Tab Driving Mechanism

The trim tab in the preexisting wingsail assembly was driven by a thin hollow carbon fiber pushrod mounted parallel to the trim tab boom. One end of the pushrod connected to the servo inside the trim tab housing near the wingsail mast, and the other end connected to a connecting piece that offset it from the pivot point on the trim tab, see Figure 11. The pushrod flexed significantly during operation, negatively impacting the mechanical performance of the servo. The initial consideration for improvements to the trim tab mechanical operation was to increase the rigidity of the trim tab pushrod. The team conducted a preliminary analysis of the loads experienced by the pushrod during operation. The analyses concluded that to achieve the desired mechanical prop-

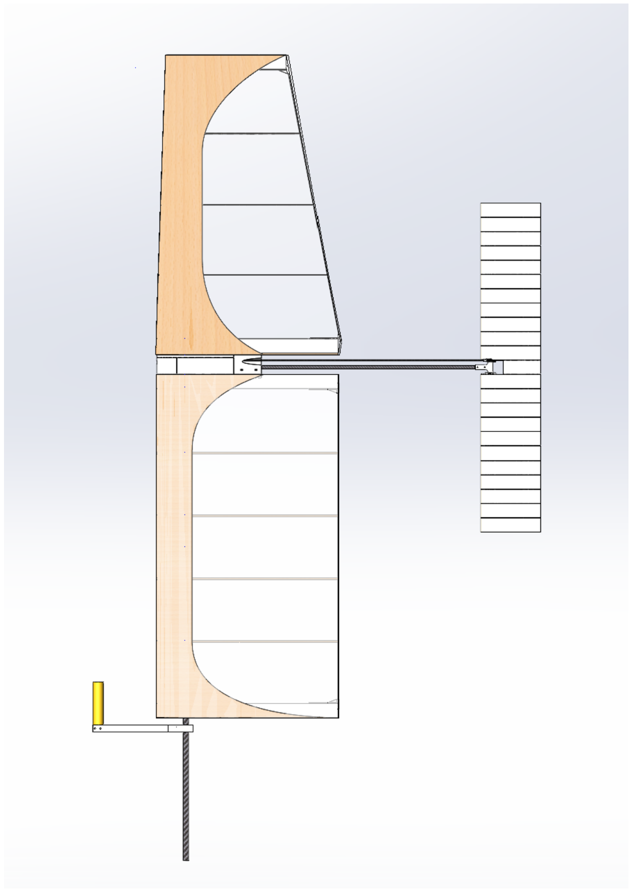


Figure 10: SailBot 2020-2021 wingsail assembly

erties, the diameter of the pushrod would need to increase drastically and the tube would likely need to be constructed out of solid carbon fiber. Applying these design changes would increase the weight of the trim tab pushrod, and only marginally improve performance. Therefore, the desired structural rigidity of the pushrod could not be achieved within the design constraints of the wingsail assembly - specifically the need for the assembly to be as lightweight as possible for proper operation.

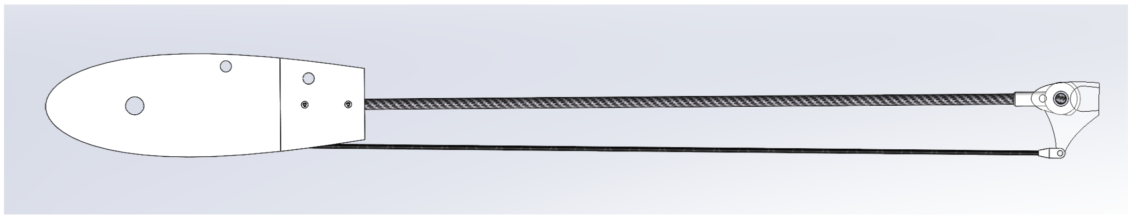


Figure 11: Old pushrod trim tab driving mechanism

After determining that the existing trim tab pushrod could not be replaced to increase the structural rigidity, we began considering alternative trim tab driving mechanisms. Our initial plan was to shorten the pushrod by mounting the pushrod assembly closer to the trailing edge of the main wingsail or by creating a linkage of shorter rods to reduce the flex on each individual component. However upon analysis, the benefits of these initial considerations did not outweigh the costs to both weight and the stability of the driving mechanism. The next concept involved eliminating the pushrod and linkages from the trim tab assembly entirely. Eliminating these additional driving components was accomplished by directly driving the trim tab using the existing splash-proof servo. The benefits of a directly driven trim tab assembly include increased control, reduced weight, and lessened torque requirements. The servo connects directly to the trim tab mast at the aerodynamic center of the trim tab airfoil as shown in Figure 12.

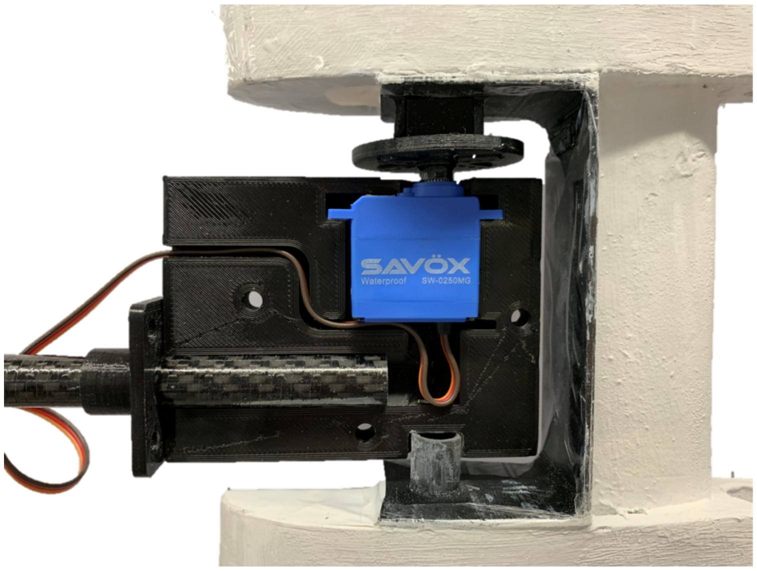


Figure 12: New trim tab driving mechanism

The main concern for relocating the servo to directly drive the trim tab was that the weight of the servo may impact the balance of the entire wingsail system. To minimize this potential imbalance, only the servo was moved to the end of the boom rather than all of the trim tab electronics. The trim tab electronics housing was relocated to the base of the main wingsail assembly. Overall, the integration of a new driving mechanism was manageable since the entire trim tab mounting system was being reevaluated and redesigned concurrently.

Trim Tab Mounting System

The redesign of the trim tab mounting system was driven by a desire to increase stability, improve modularity, and make it easier to access the trim tab electrical systems. As stated previously, the preexisting trim tab was mounted to the wingsail using one boom that was rigidly fixed to the trim tab electronics housing (Figure 10). The housing slid onto the main wingsail mast and was sandwiched between the upper and lower rigid wingsail components. This mounting system connected the trim tab to the wingsail with only one point of contact, but did not directly connect the trim tab boom to the wingsail mast. The previous design resulted in a complicated assembly-disassembly process and the components did not slide together easily. Based on the design review of the old mounting system, the new design included multiple connection points, a direct connection between the trim tab boom and the wingsail mast, an intuitive assembly-disassembly process, and added torsional support. These desired features were implemented with a modular two boom mounting system as seen in Figure 13.

The two boom mounting system increased the modularity of the full wingsail assembly by sep-

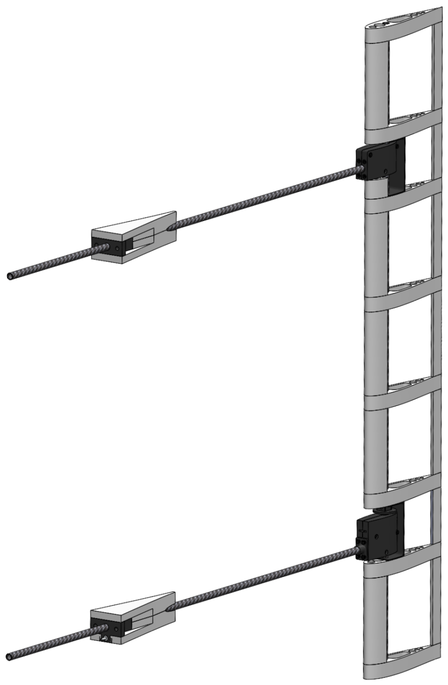


Figure 13: Two boom trim tab mounting assembly

arating the trim tab into five distinct components: the upper boom, the lower boom, the upper connector, the lower connector servo housing, and the trim tab airfoil assembly. The upper boom and lower boom slide into the main wingsail and into a connector rigidly attached to the wingsail mast. The booms provide a direct connection between the trim tab airfoil assembly and the mast of the main wingsail. The upper and lower connectors serve as the joints between the trim tab boom and the trim tab mast. The upper connector provides support for the upper section of the trim tab airfoil assembly. The lower connector supports the lower section of the trim tab and also serves as the housing for the splash-proof servo, see Figure 12. The upper and lower connectors clamp together for easy assembly-disassembly. The clamp design of the connectors also allows for the replacement of the connector if mechanical wear occurs, or if the servo needs to be replaced. The trim tab airfoil assembly remains mostly unchanged from the previous design - all the overall dimensions are the same. However, the airfoil material was changed from balsa wood to foam for increased rigidity, and the placement of the airfoils were shifted to accommodate the new connections. The utilization of two booms reduces the torsional forces acting at each connection point with the wingsail, and reduces play in the overall assembly.

4.1.3 Trim Tab Control System

Along with the revamp of the trim tab mechanical systems, the trim tab electrical control systems were also overhauled. The new control system featured a more modular design focused on ease-of-use and increased reliability. A schematic of the new system can be seen in Figure 14.

