Worcester Polytechnic Institute

A Major Qualifying Project

Sailbot

Autonomous Robotic Sailboat

Submitted By:
Nicholas Gigliotti, Robotics Engineering
James Kuszmaul, Robotics Engineering and Computer Science
Tucker Martin, Robotics Engineering and Computer Science
Ryan Wall, Robotics Engineering

Advised By:
William Michalson, Professor
Robotics Engineering, Electrical and Computer Engineering
Kenneth Stafford, Teaching Professor
Robotics Engineering

Abstract

This project created a robotic sailboat to compete in the International Robotic Sailing Competition (IRSC). The sailboat comprises a variety of custom hardware, electrical, and software components. The 2 meter long fiberglass hull was constructed by the MQP team, along with support from the WPI Robotics Club. It houses a servo actuated rudder, custom winch system for managing the main sheet, a movable ballast system for balancing the boat, structures for attaching the keel and mast, and all the electronics. The boat uses the NMEA 2000 standard, utilizing a CAN bus for effective communication between custom motor controllers, sensors, debugging equipment, and the main processor. In addition to CAN, the boat has a long range WiFi network that allows for remote programming and control and serves as the core communications method with a rigid wing sail developed by another MQP team. At the heart of the boat is a Beaglebone-Black ARM processor running custom software inspired by ROS and written in C++. It manages all the CAN messages, instructs the motor drivers, interprets the sensors, runs control algorithms, and plans high level maneuvers. All this work culminates in a unified design that can perform the IRSC challenges completely autonomously.

Acknowledgements

We would like to thank our advisors, Professors Ken Stafford and William Michalson, for their constant support throughout the project; Jordan Burklund and Hans Johnson, acting as part of the WPI Robotics Club, who helped a great deal with the build of the hull and with the testing of the boat; Paul Miller and the US Naval Academy Sailbot team for their advice and physical donations of a hull form, keel, rudder, and various sails; and the various denizens of the Donahue Rowing Center on Lake Quinsigamond for accommodating us on those weekday mornings when we went out to test the boat.

Contents

List of Figures

List of Tables

1. Introduction

We built and programmed a robot to compete in the International Robotic Sailing Competition (IRSC, aka "Sailbot"). For this project, we produced a fully autonomous sailboat that can complete most of the challenges specified in the 2017 IRSC rules (see section 2.3). The boat is able to autonomously complete all of the challenges which correspond to sailing between waypoints (essentially, every challenge except for the collision avoidance and obstacle search, for which we provide the hardware capability, but which has not been fully implemented).

The challenges are:

- Fleet Race
- Station Keeping
- Navigation
- Carrying a Payload
- Collision Avoidance
- Obstacle Search
- Endurance

The boat can complete these challenges under reasonable wind and weather conditions (the IRSC rules specify conditions under which the competition will not be held; these define the boundaries for what is considered reasonable).

Furthermore, there was a second MQP that built a rigid-wing, actuated sail [23]. In order to provide redundancy for cases in which the rigid-wing sail may not function properly, we can also use a simpler windsurfer sail that is controlled by a winch.

The canonical sailbot consists of:

- A single hull
- A main sail with a single winch-controlled mainsheet
- A single, servo-actuated rudder
- A deep, unactuated, keel with relatively large ballast
- A wind sensor, compass, 6 axis IMU, and GPS
- Minimalistic control and tacking code based on sensor inputs

Beyond this, we also built

- A prototype system for movable ballast for dynamically balancing the boat
- A simulator for the boat systems, to be able to test code on land
- An HTML-based User Interface to allow easy control and debugging of the boat
- Custom electronics boards to improve modularity and maintainability in the boat electrical system
- A mechanical interface that allows the easy swapping out of the rigid-wing and traditional sails

2. Background

This chapter includes all of the background information required to understand the competition, terminology, previous efforts, and competitors' designs.