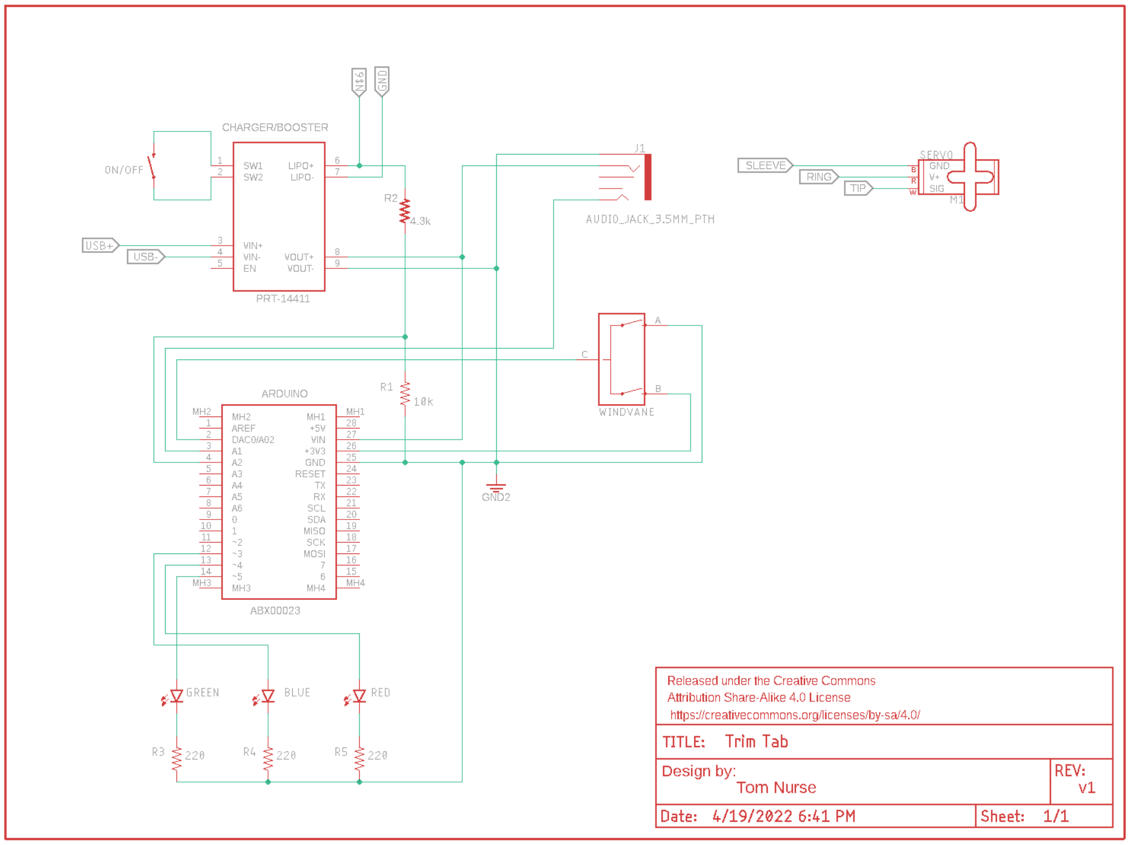


Figure 14: Schematic of the trim tab control system circuitry

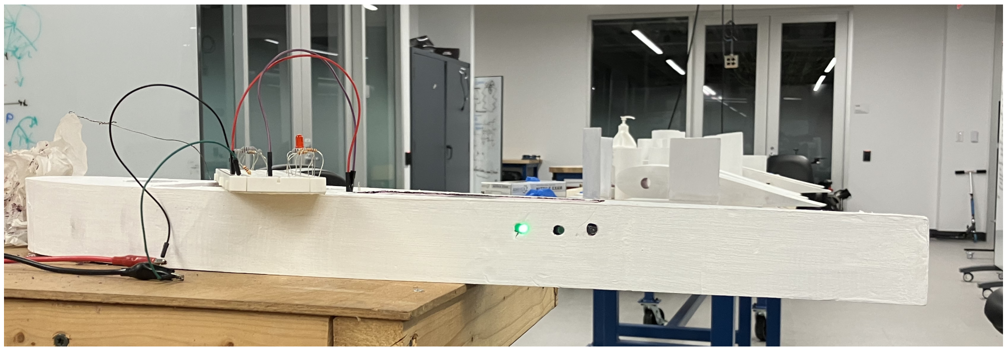


Figure 15: Trim tab LED state display

The trim tab electronics continue using the Arduino MKR 1010 as the main controller since wireless connectivity is already built in. On top of the Arduino we added a protoboard which provides pins for each of the system's inputs and outputs. The circuit includes a voltage divider for measuring the voltage of the LiPo battery, which is used for determining the approximate battery levels. This is all powered through a SparkFun LiPo Charger / Booster. The charger / booster circuit provides the system with up to 5 W (5 V at 1 A) sourced from a 3.7 V LiPo battery. It also enables users to recharge the battery without the need to take the battery out of the housing. The new controller can be charged by plugging a micro-USB cable into the port at the base of the wingsail. This port can also be used to reprogram the Arduino or for debugging purposes. Finally, a water-proof rocker switch was wired into the charger / booster in order to serve as the system's power switch.

The wind vane, LED state display, and trim tab servo all connect to the controller via removable wire connectors. This allows the controller to be removed from the wingsail entirely if needed. The new LED state display uses red, green, and blue LEDs to indicate the current state of the trim tab system (Figure 15). The LEDs are embedded inside the wingsail and use a collection of 6 fiber optic tubes to redirect light towards the sides of the sail as seen in Figure 16. This allows us to view the indicators from both sides of the wingsail with only 3 LEDs.

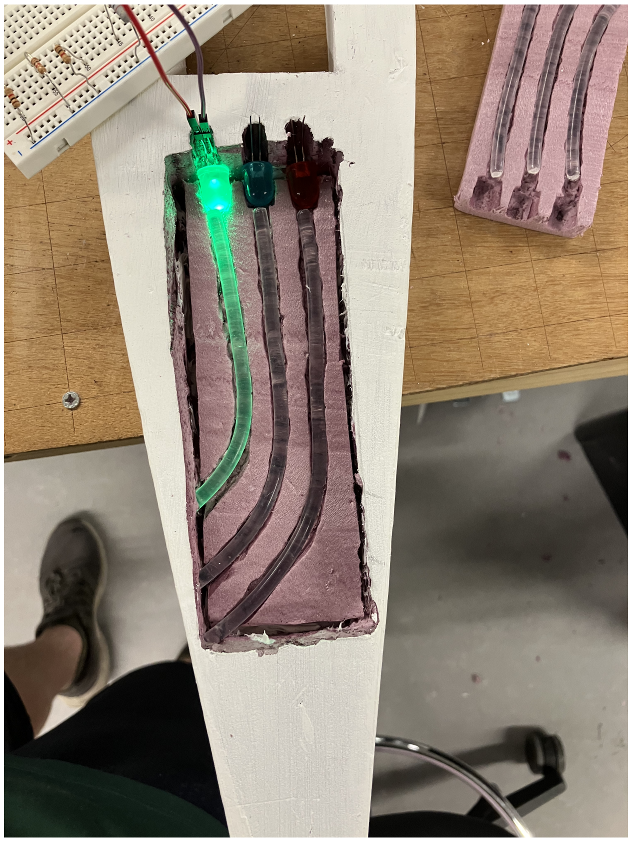


Figure 16: Fiber optic tubes used to redirect light from the 3 status LEDs for the trim tab

As described above in sub section 4.1.3, the new trim tab servo is located at the end of the bottom trim tab boom. The servo sits in the bottom connector inline with the mast and allows it to drive the trim tab directly. To control the servo, wires run up through the main wingsail and along the bottom boom. At the point where the boom meets the main sail, a TRS connector (audio jack) connects the two wires. This allows for easy, almost fool-proof setup and disassembly while ensuring a rigid connection.

4.1.4 Trim Tab Communication System

To determine which method of communication works best with the trim tab, research was conducted on several different wireless personal area network (WPAN) technologies. We compared the performance of Bluetooth Low Energy, Zigbee, infrared, and ultrawideband to see which options provide the best reliability and power efficiency. Ideally, a WPAN with at least 95% certainty of coverage at a distance of 5 meters, the maximum allowable height of the sailboat as defined by IRSR rules. We also researched various methods of hard wiring the trim tab to the hull through the use of either a slip ring or a tip, ring, sleeve (TRS) type connector.

To estimate and compare the expected coverage between these technologies, we modeled the wireless propagation characteristics of our environment. The results of our analysis are displayed in Figure 17, with the X axis representing the distance on a logarithmic scale. From these results, we determined that the distance power gradient (α) was about 3.4 - very close to what is expected for an ideal indoor obstructed line of sight (OLOS) environment (α = 3.5). The results are applicable for the trim tab communications because the connection between the Jetson and the Arduino is partially obstructed by the hull and the wingsail. Using this model, we were then able to calculate and compare the expected coverage for each of the technologies. We found that the expected coverage for Bluetooth Low Energy (BLE) came the closest to meeting the desired connectivity. The range of coverage (with 95% certainty of coverage) was around 6.5 meters. BLE is also easy to implement and much more power efficient than the alternatives, making it our top choice for replacing WiFi. Once we selected BLE, we replaced the existing WiFi implementation and performed an integration test to verify that our new system worked as expected.

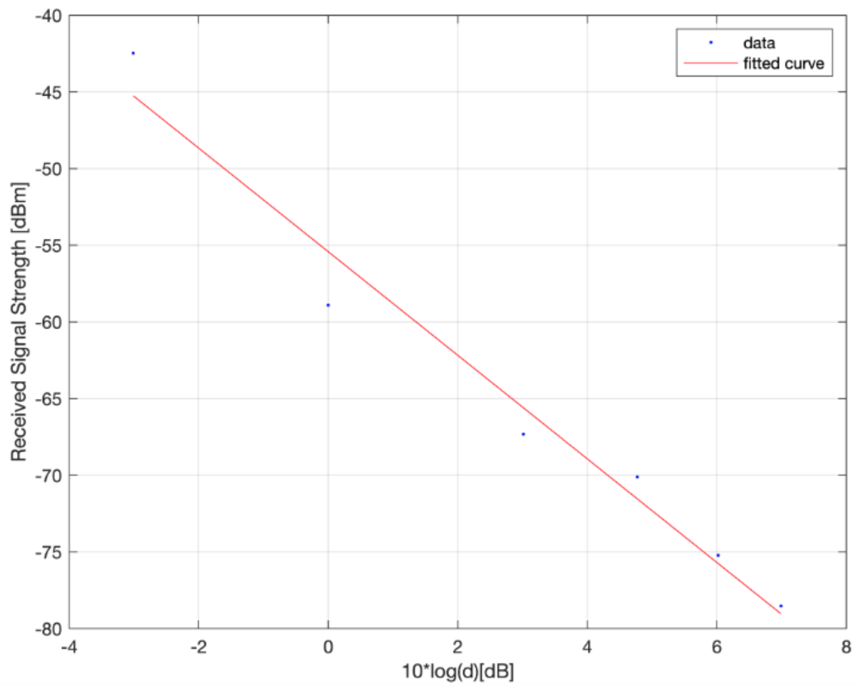


Figure 17: Wireless propagation model for the connection between the Jetson and the Arduino

4.1.5 Wingsail Redesign and Rebuild

The mast of the previous wingsail broke before performance data was collected, therefore all design assumptions had to be reevaluated. The first design assumption was the required sail area. The required sail area can be calculated based on the displacement of SailBot's hull. The guidelines for interpreting the ratio of sail area to displacement are as follows: a ratio below 16 indicates an under-powered boat, a ratio of 16-19 indicates good performance, a ratio of 20-22 indicates high performance, and any ratios over 22 indicates super-high performance (Doane, 2019). Therefore a ratio of 17.5 was chosen to determine the optimal sail area of the new wingsail assembly. The following equation was used to determine the sail area:

$$SailArea = \frac{(ratio * (\frac{displacement}{w_s}))^{2/3}}{a}$$

where the displacement of the boat is 120 lbs, w_s is 64, the weight in pounds per cubic foot of salt water, and a is the adjustment factor for a rigid wingsail, 1.83 (Silva, 2019). Using the equation above a total sail area of 14.5 square feet was determined to be preferable for SailBot.

The profile of the wingsail airfoil is the same as the airfoil selected by the 2016-2017 Robotic Automated Wingsail MQP team (Regan, 2017). The 2017 team selected a Joukowsky 18 airfoil, which is a symmetric airfoil with a maximum thickness at 18% of the chord length. A symmetric airfoil is beneficial for the SailBot wingsail as it must be able to generate lift from both sides equally. Symmetric airfoils also provide a good lift to drag ratio for a wide range of velocities, and produce no lift at a zero angle of attack.

Utilizing the desired sail area of 14.5 square feet and the Joukowsky 18 airfoil profile as a starting point, we determined the approximate dimensions and profile of the wingsail. In traditional boats, the sail area can be adjusted based on the wind conditions, to mimic this capability with a rigid wingsail, we included a removable top sail section. The main wingsail is designed to be optimal for moderate or high wind sailing in 10 to 15 mph winds, and the removable section is optimal for low wind sailing in winds under 10 mph. The main wingsail has a sail area of 10.5 square feet and the removable section has a sail area of 3.5 square feet. Therefore, the overall sail area utilized in the new wingsail is approximately 14 square feet, with an additional 2.5 square feet added by the trim tab.

The airfoils on the main wingsail have a uniform profile for the bottom two thirds of the sail and then taper inwards for the top third of the sail. The chord length of the uniform airfoils are approximately 24 inches, and the tapered section shrinks to a chord length of 21 inches. The height of the main wingsail is 5 feet, 2 inches. The removable section has a fully tapered profile. The maximum chord length is 21 inches and tapers down to a chord length of 15 inches. The height of the removable section is 2 feet 4 inches. The combination of the two wingsail sections results in an overall height of 7 feet 6 inches.

Next, we decided on the placement of the airfoils on the main wingsail using the two boom trim tab mounting assembly design as a starting point. The trim tab is centered with respect to the main wingsail, which dictated the placement of the airfoils within the wingsail. Each airfoil in the

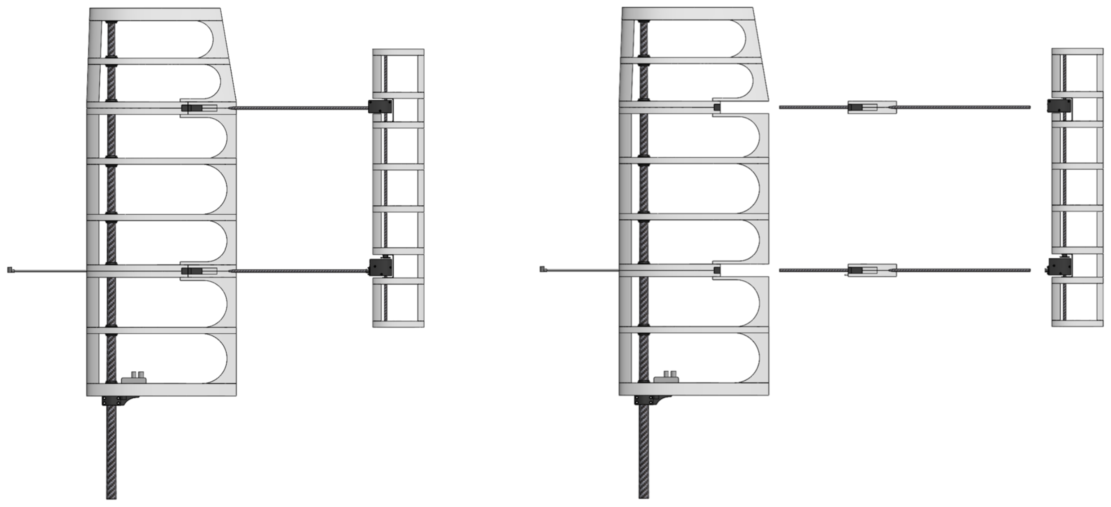


Figure 18: New wingsail assembly: standard view (left) and exploded view (right)

main wingsail is separated by approximately 7 inches. The final placement of the airfoils is shown in Figure 18.

The two mounting airfoils within the wingsail that connect it to the trim tab assembly are made up of four components. The mounting assembly consists of two foam airfoils with channel cutouts for the boom. Glued between the foam pieces are two different 3D printed alignment and attachment pieces. The first is a L-shaped mount, glued to the mast, that the end of the trim tab boom slots into. The second is an alignment piece mounted near the trailing edge of the airfoil. This alignment piece matches another mounted rigidly to the middle of the trim tab boom, as seen in Figure 13. The two alignment pieces are fastened together to form a stable mount. The wingsail fully assembled and in an exploded, dissembled view is shown in 18.

As stated, previously in subsection 4.1.2, the electrical housing was relocated to base of the main wingsail assembly, see Figure 19. The relocation of the electrical housing to the bottom airfoil allows for easy access to the housing while the wingsail is mounted to the boat. The housing is attached to a T-shaped wooden plate that has two mounts for a micro-USB port and a rocker switch for power, see subsection 4.1.3. The wooden plate is fixed to a 3D printed mount that clamps around the mast. The housing and plate slide up into the housing for mounting.

The removable wingsail section is built around a smaller carbon fiber mast which slides into the main wingsail mast. A 3D printed insert in the main mast secures the removable mast into the assembly. The removable section is held onto the main wingsail using magnets. Two magnets are mounted to the top airfoil in the main wingsail assembly and also to the bottom airfoil of the removable section.

The assembly process of the wingsail included 3D printing various connectors including: airfoil to mast supports, an electronic housing mount, trim tab to wingsail mounting connectors, and upper & lower trim tab connectors. The wingsail and trim tab airfoils were manufactured by laser cutting wood templates, then using these templates to hot wire cut the shape out of the foam. The leading edges and trailing edges were cut out of foam using the same method as the airfoils. Once all

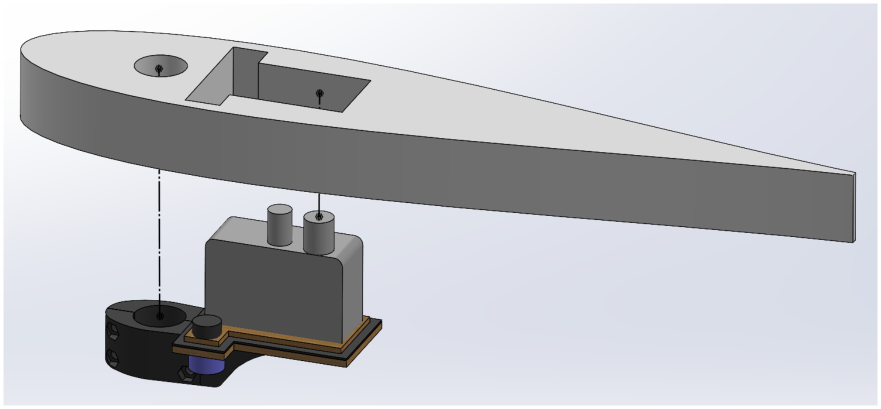


Figure 19: Electronic housing assembly

the components needed for assembly were constructed, each foam piece was sanded and painted with white acrylic paint. For assembly, the components were glued together using Gorilla Glue. Finally, MonoKote, a plastic heat-activated adhesive film, primarily used for model airplanes, was applied to the outside of the assembled wingsail.

4.2 Ballast Assembly

As described in the background, the ballast is integral to SailBot's operations as it acts as a counterweight to the system when the boat heels, influencing the vessel's ability to sail.

The previously implemented ballast system was operational, but had many shortcomings. This included a lack of mechanical control, over-rotation, and a damaged gearbox. We identified the previously implemented gearbox to be the root cause. The reason being that speed was prioritized for the gear ratio design, which caused the ballast to move too fast to control, leading to over rotation. Additionally, the physical properties of the implemented gears were not strong enough to withstand the forces from over rotation. This led to notable damage to the gears. Furthermore, the previous shaft (connecting the initial driving gear to the motor) did not fit the gear that it controlled to the tolerance required, resulting in play in the gearbox. All aspects considered, the reliability of the gearbox decreased over time, which increased the potential for damage. It is for these reasons that we decided to redesign the gearbox. The primary goal for the redesign was to implement a mechanical control system that prioritized torque output. The following subsections outline our redesign process.

4.2.1 Determining the Motor's Characteristics

We found that there were no markings on the motor indicating its make, model, or any other characteristics. As a result of this, it was necessary to determine the ballast's motor characteristics by physically testing it. The purpose of testing the motor was to determine the following:

- Free running speed
- Free running current
- Stall torque
- Stall current

These characteristics were measured with a reference voltage of 14.8 V (the battery powering the motor is rated as such), a multimeter, an adjustable power supply, a spring scale, and a lever arm extension that was custom made for the motor (Figure 20). To measure free running speed, we marked the motor's output with a bright green line and super-imposed it over some paper with a similar green line. Starting with these two lines parallel and overlaid with each other, we ran the motor at 14.8 V with no load and recorded a short video, then counted the number of times the green line on the motor crosses the green line on the paper. To calculate stall torque, we secured the motor to a table with a vise and the output shaft was connected to the lever arm extension. We measured the distance between the motor output shaft and the hole at the end of the extension for calculation purposes. Then, a string was tied between this hole and a spring scale for the stall torque test. With the motor on, the spring scale was held in place and its value observed and recorded. The motor was run at various voltages less than 14.8 V. For each data point we calculated a stall torque, then we extrapolated this data to determine the stall torque equivalent at 14.8V. We used this method of extrapolation because during the test, we realized that the adjustable power supply could not supply sufficient current.

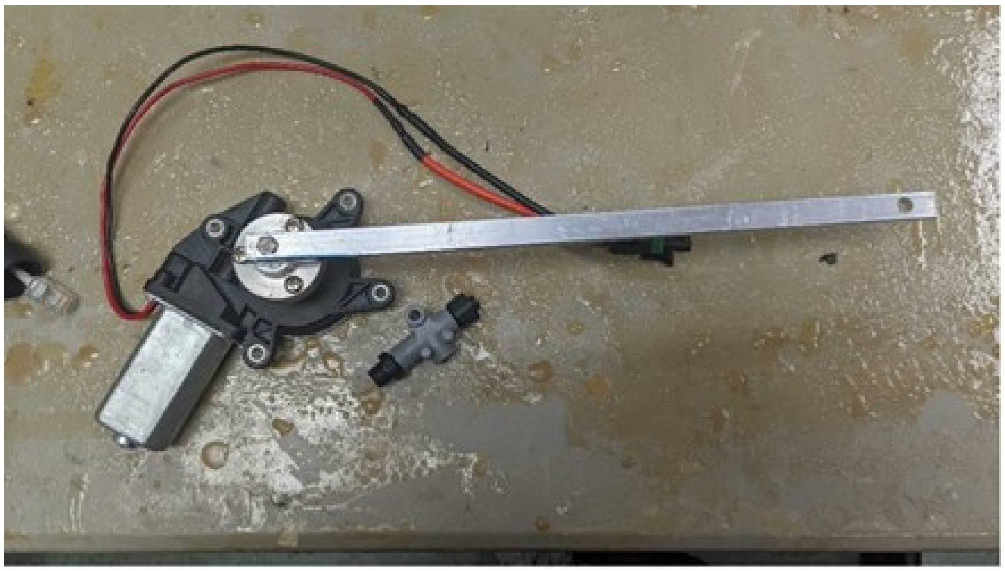


Figure 20: Lever arm used for stall torque and current motor tests

Once complete, the results were entered into a spreadsheet which calculated output torque, current draw, power, efficiency, and heat generated at various speeds between zero and the motor's free running speed. After running tests on the motor, we graphed the motor's characteristics. From Figure 21, we determined that the motor is most efficient when powered at 30 W. At this power, the motor produces the following inputs into the gear train:

- Torque: 3 N-m
- Speed 96 RPM

For more details on the results of this test, see Appendix A.

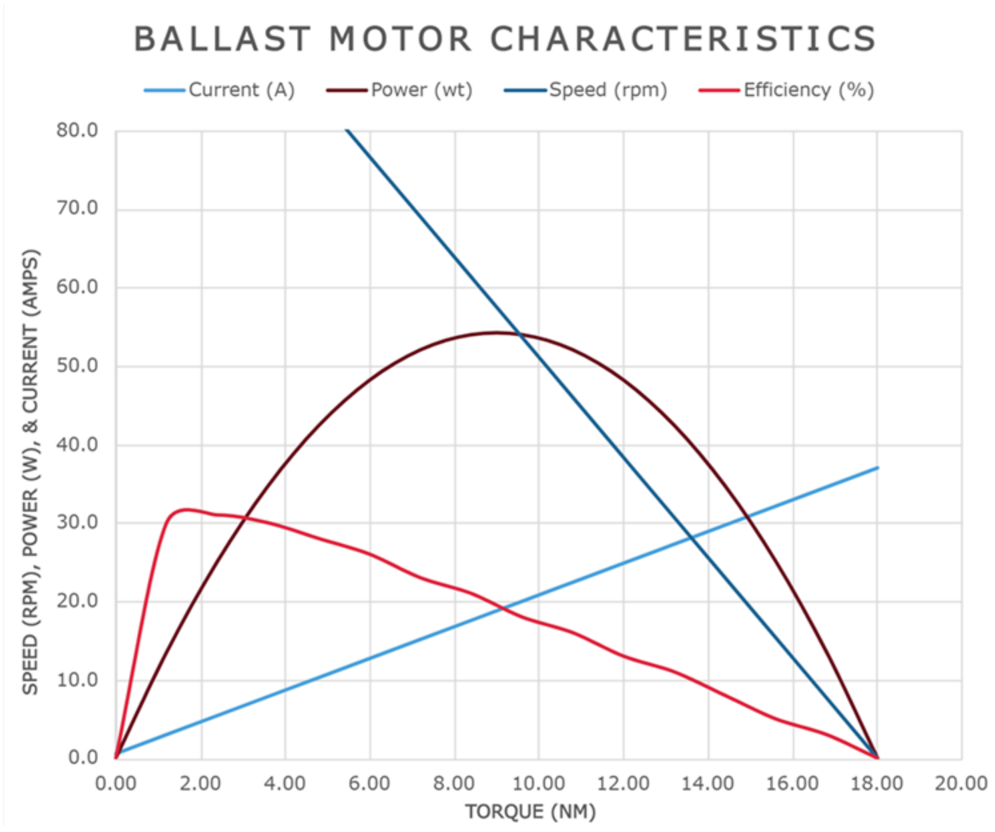


Figure 21: Results from ballast motor characteristics' tests

4.2.2 Determining the Minimum Required Torque Output of the Control System

As previously stated, in the re-design of the control system, our goal was to prioritize torque output. Therefore, it was necessary for us to determine the minimum torque that the control system would need to output. To do so, we needed to calculate the moment exerted by the ballast. This was a simple calculation (See the equation below). See Table 3 for nomenclature.

$$M_{ballast} = W_{ballast} * l_{arm}$$

The only considerable variable was the weight of the ballast since the ballast could be one of three weights. This is because the MQP team that designed the ballast included the option of mounting two additional weights to the ballast. From our advisor's insight, it was determined that

Table 3: Nomenclature for Section 4.3.2

Variable	**Meaning**	**Unit**
$M_{ballast}$	Moment of Ballast	N-m
$W_{ballast}$	Weight of Ballast	kg
l_{arm}	Length of Ballast Arm	m

the weight of the ballast without the two additional mounts was sufficient. This weight is equivalent to 7.36 kg. With the weight and the known ballast arm length of 0.658 m, the minimum torque required was calculated to be 47.5 N-m.