2.1 Sailboat Terminology

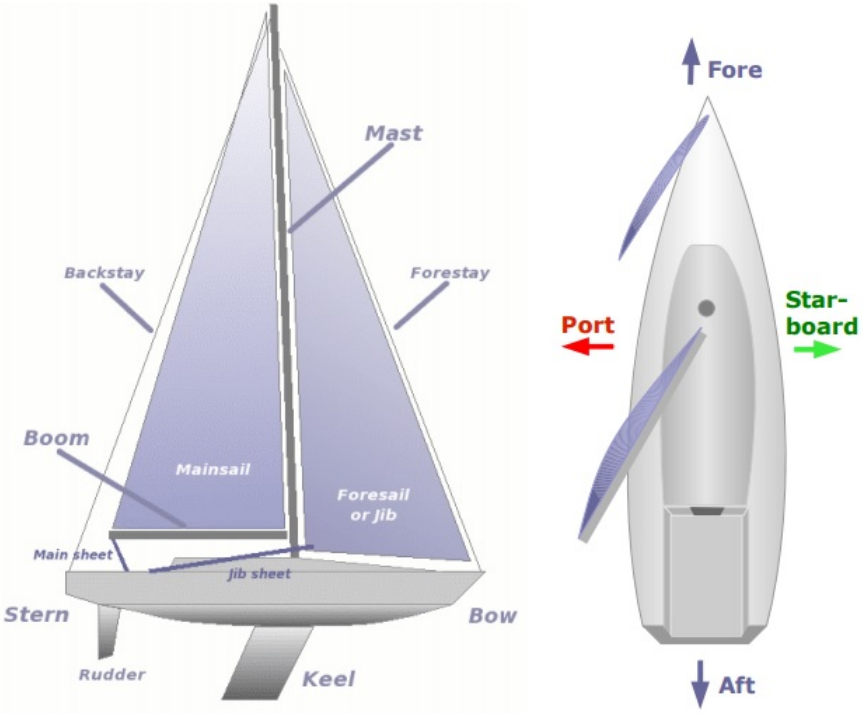


Figure 2.1: Boat Terminology

2.1.1 Directions

Fore Towards the front of a boat.

Aft Towards the back of a boat.

Port Towards the left when facing the front of a boat.

Starboard Towards the right when facing the front of a boat.

2.1.2 Structure

Hull The main body of the boat including the bottom, sides, and deck.

Bow The front most structure of a boat.

Stern The rear most structure of a boat.

Mast A structure rising above the hull that holds the sails.

Main Sail The sail located behind the main mast of a boat.

Jib Any sail set forward of the main mast.

Sheet A rope used to control the angle of attack of a sail.

Stay A rope, wire, or rod that supports the mast in the fore/aft direction.

Shroud A rope or wire that supports the mast in the port/starboard directions.

Rudder A vertical, movable fin that is used to steer the boat.

Keel A vertical structure that extends from the bottom of a boat, often with a weighted ballast, used to prevent lateral movement and stop the boat from tipping over.

Daggerboard A retractable keel, typically without extra ballast, used to maintain lateral resistance.

Heel The angle between the deck of the boat and the waterline.

2.1.3 Sailing

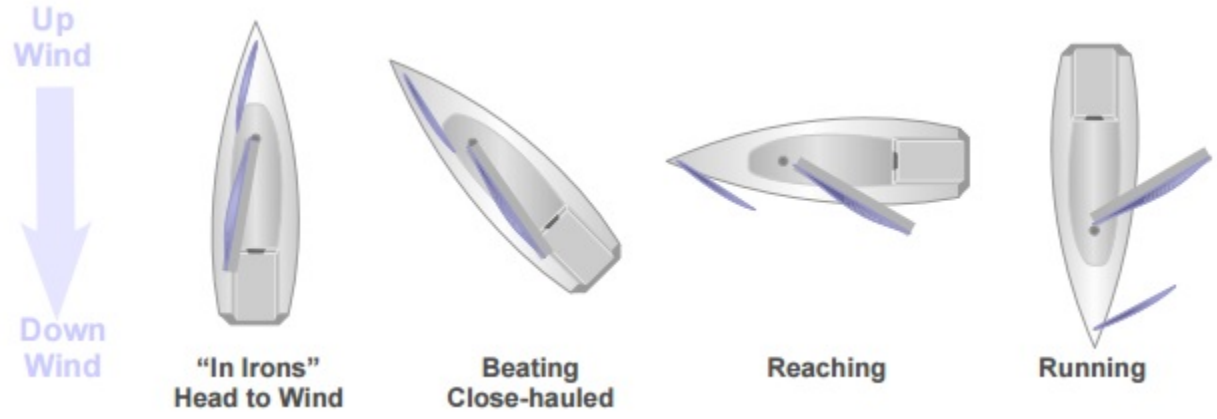


Figure 2.2: Sailing Terminology

Point of Sail The relationship of a boat to the direction that the wind is blowing

In Irons When a boat is stalled with the bow facing directly upwind.

Running When a boat is sailing with the bow facing downwind

Reaching When a boat is sailing with the wind coming from either side.

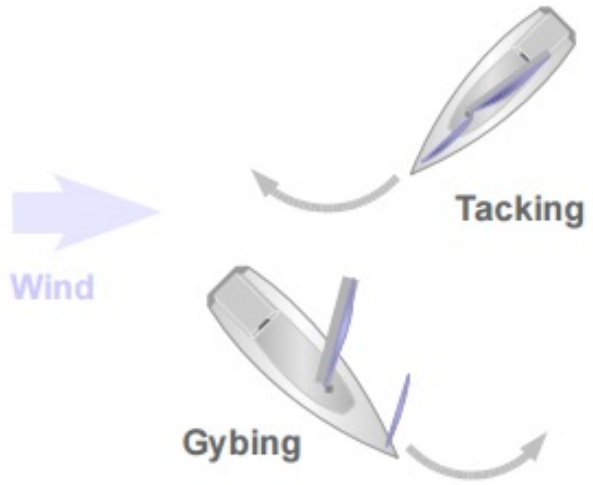


Figure 2.3: Tacking and Gybing

Tacking Moving the bow of the boat through the wind to the other side.

Gybing Moving the stern of the boat through the wind to the other side.

2.2 IRSC Robot Rules

The below rules are all pulled verbatim from the SailBot rules [17].

1. Overall length including hull, all spars and foils oriented in their fore and aft directions and at their maximum extensions if applicable, shall not exceed 2 meters measured parallel to the waterline. Sensors and their mountings are not included in the overall length measurement. Floating waterline must be clearly marked for use during measurement.
2. Beam shall not exceed 3 meters overall width at zero heel angle. .
3. Number of hulls, depth, mast height, number of masts, sail area, and number of sails are unrestricted subject to the following event limitations:
   (a) The draft in normal sailing condition shall not exceed 1.5 meters.
   (b) Total overall height from the lowest underwater point to the highest point on the largest rig shall not exceed 5 meters. Sensors and their mountings are not included in the height measurement.
4. Teams shall provide a suitable stand to support the boat in a vertical position while fully assembled for the purposes of judging.
5. Boats shall not have any direct human contact during on-the-water events except as permitted by the event Notice of Competition and Sailing Instructions
6. External control shall be limited as specified in the event Notice of Competition and Sailing Instructions, however a means for full remote radio control is required at all times during competition to allow avoidance of collisions with other boats.
7. Data transfer from the boat to shore is unlimited, but shall be on an approved frequency.

8. Radio frequencies used by each boat shall comply with host country regulations and are subject to approval before competition.
9. In each event, boats shall compete by sailing only (no alternate sources of propulsion).
10. Construction materials are not restricted provided they cannot cause environmental damage during operation. In particular, lead ballast must be completely sealed.
11. Power for onboard control systems may be provided by any source other than biological, and must be fully contained within the vessel.
12. The configuration of the boat shall not change during the course of any event. (i.e. components cannot be jettisoned or added during the race). Bilge pumps are permitted as long as they do not provide additional propulsion to the vessel.
13. Any parts may be replaced or repaired between events as necessary.
14. Sail configuration changes (hoisting/dousing) are allowed during a race provided such a change is initiated and executed only by the onboard systems.
15. In case of uncertainty about interpretation of these rules, please contact the event organizers for clarification.

2.3 IRSC Challenges

This section describes the challenges presented in the IRSC 2017 rules. All of the challenges are completed independently except for Payload, Collision Avoidance, and Search. Of the three challenges, the best single score is recorded [17].

2.3.1 Fleet Race

The boats will sail between two buoys and then will sail around two other buoys before returning to the original two buoys. Remote control must be used for steering but autonomous sail control is allowed. All the boats will sail together.

The fastest boat scores 10 points, the second fastest scores 9, etc. The minimum score is 5 points if the boat completes at lease one race.