4.2.3 Calculating the Gear Ratio for the Gearbox

Once the motor's characteristics and minimum required output torque were determined and the gearbox efficiency assumed to be 95%, the next step was to calculate a suitable range of gear ratios using the equation below. See Table 4 for nomenclature.

$$r = \frac{\tau_{output}}{\tau_{input} * n}$$

To determine an appropriate range of r values, τ_{output} ranged from the minimum required torque output determined in section 4.2.2 to an output that had a safety factor of 2. In addition to calculating this range of values, we determined the time it would take the ballast to move from one side to the other for each r value. Though this design process prioritizes torque output, the output speed was still an important factor to consider. We did this using the following equations (See Table 4 for nomenclature):

$$\omega_{output} = \frac{\omega_{input}}{r}$$

$$t = \frac{60}{\omega_{output}} * \frac{1}{2}$$

The full results of these calculations can be seen in Appendix B.

From the determined range of appropriate gear ratios, we decided to use a ratio of 25:1. This ratio was chosen because the team determined that it offered a suitable trade-off between out-

Table 4: Nomenclature for Section 4.3.3

Variable	**Meaning**	**Unit**
r	Gear Ratio	-
τ_{output}	Output Torque of Gearbox	N-m
τ_{input}	Input Torque of Gearbox	N-m
n	Efficiency of Gearbox	-
ω_{output}	Output Speed of Gearbox	RPM
ω_{input}	Input Speed of Gearbox	RPM
t	Time to Rotate from One Side to the Next	s

put torque and output speed considering our system (see Table 5). This output torque provides a safety factor of 1.5, which is appropriate considering that the actual moment that the ballast will exert is expected to be greater than the calculated moment. This is due to the highly dynamic system that is endured in normal sailing operations, which increase the minimum torque output required by the gearbox. These dynamic factors include natural occurrences such as the wind and the waves. Additionally, this reduced speed is appropriate since the previous output speed was known to be too quick.

Table 5: Calculated Outputs of 25:1 Gearbox

τ_{output} **[N-m]**	ω_{output} [RPM]
71.25	3.84

4.2.4 Designing and Implementing the Control System

Once the gearbox's gear ratio was determined, the next step was to design the new control system. After researching various gear transmission types, it was decided that a planetary gear transmission would best suit our needs. The reason being that a major constraint in this redesign is the space available for the gearbox in the hull. Planetary gear transmissions are able to output high amounts of torque in a compact space. Ultimately, we custom designed and ordered a VersaPlanetary Gearbox from VexPro. This designed includes the VersaPlanetary Integrated Encoder which is a magnetic encoder that reads position. This decision meant removing the previous two-stage spur gear gearbox.

Once this decision was made, we designed coupling systems between the motor and the gearbox and the gearbox and the ballast. To complete the transmission system, we machined a custom input shaft to connect the motor and the gearbox. Additionally, we purchased a coupler that was properly rated for high torque loads from McMASTER-CARR. This coupler connects the gearbox output shaft with the ballast.

Finally, with a transmission system in place, we designed a mounting system. This mounting system consists of plywood ribs, an angled motor stand, and custom hat channels. These hat channels were manufactured using the WPI RBE Department's Markforged 3D Printer. The reason being that the printer is able to manufacture stronger parts due to its capability to print with more reliable material. This was important for our purposes because the hat channel was designed to apply compression forces to the gearbox to improve stability. This mounting system fastens the gearbox to the Plexiglas plate grounding it. Figure 22 is a model of this design.

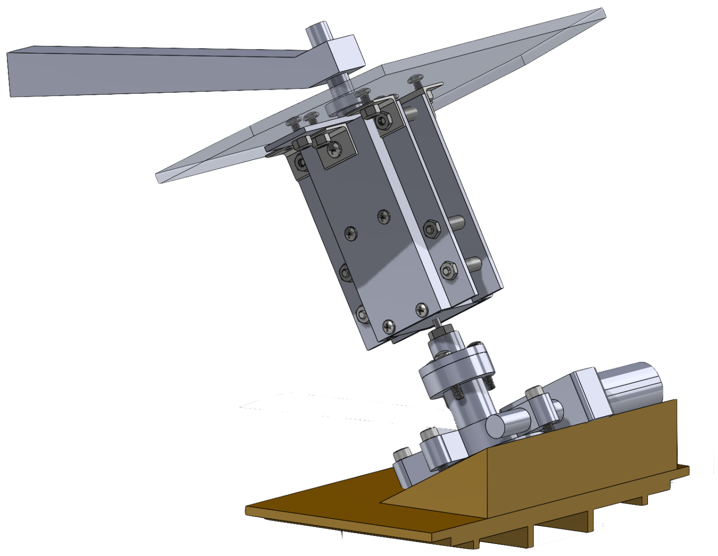


Figure 22: CAD model of mounted gearbox

4.3 Electrical System

To improve the electrical system, we began by providing proper voltages to each of the components, securing components within the hull, and replacing some of the wiring. We also re-implemented the magnetic power switch and added a battery indicator circuit.

4.3.1 Improve Wiring

The wiring inside of the hull has gone largely unchanged for a few years. Continual usage and wear from moving around inside of the vessel has led to the deterioration of some of the insulation. To improve the wiring, we replaced the old wiring with the proper gauge wire and routed the wires such that they wouldn't rub against the hull. Notably within the Jetson housing box, some of the wires coming from the NMEA bus to power the Jetson were exposed and causing shorts. For this the wires were replaced. The 5V regulator had 14 AWG wire spliced with 22 AWG wire going to the Jetson, and was redone because to the 22 AWG wire becoming loose. Wires going through different sections of the hull now go through grommets and watertight connectors. Lastly, the extra wiring from past components was removed and excess wire shortened.

When we started work on the boat, some of the electrical components were operating outside of their required voltage specifications. To provide the proper voltages, we first had to get the datasheets for each of the components we are using. After we got the minimum and maximum voltage requirements, we measured each of the components' power lines to compare the voltages being delivered against these requirements. The components receiving inadequate voltage were then marked and their voltage regulators or sources of voltage checked to see where the problem was. From this, we discovered the problem with the 5V regulator wires mentioned above. Similar analysis was then performed on the components which were not receiving enough voltage – such as the 900 MHz Ethernet bridge.

We have also mounted the electrical housings onto a plywood base plate that utilizes the ribs of the hull to keep everything stationary. The housings are held on with velcro strips making it easy to take the electronics out if needed. A layout plan drawn on the base plate also helps us position the electronics such that the center of weight is oriented along the center line of the boat.

4.3.2 Battery Indicator

The only indication of the charge of the battery on the hull was the LED on the magnetic switch. It showed if the hull was being powered or not and whenever the LED was dimming, the team would know that the battery was getting low on charge. However, the team was unable to really use this as a reliable way to get a sense of what voltage was currently being supplied by the battery. The new battery indicator circuit fixes this and sends battery voltage information back to the land-based team during SailBot operation.

For this we used a custom 4-bit ADC as seen in Figure 23. We are using a four cell LiPo battery which has a voltage ranging from 16.8 V fully charged to 12 V at its recommended safe minimum. The Jetson Nano has a pin voltage threshold of 3.3 V. The Jetson Nano reads P29 first and if it is below the 3.3 V threshold it will read P31, and this is repeated until P37. The Jetson takes these readings and sends them to the telemetry dashboard which displays the battery status as a battery symbol with 3 bars. Three bars means P29 is being read as a binary 1, while two bars means P29 is a binary 0 and P31 is a 1. This is repeated until there are no bars. When the battery falls below 13.3 V P37 will read a binary 0 on the Jetson. This is the team's safe minimum, allowing a 1.3 V buffer within which the operators should return the boat to shore for charging. The physical

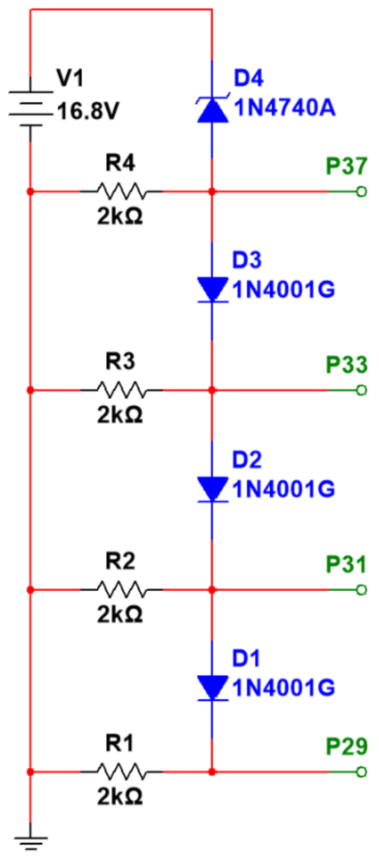


Figure 23: Battery voltage indicator

circuit is located within the main control box to be closer to the Jetson for reduced wiring.

4.3.3 E-paper Screen

Currently on the robot there is an E-paper display that in previous years had been used to display debugging information in a way that can be easily viewed by anyone working on the robot or close to it. However, this display has sat unused for several years, and the code to operate it has not yet been translated into the current ROS2 architecture. Because of the utility of this display as a debugging tool, we planned to re-integrate it into the boat. The screen is a tri-color 2.9-inch Waveshare model B E-Paper screen. Previously it was controlled by a SMT32 type microcontroller which collected data from the CAN bus and printed the desired information to the screen. The CAN bus uses a serial-based protocol commonly used in many automotive and nautical applications. It required data to be parsed into and out of messages according to the NMEA2000 and NMEA 183 formats. Due to the switch to ROS2 and the Jetson Nano as our central processor, many sensors now communicate directly with the Jetson rather than through the CAN bus. We decided to interface the E-paper screen directly with the Jetson in order to drastically decrease the software complexity. To do this we used a Cat5 wire which ran straight to the Jetson. The E-paper screen would also get its own ROS2 node which listens for data on the desired topics and displays it on screen.

4.3.4 Magnetic Switch

The electrical system used to be connected directly to the battery with no on/off switch between. A few years back, a magnetic power switch was added as a way to quickly turn the boat on or off as needed without having to open the hull and connect or disconnect the battery by hand. It had since been disconnected. This year, we reimplemented the switch and updated the connectors to an XT90 connector to make it compatible with our current LiPo batteries. The magnetic switch sits between the battery and main powerline leading to the fuse box. It works by connecting or disconnecting the positive power line with a 12-volt relay. To turn the electronics on, the north pole of a magnet can simply be presented over the switch. To turn the robot off the south pole of the magnet is used. A green LED indicates if power is on or off.

4.4 Communications

This year we looked into extending the range of our ship-to-shore communication systems. When it came to RC control, we noticed that the receiver had previously been located close to or below the water line. This reduces the overall Received Signal Strength (RSS) since 2.4 GHz radio waves have a hard time passing through water. To improve reception, we mounted the antennas as high as we could inside the hull. We also switched to using a FRSky L9R receiver which has a higher receiver sensitivity and is reported to have twice the range when compared to the X6R.

5 Software Development

5.1 Navigation Algorithms

While previous years' teams created the framework for several autonomous navigation algorithms, these systems had not been tested and were largely incomplete. Because of these issues, we decided to build new systems that would be responsible for navigating the boat throughout the competition. Specifically, navigation systems needed to be able to autonomously move the boat through given target locations. To do this, We first decided to design a more robust system for creating maps of sailable areas so that target locations could be generated quickly. Additionally, we decided that planing tacks and jibes, previously done by the A* algorithm, should instead be handled at a lower level so that local changes could be handled without completely recalculating the path between targets. Finally, the rudder control system needed to be overhauled to more reliably keep a desired heading while being jostled by environmental forces.

5.1.1 Target Generation

The role of the target generation algorithm in our navigation stack is fairly straightforward. Our point-to-point navigation algorithm assumes that there are no obstacles between the boat's current location and the target. Therefore, we needed a way to ensure that the targets we pass into it would not cause the boat to travel into unsailable areas. To generate a target location, our robot first needed a grid map of the sailing area. Here we took advantage of the fact that the course was known ahead of the competition and built our map manually. This was done through a program which downloaded a satellite image of the course location from Google Maps and overlaid a grid on top of that image. This way, the operator could manually select areas on the map which they deemed unsafe to sail, and those grid cells would be marked accordingly. After a buffer was applied around the user-selected parts of the grid, the program output a grid map of the area. Using this grid, we ran a basic A* algorithm on manually selected target locations. This generated straight lines between them, adding intermediate points if it was necessary to avoid an obstacle. By including the intermediate points in our list of target locations, we ensured that a navigable path could always be found between two targets, and our full list of targets could be safely passed to the next step in the navigation stack (see Section 5.1.2). While on the water, the target generation function would monitor the robot's position and send a new target whenever the current one was reached.

5.1.2 Point-to-Point Navigation

Once a target location was selected, the robot needed to calculate the best way to reach that target. When the target was upwind or directly downwind, this required planning tacks and jibes so that the boat would avoid the no-go zone. To do this, we implemented a system inspired by Roland Stelzer's short course routing system for autonomous sailboats (*Autonomous Sailboat Navigation*, 2012). Stelzer's system uses a boat's polar speed diagram to continuously calculate the optimal heading to a target while taking wind direction into account. Speed predictions are made for true wind angles between 0° and 180° at several wind speeds. A boat's polar speed diagram, an example of which is shown in Figure 24, shows how quickly a boat can travel in any direction given an observed wind direction and speed.

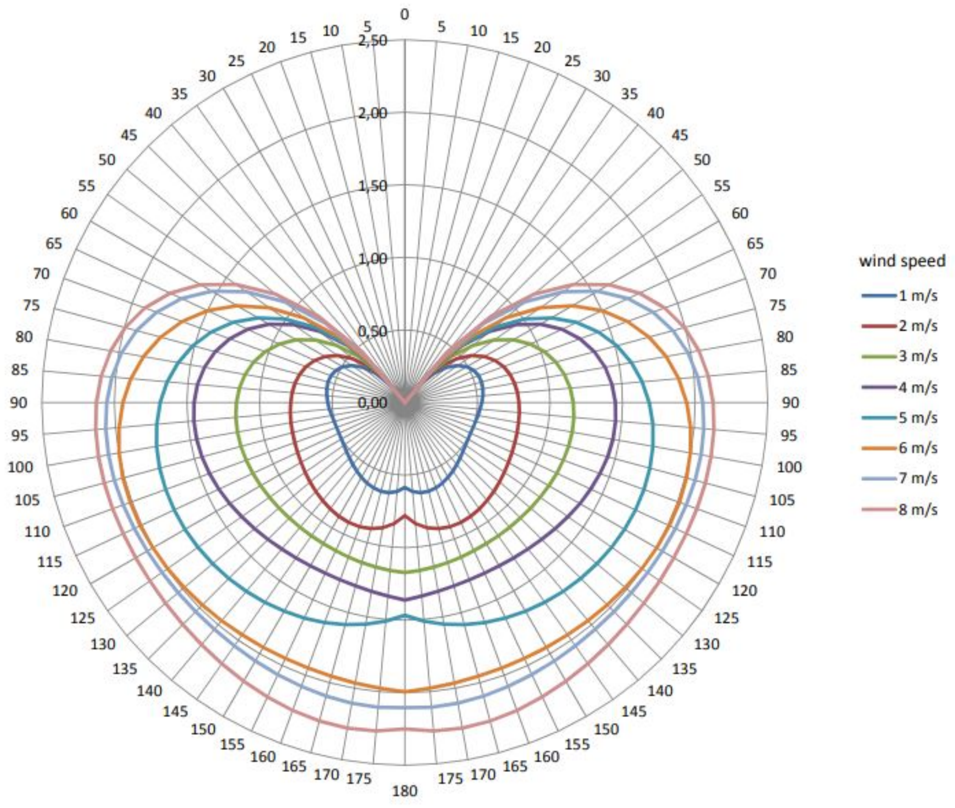


Figure 24: Polar speed diagram for a sailboat

Once the polar diagram is obtained, the efficiency of traveling in a given direction is calculated by projecting the speed vector created by the graph in the direction of the target, as seen in Figure 25. It is important to note that the orientation of the polar diagram is based on the direction of the wind rather than the boat. Finding the optimal heading then requires maximizing the efficiency equation over all possible directions. To reduce the computational load required by this process, our system generates a finite number of potential headings between 0° and 360° and finds which of those headings is most efficient. Furthermore, creating the true polar diagram of a boat requires extensive testing under a variety of wind conditions, which we could not do while the sail was still being built. To get around this, our algorithm was based around a simplified polar diagram, shown in Figure 26, that captured the behavior we wanted our boat to exhibit without the need for precise velocity predictions.

One issue that arises when using a velocity-made-good diagram to navigate is that when the target is directly upwind, the efficiencies of the nearest navigable headings are almost equal. This causes the algorithm to quickly beat back and forth between disparate headings, which is not desirable when the boat is actually on the water. To avoid this, newly calculated headings were compared against the current heading multiplied by a scaling factor called the beating parame-

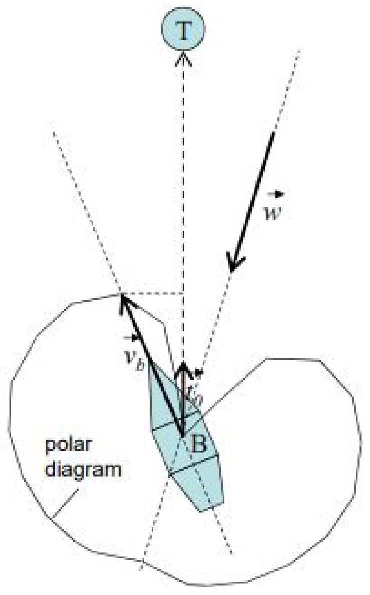


Figure 25: Projecting the boat's speed vector, v_b, in the direction of the target

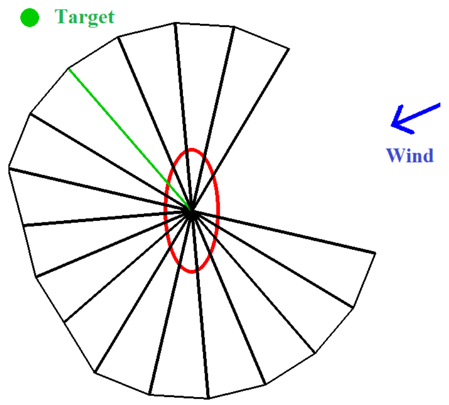


Figure 26: Simplified polar diagram calculating the optimal heading for a simulated target location and wind direction

ter. By tuning this beating parameter we were able to control how often the boat turns, creating a smooth, navigable path.

5.1.3 Low-Level Control

Once an optimal path is generated, the boat needs to be able to maneuver itself along that path by controlling the rudders and trim tab. Control of the trim tab is relatively unchanged from previous years and still moves to a predetermined state, but rudder control has been completely overhauled. Finding an optimal rudder position in an uncertain environment poses its own set of challenges. In our case, fluctuating wind and waves mean that continuously calculating the exact optimal position of the rudders at a given time would be impractical. To solve this problem, we implemented a fuzzy logic control system to continuously calculate the best position for the rudders.

Fuzzy logic is a system of logic rules for defining approximate control states rather than exact numerical values. Fuzzy logic control systems are most useful when it is difficult to precisely map a system's input to its outputs, usually due to stochastic environmental variables. These systems are broken up into three parts. The first, *fuzzification*, converts "crisp", real-world variables - in our case, the difference between the boat's current and desired headings and the rate that the boat is turning - into fuzzy variables, such as "left" or "hard right."

The next part defines the rules that control the system, allowing programmers to easily incorporate real-world expertise into the codebase. For example, our system is built on simple rules such as "If the boat is currently turning right, and the desired heading is to the left, then the rudders should turn hard to the left." These kinds of "if x and y then z" statements make up the knowledge base of the fuzzy logic system and allow for intuitive control.

Finally, the fuzzy logic system converts the fuzzy output variables into crisp real-world variables using *defuzzification*. Using this system, the boat is able to continuously calculate the best rudder position while reacting to shifting environmental conditions. An example of how the fuzzy logic system reacts to different situations is shown in Figure 27. In this example, the first (top) image, the desired heading is 80° to the right and the boat is travelling straight ahead, so the output is a strong turn to the right. In the second (bottom) image, the same desired heading is given, but the boat is measured to be already turning in the direction of the target, so the output is a much smaller move to the right.

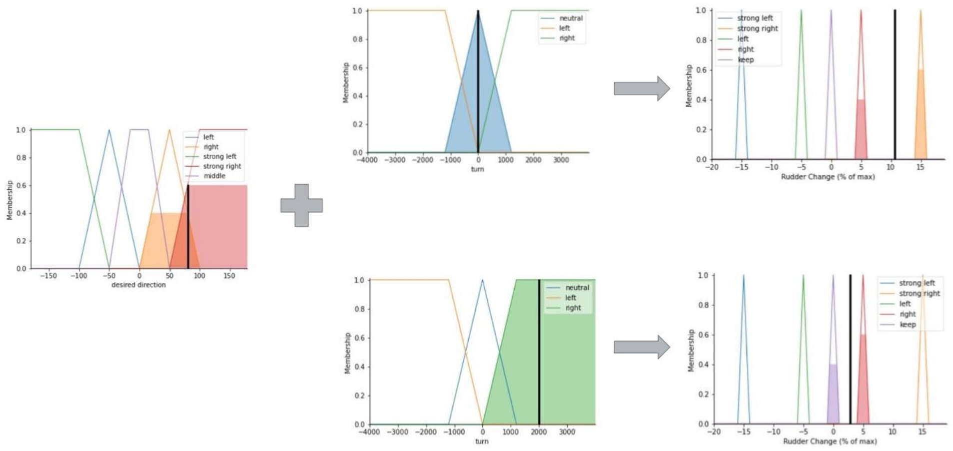


Figure 27: Two examples of the fuzzy logic control system determining how to move the rudders

5.2 Computer Vision

This year's project team implemented a computer vision system for identifying IRSR competition buoys and reporting their location in the water. The system includes:

- A ZED 2 stereo camera mounted to SailBot's bow, which captures a stream of input images;
- A custom object detector model, trained to identify buoys in each image; and
- A detector script, originally provided by Stereolabs, and modified by the project team to publish details on the buoy's location to navigation-planning ROS topics.

To connect the camera to the Jetson, a USB 3.0 cable was run from one of the Jetson's USB ports through the hull to the battery section in the bow of the boat.

5.2.1 Model Selection

Our team chose to use the fourth version of the You Only Look Once algorithm, or YOLO, as the basis for our object detection. YOLO systems implement a convolutional neural network framework known as Darknet which, when configured for a specific application, breaks images into distinct regions and attempts to identify objects of a pre-specified class within each region. It then reshapes the region across multiple iterations to determine a bounding box for each recognized object in the image. We chose the fourth version of YOLO instead of the third because it provides 10% higher average precision (*What's new in YOLOv4?*, 2020), and we opted not to use the fifth version as its dependencies rely on software not compatible with our Jetson Nano.

5.2.2 Building Darknet

The process of developing SailBot's computer vision system began with the use of Google's Colab virtual machine to compile and run Darknet. The Medium article *Train A Custom YOLOv4 Object Detector (Using Google Colab)* (Techzizou, 2021) provided detailed descriptions of how to configure, train, and evaluate the performance of a custom detector. Following the steps laid out in the article, we cloned the Darknet repository to Google Drive, customized a .cfg file for one object identification class ("buoy"), uploaded relevant files to Google Drive and attempted the "make" command to build the Darknet binary. This did not work initially, and through research and debugging we determined we needed symbolic links to certain libraries in order for the Darknet binary to compile properly online. When all the necessary libraries were recognized by the compiler, the Darknet binary was able to compile in under 1 minute. Once the Darknet binary was built, we used it to train the detection model on our training set.