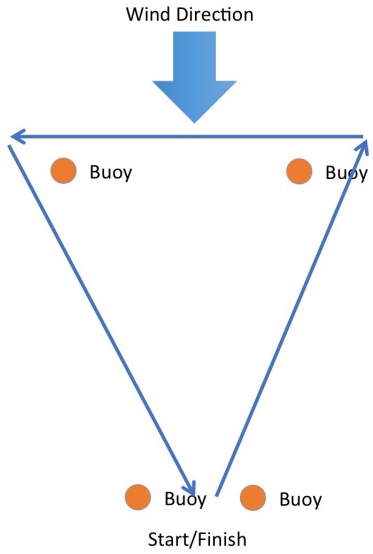


Figure 2.4: Fleet Race Diagram

2.3.2 Station Keeping

The boat will enter a 40 x 40 meter box and will autonomously stay inside the box for 5 minutes. It must then exit within a minute.

There are 10 points awarded for staying inside the box for 5 minutes; partial points are awarded for less than 5 minutes. A boat may leave and return numerous times in the 5 minutes. For instance, if the boat is in the box a total of 2 minutes it scores 4 points. A point will be subtracted for each 30 seconds over 6 minutes that it remains in the box. Performing the task in remote control will incur a 50% point penalty.

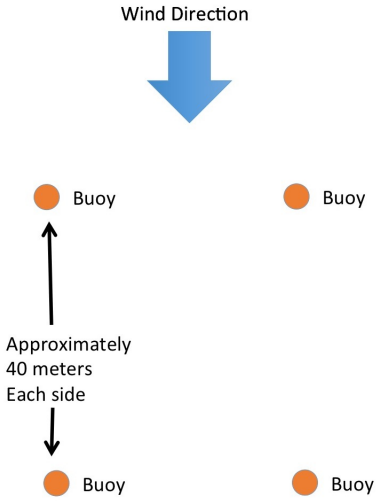


Figure 2.5: Station Keeping Diagram

2.3.3 Navigation

The boat will start between two buoys and then will autonomously sail a course around two buoys and then return between the two start buoys.

There are 2 points awarded for rounding the first buoys. An additional 2 points are awarded for rounding the second buoy. Finally, 6 more points are awarded for finishing between the start buoys or 4 points for finishing outside the start buoys.

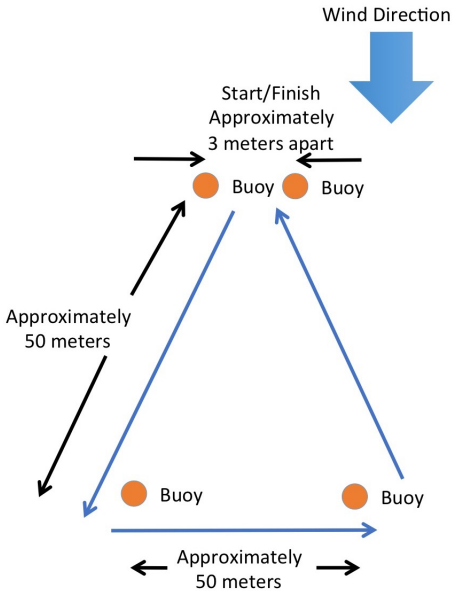


Figure 2.6: Navigation Diagram

2.3.4 Payload

Each boat starts between two buoys and will carry a (non-human) cargo of their choice and sail around a buoy and return. Remote or autonomous control is allowed.

The score is based on the weight of the cargo. The scores are calculated in relation to boat's empty weight according to the chart below:

Table 2.1: Payload challenge point allocation

Weight	**Score**
5 - 40%	1pt
41 - 60%	2pt
61 - 80%	3pt
81 - 100%	4pt
>100%	5pt

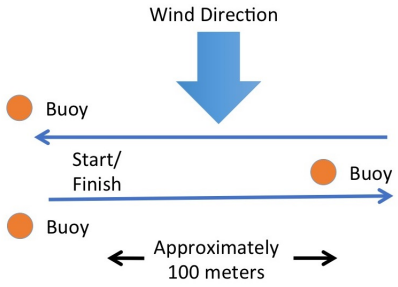


Figure 2.7: Payload Challenge Diagram

2.3.5 Collision Avoidance

The boat will start between two buoys (within 10 meters) and will sail autonomously on a reach to another buoy and return. Sometime during the trip a boat will approach on a collision course. Remote control is not permitted after the start.

If a collision is avoided then 10 points are awarded. If the boat responds to the approaching boat but a collision still occurs, 7 points are awarded. Two points will be subtracted if the buoys is not reached. If the challenge is demonstrated on land but not on the water, 5 points will be awarded.