5.2.3 Training & Testing the Model

To detect competition buoys, we required two sets of images to use: the training set and the testing set. The training set consisted of nearly 300 JPG images of our competition buoy, and were captured at variety of different angles, lighting, and physical settings (including indoor shots in the lab and outdoor shots at Lake Quinsigamond). To learn from the buoys in these images, all training images need their buoy(s) labelled with bounding boxes. Thus, after capturing pictures for the training set we utilized the open-source software OpenLabelling to label all pictures in the training set. The testing set consisted of roughly 20 JPG images of orange buoys found on Google Images, which were disjoint from the training set images. This set did not need to be labelled, since they were used for evaluating the model's ability to detect and label a buoy in an unrecognized image.

After compiling the Darknet on the Colab virtual machine, the training set is used to train the YOLO model and output a .weights file. This file is the primary output for computer vision training – in a sense, these weights constitute the model itself. The Medium article showed us how to train the model, where the .weights file is placed in Drive, and how to restart training when the virtual machine disconnects. It also specifies a goal for model training: the average loss should rest between 0.005 and 0.3. After about 6 hours of training, SailBot's YOLOv4 model had been trained to an average loss between 0.3 and 0.5. While this rests slightly above the recommended loss, we found that in practice the detection is consistent enough to use for competition.

The model's first tests were ran in the virtual machine. For these primary model tests, the team used images from the testing set as inputs to the Darknet detector, along with the newly trained weights, the custom .cfg file, the names of our object classes, and a floating-point threshold parameter to filter out lower-confidence predictions. Through these tests, we demonstrated the validity of this model as it detected the buoy or buoys present in most test images with high (90% or higher) confidence. This model, given a single image, will accurately return a 2D bounding box of any buoy present in the image if its confidence exceeds the threshold.

5.2.4 ZED2 Stereo Camera Integration

The ZED2 is a stereo vision camera created by Stereolabs which features a fully integrated IMU (Accelerometer, Gyroscope, Magnetometer) as well as a temperature sensor and barometer. Stereolabs also maintains a Python API which integrates the ZED2's raw sensor data with high level functions and concepts like depth sensing, positional tracking, spatial mapping and object detection. In our case, the ZED SDK Python API was used to connect the ZED2 camera with our ROS2 dashing code base.

Conveniently, Stereolabs maintains a ROS2 wrapper for releases Foxy Fitzroy and Eloquent Elusor, with main support for ROS2 Foxy. This wrapper parses all high level functions, and data types into the appropriate ROS2 nodes and custom messasge types. To utilize this wrapper on the Jetson, our code base needed to be migrated from ROS2 Dashing over to ROS2 Eloquent. This is because the Jetson Nano does not currently have official support for Ubuntu 20.04 which is required for ROS2 Foxy. Leaving only one option, ROS2 Eloquent was installed alongside ROS2 Dashing to test the capabilities and viability of the zed-ros2-wrapper. The zed-ros2-wrapper proved to be a valuable tool to easily visualize and learn about the inner workings of object detection, positional tracking, and spatial mapping, but the wrapper also put an extreme load on the Jetson Nano.

For this reason, the decision was made to keep all computer vision code and the ZED SDK within the object detector script, only integrating the essential data (i.e. detected objects' positions, velocities, and camera pose) into ROS2 Dashing.

5.2.5 Buoy Localization

As stated in Section 3, one of our goals was to autonomously detect and maneuver around buoys and obstacles. To do this, SailBot must be able to identify a detected object with a high accuracy. It also must be able to detect more information about an object than its just position, such the object's movement direction and velocity. We combine our custom YOLOv4 buoy detector with the robust functions from the ZED SDK to achieve this. Localization of the buoy specifically was the focus when working towards this goal, and as a result it is the most developed localization process. This process can ideally be expanded to more moving objects, such as boats, to perform collision avoidance maneuvers. The YOLOv4 detection model returns a 2D box around a detected buoy and the confidence of that detection. The YOLOv4 uses a single image, while the ZED SDK uses all of the ZED2's sensors and operates specifically in the 3D domain. The ZED2 camera also automatically fuses its IMU data with its visual odometry to keep track of its starting position. This allows us to define the location of any detected object in the world as the camera moves. The purpose of this function is to combine the available information and precisely define the position of a detected buoy in the world.

The three steps used to define the position of a buoy are:

1. Retrieve buoy 2D bounding box from YOLOv4
2. Supply the ZED SDK with said 2D bounding box to locate the buoys
3. Retrieve new 3D objects from the ZED SDK containing the position with respect to a zeroed world frame

As stated above, the YOLOv4 detection model only uses one frame of a video feed to detect an object. Four numbers are returned from the YOLOv4 detector that describe the 2D bounding box, and its confidence value. The four numbers of the 2D bounding box signify the following parameters in pixels:

1. Box center x-coordinate
2. Box center y-coordinate
3. Box width
4. Box height

Each detected buoy in a frame is represented with the above parameters and a confidence value. More information is needed if the buoys position in the 3D world is desired, which is the purpose of the ZED SDK.

The ZED SDK is passed the current 2D bounding box of each object we want to track. With the 2D location of each object, the ZED SDK can use the ZED2's sensor data to track and define the object in the 3D world. This process is called "ingesting". The ZED SDK automatically identifies and tracks any object with its depth and IMU data.

To ingest the YOLOv4 detections into the ZED SDK, the "ingest_custom_box_objects()" function is used. Supplied with an "sl.CustomBoxObjectData" object, the ZED SDK disguises, identifies and tracks any object throughout a session. sl.CustomBoxObjectData object supplied requires these attributes:

1. unique-object-id (randomly generated)
2. label ("buoy")
3. probability (confidence)
4. bounding-box-2d (in a,b,c,d form)
5. is-grounded (set true if buoy, set false if trying boat detection)

The last step needed to retrieve the 3D position, velocity and other information is to simply call "retrieve_objects()". It was assumed that a buoy‘s position with respect to the world is grounded, as signified by the "is-grounded" attribute. This means that buoy’s velocity in fixed 3D world space will always be zero. Not enough testing was done to conclude if currently assuming grounded buoys can account for drift or not, and more testing will have to be done in the future to determine the proper method for accurate buoy position. This "is-grounded" attribute would also have to be changed if a moving object was being tracked, like a boat for collision avoidance. Do also note, the ZED SDK automatically estimates objects paths, so two objects which cross each other’s path will still be identified as the same 2 objects after the intersection.

6 Testing and Validation

6.1 Electronics

6.1.1 Hull

The Battery Voltage Monitor in Figure 23 is able to read the state of the battery and send it to the Jetson Nano. These values are sent from the Jetson Nano to the telemetry dashboard and updated every three minutes. The monitor has been used in three water tests. The battery status was accessible through ROS messages in all three tests and the telemetry dashboard during the two most recent tests.

6.1.2 Communications

We were unable to determine the full range of our communication systems. A variety of factors ranging from high winds to crew races taking place on Lake Quinsigamond prevented us from collecting data at ranges beyond 1,000 ft. Fortunately, at this range we were able to validate that we could successfully connect to the boat over both RC and the 900 MHz radio link. We did however encounter a higher error rate at this range when using the 900 MHz link. At about 1,000 ft, we noticed that the link had a tendency to start failing whenever line-of-sight (LOS) was disrupted. After conducting a wireless propagation analysis (Appendix C) on Lake Quinsigamond, we found that the expected coverage of our current telemetry radio link (with 95% certainty in LOS conditions) is only about 677 m (or about 0.37 nautical miles). The RC link on the other hand did not lose connection during our tests. Although further testing will be required to confirm the full range of the RC link.

6.1.3 Wingsail Control System

The new wingsail control system is a significant improvement over the previous system. The introduction of Bluetooth Low Energy (BLE) reduced the overall power consumption and proved to be more reliable and easier to use. Compared to when it was using WiFi, the controller now draws less current. While using BLE, we found that the overall current draw only increased by about 20 - 40 mA when turned on. When using WiFi, the current draw would increase by a little more than twice that. We also found it easier to debug the BLE connection. There are a variety of freely available tools, such as nRF Connect, that make it easy to connect to and interact directly with the controller.

The new control system also has yet to encounter any brownouts. After discovering an internal wiring issue in the old trim tab servo, we replaced it with another one of the same type. Since then, the system has been running smoothly without regular restarts.

Furthermore, the new modular design works well. The controller box is now much easier to access and can be fully disconnected from the wingsail if needed. Additionally, at the point where the trim tab booms meet the main sail, the audio jack connector has made setup extremely straightforward. Everything slides together and if components ever need to be replaced, they now can be without the need to tear the wingsail apart.

As for the new fiber optic LED state display, we found that it is not quite as bright as we would have liked when viewed outdoors. It is visible indoors, but when we took it outside we found that it was not always bright enough to clearly see it from a distance. While the LEDs aren't currently at full brightness, the fact that the wingsail is painted white is likely also a contributing factor.

6.2 Navigation Algorithms

6.2.1 Target Generation Algorithm

In testing, the target generation program was able to reliably produce a series of targets that could be safely sent to the point-to-point algorithm. One issue we ran into was the system had no way to vary the buffer it applied to unsailable location, resulting in easily maneuverable obstacles, such as a piece of the dock, having the same buffer applied to it as the shoreline. This results in a few areas appearing more difficult to sail than they actually are, but there were few enough obstacles in the middle of the lake that this was not a major issue.

6.2.2 Point-to-Point Algorithm

Due to acclimate weather conditions, we were unable to test the point-to-point navigation algorithm on the water and needed to rely on simulated path outputs to observe its behavior. Here we found that the beating parameter was extremely helpful when creating a path with as few turns as possible. Figure 28 shows an example of a path created to an upwind target. In this example we used a beating parameter of one, meaning that any new heading had to be at least as good as the current heading before the boat would change direction. In testing, we found that higher beating parameters decreased the number of turns the boat took but ran the risk of overshooting the target. In practice, however, the boat does not need to precisely reach the location of a target, and it was unclear if these simulated errors would translate into significant error when on the water. Furthermore, in testing we found that decreasing the number of turns resulted in greatly increasing the width of the path traveled by the boat. This could cause issues in competition if the course passes near the shore or other unsailable areas. Some solutions to this problem are proposed in Section 7.4.

6.2.3 Low-Level Control

In testing, this part of the navigation stack performed remarkably well. We were able to perform several tests on the system and found that the fuzzy logic controller was able to align the boat

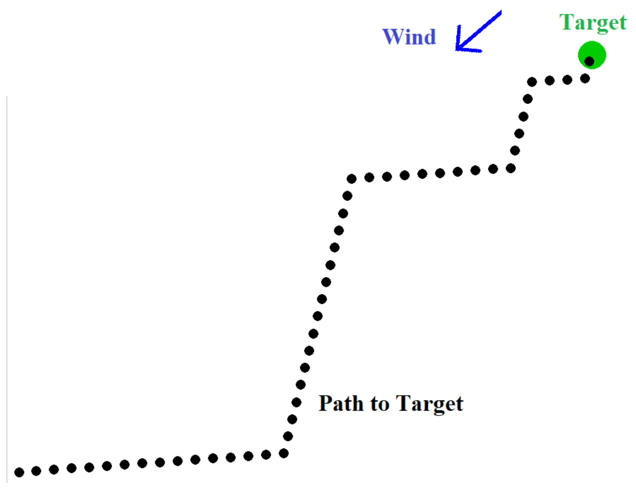


Figure 28: A simulated path to an upwind target using a beating parameter of one

with a given heading much faster than the previous rudder control algorithm. Furthermore, the system was extremely resilient. In water tests, the boat was able to maintain its heading while being jostled by large waves, and when the boat was purposefully pushed off course by a kayak it would quickly right itself.

6.3 CV

The team was able to implement computer vision using the ZED 2 camera. After being trained with a large selection of buoy pictures, the current computer vision system maintains a 98% average confidence while recognizing a buoy within 2 meters of the boat. What's more, the computer vision model was also able to avoid detecting duplicate buoys by tweaking the threshold for the detector.

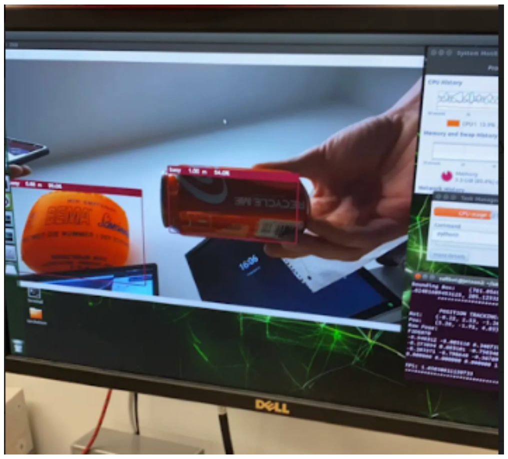


Figure 29: A false positive in the buoy detection

Further development of the computer vision system was limited due to the time constraint and the limited training data set. At its current stage, there are still some false positives when detecting buoys on objects with similar outlines and colors. For example, a Coca-Cola can (Figure 29) and the project team's kayak are two examples of objects that get detected despite not being competition buoys.

7 Future Work and Recommendations

7.1 CV

One of the areas which needs the most work going forward is the computer vision system. While the computer vision system was trained and is usable at its current state, the team has yet able to integrate its implementation with the navigation pipeline. Such integration would help the system to navigate around buoys much more precisely if successfully embedded instead of purely relying on the buoys' hard-coded GPS coordinates, which could fluctuate significantly in competition. What's more, such implementation will most definitely benefit from more precise and accurate recognition from the YOLOv4 model. Regarding the problems observed in testing, both increasing the size of the training set and making fine changes to the threshold would be favorable. The training set could also include live video feeds of buoys on the water - OpenLabelling allows the labelling of individual video frames - to train even better models by providing a higher volume of valid training data.

7.2 Debugging Tools

There are two main debugging tools we would like to see improved upon in future iterations of Sailbot. First is the telemetry dashboard and second is the E-paper screen. More information about the vessels systems can be put onto the telemetry dashboard such as the ballast rotation and trim tab states. A limiting factor to this is the 900MHz bridge being our bottleneck for data transfer. Upgrading the 900MHz bridge to a 2.4GHz bridge would drastically increase the amount of information we could send but the downside would be a decreased range. The E-paper screen is a quick way to check the status of the vessel without having a connection to the Jetson or when that connection is being set up. For this it can be helpful while the land team is connecting to the Jetson and resolving any issues faster.

7.3 Ballast Algorithm

Currently we are using the previous iteration of the ballast algorithm for the control of the ballast motor. This algorithm manipulates the power sent to motor to control the motor's output torque and speed to the gearbox. However, as mentioned in 4.2.1, the ballast motor is most efficient when run at 30 W, and the implemented gearbox was designed under the assumption that the motor would be run at its peak efficiency. Changing this power input would invalidate all ballast gearbox outputs that were calculated during this project. Creating a new algorithm for the ballast that coincides with our controls would be ideal moving forward. As of this year, the planetary gearbox has an encoder that can send data about the ballast's position to the Jetson. In the future, this encoder can be used for developing a system to prevent over-rotation. Utilizing the encoder with our current controls would be beneficial to the overall handling of Sailbot.

7.4 Navigation Algorithm

The navigation algorithms can be improved in several ways. Firstly, as previously stated, the target generation algorithm is very simplistic. It accomplished its goal of making sure the boat does not travel through obstacles, but lacks the ability to generate targets based on environmental factors like wind or water depth. The program is also configured to generate a map of Lake Quinsigamond specifically, and should be configured to allow for new map locations to be imported easily.

As stated in the Section 6.2.2, the point-to-point navigation algorithm needs extensive water testing. Currently, a beating parameter of 1.0 works best in simulations, but work needs to be done to determine the optimal parameter on the water. Furthermore, more work needs to be done to control the width of the path created by the algorithm. Currently the path width is undefined and varies based to the distance to the target. To solve this, Stelzer makes the beating parameter a function of target distance, increasing the value as the boat nears the target (*Autonomous Sailboat Navigation*, 2012). This ensures that boat turns at consistent intervals, but comes with the trade-off of turning more often than with a constant beating parameter. Further testing is required to determine which system performs better in a competition environment.

7.5 Testing the Ballast Assembly

The redesigned ballast assembly was completed at the end of the project's timeline. This means that it was never fully tested. Though we are confident in the new gearbox, it will be important to test how it performs. This includes an endurance test, a RC test (long and short range), and testing the ballast algorithm. The latter two tests will need to be done both on land and in water, as the ballast will operate differently in real conditions. Additionally, it will be important to analyze the mechanical control system after it endures a full water test.

7.6 Testing the Wingsail Assembly

The redesigned wingsail assembly was completed at the end of the project, and therefore was not thoroughly tested. The new wingsail was tested once during a water test without the removable top section. The water test confirmed that the wingsail can successfully operate and withstand winds of 13 mph with gusts of up to 18 mph. However, a single water test cannot completely validate the success of the system, and thus more water testing should be conducted.

7.7 Hull Electronics

As of the writing of this report the E-paper screen is not fully installed. The cables and mounting hardware have been manufactured and tested, but switching the software over from supporting the CAN bus to ROS2 came with some difficulties. Implementation of the e-paper screen has been left for next year's team.

8 Conclusion

As a dynamic system, SailBot has been continuously changed and improved throughout the past five years of its development. This year we implemented a variety of changes aimed at improving the durability and reliability of the system as a whole.

8.1 Goal results

8.1.1 Goal 1: The vessel can maintain control of its basic sailing functions for at least 7 hours

In water tests, we were able to control all the robot's basic sailing functions remotely using RC control. Due to scheduling conflicts and inclement weather, we were not able to perform a full endurance test on the water with all sailing systems running. However, with the exception of the new ballast, we have completed endurance tests for all of the individual systems independently under lab conditions. Furthermore, calculations done using the new electronics system show that the battery should be able to power the boat for the full duration of the endurance test.

8.1.2 Goal 2: A human operator can assume control of the vessel within 1 nautical mile at any time during the vessel's operations.

This goal was formed based on the longest distance we would be expected to communicate over during the IRSR with a safety factor of 2 or more. Unfortunately, we were unable to validate the full range of our communications this year. However, as the full course the boat will traverse in competition is a rectangle with a perimeter of one nautical mile, the maximum distance the boat will travel from its human operators is much closer to 0.5 nautical miles, so we should be able to maintain an RC connection. Our calculations suggest that we may start to lose our telemetry link at this distance, however as long as we can maintain control of the vessel, this is fine for our purposes. Furthermore, in testing the boat was able to switch between autonomous and RC control without any issues.

8.1.3 Goal 3: The vessel can plan a path and maintain course in winds below 15 knots.

In testing, the boat was able to comfortably sail in winds gusting up to 16 mph. However, as previously stated the point-to-point navigation algorithm performed well in simulations but was never able to be tested on the water. The low-level control algorithms, on the other hand, performed very well in water tests and the boat was able to maintain its heading in high winds and could quickly right itself after being pushed off-course.

8.1.4 Goal 4: The vessel can autonomously detect and maneuver around buoys and obstacles.

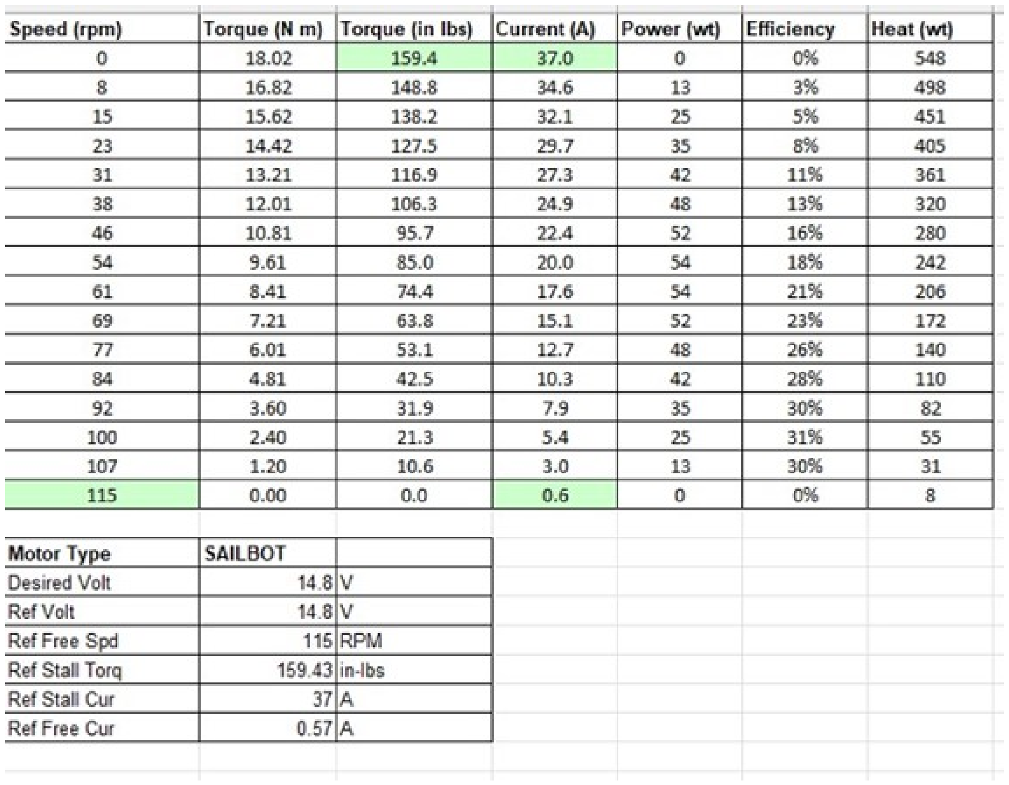
Both in the lab and on the water the computer vision system was able to reliably detect buoys and calculate their position relative to the camera. However, due to time constraints only very basic logic was implemented to maneuver around detected buoys. By sending a new desired heading to the low-level control system, the boat was able to swerve out of the way of a detected buoy; this was a temporary solution and did not change any of the higher-level navigation logic.

References

- *Angle of Attack*. AviationChief.com. (n.d.). Retrieved October 7, 2021, from http://www.aviationchief.com/angle-of-attack.html.
- Burklund, J., Johnson, H., Norris, T., & Shanahan, L. (2018). *Sailbot 2017-2018* [Undergraduate Major Qualifying Project, WPI]. Digital WPI. https://digital.wpi.edu/show/4m90dw93r.
- Burri, C., Eusman, N., Jackson, N., Laks, M., Scholler, C., Thammana, A., & Zebrowski, L. (2021). *SailBot: Autonomous Sailing Robot* [Undergraduate Major Qualifying Project, WPI]. Digital WPI. https://digital.wpi.edu/show/3x816q53b.
- Digi International. (2016). XPress™ Wireless Ethernet Bridge: User Guide. Hopkins, MN: Author.
- Guzman, M., Lavryonova, I., Reese, K, & Thomas, A. (2020). *Sailbot 2019-2020 MQP* [Undergraduate Major Qualifying Project, WPI]. Digital WPI. https://digital.wpi.edu/pdfviewer/x059c990m.
- International Robotics Sailing Regatta. (n.d.). *SailBot International Robotics Sailing Regatta*. SailBot. Retrieved September 20, 2021, from https://www.sailbot.org/.
- Orac, R. “What’s New in YOLOv4?” Medium, Towards Data Science, 24 Aug. 2021, https://towardsdatascience.com/whats-new-in-yolov4-323364bb3ad3.
- Regan, K., Schifilliti, D., & Singer, D. (2017). *The Robotic Automated Wingsail* [Undergraduate major qualifying project, WPI]. Digital WPI. https://digital.wpi.edu/show/sb3979488.
- Stafford, K. (2021). (tech.). ’21-’22 sailbot trimtab operation with sketch.
- Stelzer, R. (2012). *Autonomous sailboat navigation.*, from https://dora.dmu.ac.uk/handle/2086/7364.
- Techzizou. *“Train a Custom Yolov4 Object Detector (Using Google Colab).”* Medium.com, 12 Mar. 2022, https://medium.com/p/61a659d4868#e5b4.
- *Understanding Boating “Right of Way” Rules*. (2021). Discover Boating. Retrieved October 7, 2021, from https://www.discoverboating.com/resources/boating-right-of-way.
- Silva, M., Malheiro, B., Guedes, P., & Gerreira, P. (2019). Airfoil Selection and Wingsail Design for an Autonomous Sailboat. *ISEP/PPorto School of Engineering* from https://recipp.ipp.pt/bitstream/10400.22/15346/1/CAPL_LSA_MBM_2019.pdf
- Doane, C. (2019). *Crunching Numbers: Sail-Area/Displacement Ration*. Wave Train. Retrieved April 28, 2022, from https://wavetrain.net/2011/03/17/crunching-numbers-sail-areadisplacement-ratio/

A Appendix A:Ballast Motor Characteristics

See below for the full data set when calculating the characteristics of the motor.