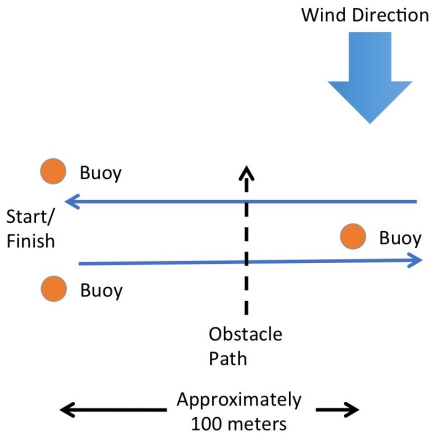


Figure 2.8: Collision Avoidance Diagram

2.3.6 Search

A small buoy will be placed somewhere within 100 meters of a starting position. The boat must autonomously locate and touch the buoy within 5 minutes by performing a standard search pattern. After touching the buoy the boat must go in to "Station Keeping" mode. The buoy will be moved after each attempt. Remote control is not permitted after entering the search area.

There will be 15 points awarded for a successful location, touch, and station keeping within the time frame. Only, 12 points will be awarded for just touching or 10 points for passing within 1 meter of the buoy. Lastly, 5 points will be awarded for performing a search pattern (Expanding Square, Sector, Creeping Line, etc.).

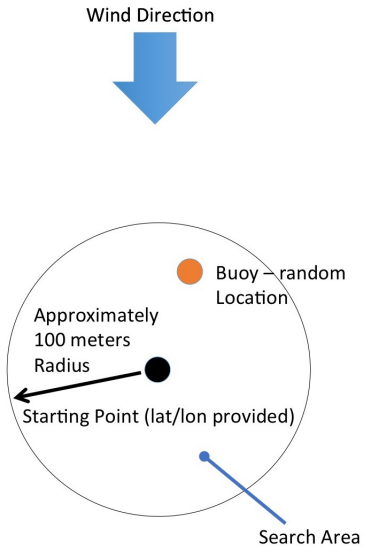


Figure 2.9: Search Challenge Diagram

2.3.7 Endurance

The boats will sail around four buoys (within 10 meters of the buoys is OK) for up to 8 hours. A single point will be awarded for each 1 nautical-mile (NM) lap completed, up to 4 points. An additional 1 point for each continuous (no pit-stop) hour sailed, up to 6 points. Remote control is allowed but will result in a 50% reduction in points (for instance, to avoid running aground). At least one lap must be completed to earn points. Using remote control to avoid collisions does not invoke a penalty.

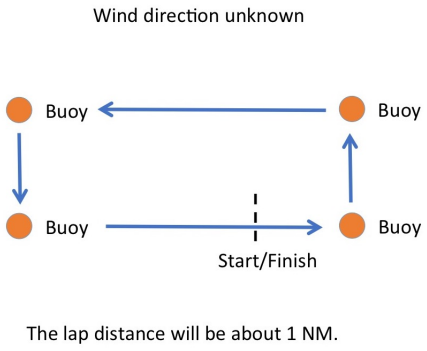


Figure 2.10: Endurance Race Diagram

2.4 Sailing Mechanics of a Robotic Sailboat

To make a robotic sailboat, some adaptations need to be made from a conventional human operated sailboat. Robotic sailboats are generally smaller that traditional sailboats for two main reasons: buoyancy and cost. Since the boats don't have to carry a large dense payload (the human), they don't need as much internal volume to displace water. Additionally, these boats are often for recreation or education, and so

the low cost that comes with the small size is highly desirable. The smaller size of robotic sailboats is also reflected in the class sizes for the IRSR (see section 2.2).

Another major change from traditional sailboats is the added actuation of all vital aspects of the boat. All sail trims and rudder control must be actuated (for robotic sailboats). The selection of actuator types and mechanical systems is bounded by the following three considerations listed below. Optimization of these three points will allow the sailboat to achieve maximum efficiency and overall faster travel.