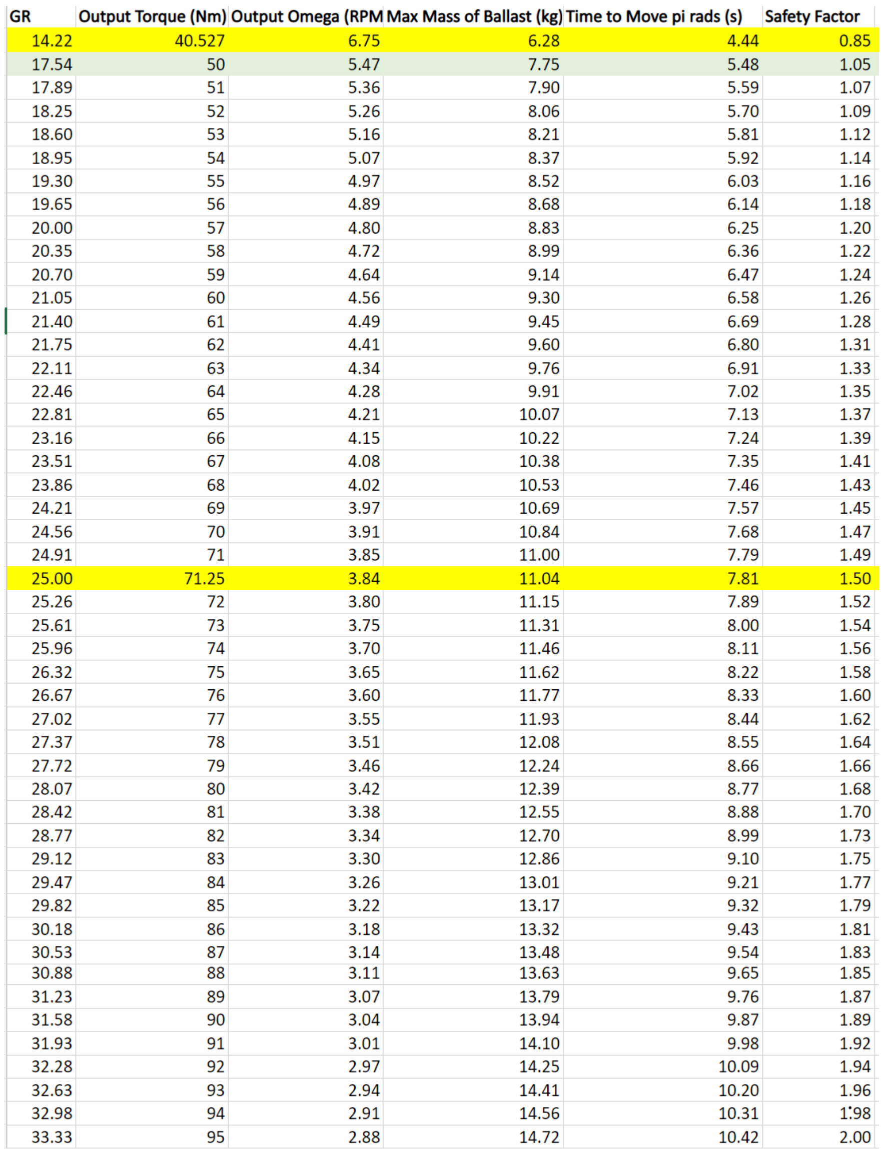


Speed (rpm)	Torque (N m)	Torque (in lbs)	Current (A)	Power (wt)	Efficiency	Heat (wt)
0	18.02	159.4	37.0	0	0%	548
8	16.82	148.8	34.6	13	3%	498
15	15.62	138.2	32.1	25	5%	451
23	14.42	127.5	29.7	35	8%	405
31	13.21	116.9	27.3	42	11%	361
38	12.01	106.3	24.9	48	13%	320
46	10.81	95.7	22.4	52	16%	280
54	9.61	85.0	20.0	54	18%	242
61	8.41	74.4	17.6	54	21%	206
69	7.21	63.8	15.1	52	23%	172
77	6.01	53.1	12.7	48	26%	140
84	4.81	42.5	10.3	42	28%	110
92	3.60	31.9	7.9	35	30%	82
100	2.40	21.3	5.4	25	31%	55
107	1.20	10.6	3.0	13	30%	31
115	0.00	0.0	0.6	0	0%	8

Motor Type	SAILBOT	
Desired Volt	14.8	V
Ref Volt	14.8	V
Ref Free Spd	115	RPM
Ref Stall Torq	159.43	in-lbs
Ref Stall Cur	37	A
Ref Free Cur	0.57	A

B Appendix B: Range of Potential Gear Ratios

See below for the range of potential gear ratios. Note that what is highlighted in green is the approximate minimum values. What is highlighted in yellow is either the old ratio (14.22:1) or the new ratio (25:1).



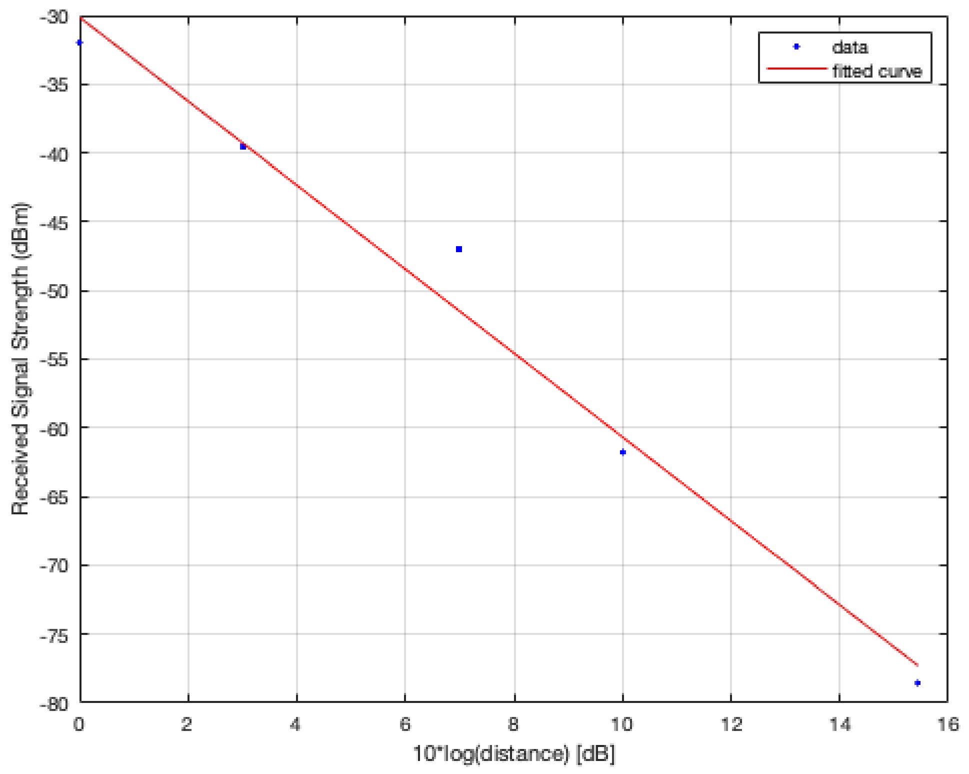
GR	Output Torque (Nm)	Output Omega (RPM	Max Mass of Ballast (kg)	Time to Move pi rads (s)	Safety Factor
14.22	40.527	6.75	6.28	4.44	0.85
17.54	50	5.47	7.75	5.48	1.05
17.89	51	5.36	7.90	5.59	1.07
18.25	52	5.26	8.06	5.70	1.09
18.60	53	5.16	8.21	5.81	1.12
18.95	54	5.07	8.37	5.92	1.14
19.30	55	4.97	8.52	6.03	1.16
19.65	56	4.89	8.68	6.14	1.18
20.00	57	4.80	8.83	6.25	1.20
20.35	58	4.72	8.99	6.36	1.22
20.70	59	4.64	9.14	6.47	1.24
21.05	60	4.56	9.30	6.58	1.26
21.40	61	4.49	9.45	6.69	1.28
21.75	62	4.41	9.60	6.80	1.31
22.11	63	4.34	9.76	6.91	1.33
22.46	64	4.28	9.91	7.02	1.35
22.81	65	4.21	10.07	7.13	1.37
23.16	66	4.15	10.22	7.24	1.39
23.51	67	4.08	10.38	7.35	1.41
23.86	68	4.02	10.53	7.46	1.43
24.21	69	3.97	10.69	7.57	1.45
24.56	70	3.91	10.84	7.68	1.47
24.91	71	3.85	11.00	7.79	1.49
25.00	71.25	3.84	11.04	7.81	1.50
25.26	72	3.80	11.15	7.89	1.52
25.61	73	3.75	11.31	8.00	1.54
25.96	74	3.70	11.46	8.11	1.56
26.32	75	3.65	11.62	8.22	1.58
26.67	76	3.60	11.77	8.33	1.60
27.02	77	3.55	11.93	8.44	1.62
27.37	78	3.51	12.08	8.55	1.64
27.72	79	3.46	12.24	8.66	1.66
28.07	80	3.42	12.39	8.77	1.68
28.42	81	3.38	12.55	8.88	1.70
28.77	82	3.34	12.70	8.99	1.73
29.12	83	3.30	12.86	9.10	1.75
29.47	84	3.26	13.01	9.21	1.77
29.82	85	3.22	13.17	9.32	1.79
30.18	86	3.18	13.32	9.43	1.81
30.53	87	3.14	13.48	9.54	1.83
30.88	88	3.11	13.63	9.65	1.85
31.23	89	3.07	13.79	9.76	1.87
31.58	90	3.04	13.94	9.87	1.89
31.93	91	3.01	14.10	9.98	1.92
32.28	92	2.97	14.25	10.09	1.94
32.63	93	2.94	14.41	10.20	1.96
32.98	94	2.91	14.56	10.31	1.98
33.33	95	2.88	14.72	10.42	2.00

C Appendix C: SailGoat Wireless Coverage Analysis

See next page

SailGoat Wireless Coverage Analysis - Telemetry (LOS)

```
clc;
clear all;
close all;

% Distances in meters
d = [1;2;5;10;35];

% RSS in dB
Pr = [-32;-39.55;-47;-61.77;-78.55];

% Lp vs. distance (in dB)
d_db = 10*log10(d);

% Linear fitting
F1 = fit(d_db, Pr, 'poly1');

plot(F1, d_db, Pr);
grid on
xlabel('10*log(distance) [dB]');
ylabel('Received Signal Strength (dBm)');
```



```
disp(F1);
```

```
     Linear model Poly1:
     F1(x) = p1*x + p2
     Coefficients (with 95% confidence bounds):
       p1 =      -3.054  (-3.838, -2.271)
       p2 =      -30.13  (-37.1, -23.15)
```

Path Loss and the Environment

To determine the ideal expected path loss at 1 meter (assuming we are using the highest frequency in the selected ISM band), we can use the following equations:

Path gain: $\frac{P_r}{P_t} = \left(\frac{\lambda}{4\pi d}\right)^2$

Path loss at first meter: $L_0 = \frac{P_t}{P_r}$

```
f_c = 928e6;
wl = 3e8 / f_c;
l0 = (wl / (4*pi))^2;
l0_db = 10*log10(1/l0)
```

```
l0_db = 31.7927
```

Standard deviation of shadow fading is:

```
disp(std(Pr+l0_db+2*d_db));
```

```
    6.8496
```

$P_0 = -31.79dBm$

$\alpha = -3.054$

$\sigma = 6.8486$

For $\gamma = 95\%$ certainty of coverage, fade margin (F_σ) is determined using:

$$1 - \gamma = 0.5erfc\left(\frac{F_\sigma}{\sigma\sqrt{2}}\right)$$

$$F_\sigma = 6.8486\sqrt{2}\,erfcinv(0.1)$$

```
fm = std(Pr+l0_db+2*d_db) * sqrt(2) * erfcinv(0.1)
```

```
fm = 11.2666
```

Coverage:

```
p_t = 21;  % Transmit power in dBm
p_r = -97;  % Receiver sensitivity in dBm
lp_max = p_t - p_r;  % Max allowable path loss
g_a = 11.5;  % Gain of the yagi (9 dB) plus the receiver
r = 10^((lp_max - l0_db - fm + g_a) / 30.54)
```

```
r = 676.7209
```

Estimated coverage with at least 95% certainty, is roughly 677 m (or about 2,220 ft).