- Maximum control over the boat in a variety of weather conditions
- Minimum power required to operate the boat
- Minimum weight required in order to optimize speed

In addition to the sail and rudder actuators, there are many systems that can be added or subtracted to enhance control and speed of the boat. Some of these systems include:

- Movable Ballast: This system attempts to provide additional stability to the boat by shifting around ballast. A few methods for movable ballast are listed below:
  - Water-Ballast: This method of movable ballast pumps water in and out of the ballast tanks located on separate part of the boat. However, due to the low specific weight of water compared to most ballast materials, this system takes up much more volume than other methods and may be less effective at providing stability.
  - Canting Keel: This system actuates the angle of the keel relative to the mast from left to right. This increases the righting moment of the boat at the most extreme heel angles allowing for more stability. Since the righting moment is increased, the ballast fixed to the end of the keel can often decrease in weight, reducing the overall weight of the boat.
  - Shifting Deck Weight: This method actuates a weight suspended on or just above the deck. This system has the advantage of simulating an actual human crew member shifting his or her weight on an actual boat. However, this method may require an active control loop in order to keep the boat from capsizing in certain situations, adding uncertainty and complexity.
- Gybing Keel or Daggerboard: This system entails rotating the keel or daggerboard by a few degrees on the vertical axis in either direction in order to allow the hull to meet the water at a more optimal angle which, as a result, decreases drag. This will allow the boat to speed up slightly especially in light to medium winds.

All of these systems increase control of the boat and decrease overall weight. Each of these systems has a mix of advantages and drawbacks which must be considered when selecting a particular design.

2.5 Rigid Wing Sail

A second MQP team was responsible for designing and building the sail for the boat. This sail is a rigidly built sail with a servo-actuated trim tab that controls the overall angle of attack of the sail. The rigid-wing nature of the sail improves the sails upwind performance relative to a traditional cloth sail, while the trim tab substantially reduces the power requirements for the sail relative to the sort of winch that would typically be used on such a boat.

Figure 2.11: The Rigid Wing Sail on the Boat on Lake Quinsigamond

2.6 Self Stabilizing Sailboat

During the 2013-2014 academic year, an MQP team designed and build a movable ballast system designed for use on a sailboat. This system used a "horizontal two-stage bidirectional telescopic slide" driven by a CIM motor and a chain and sprocket assembly to move a 70 lb weight [8]. The system was designed to replicate a crew of a 5 foot 8 inch tall person weighing 150 lbs. The final weight of the device was 130 lbs, just 20 lbs short of their attempted 150 lb target. Additionally, the system was able to generate 466 ft-lbs of righting moment, more than their requirements [8].

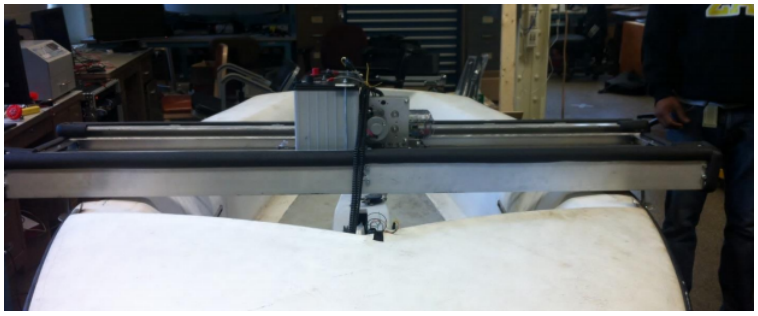


Figure 2.12: Self Stabilizing Sailboat System [8]

2.7 Previous Sailbot

In the 2015-2016 school year, a group competed in the International Robotic Sailing Regatta[1] with a 1 meter stock RC boat; it is a replica of the NZL 92 Americas Cup boat. The boat was retrofitted with a minimalistic control system. Although the boat did function when tested in Worcester MA, it had many issues while participating in the competition in Kingston:

- It could not handle high winds; even with the sails heavily reefed, it tended to heel over significantly and lose any ability to control the boat with the rudder.
- The boat was not adequately waterproof, especially in rougher conditions.
- Insufficient time was spent testing the boat, meaning potential bugs were not located in time for the competition.
- Some of the sensors were unreliable.
- Due to the small size of the boat, it was generally difficult to work with.

2.7.1 Waterproofing

The boat displayed some minor issues with waterproofing during testing, but nothing alarming. However, during the competition, the boat filled up with water relatively rapidly. The water compromised both servos controlling the winch and rudder. A multitude of attempts were made to further waterproof the boat, but water still leaked in through the interface between the keel and the hull.

Figure 2.13: The 1 meter modified RC boat used by the WPI team in the 2016 IRSR

Additionally, the electronics held inside a sealed wooden box on the deck had faults cause by water intrusion. Moisture accumulated over time resulting in temporary malfunctions, later resolved by allowing the electronics to dry out.

[1]The name was changed from International Robotic Sailing Regatta to the International Robotic Sailing Competition between 2016 and 2017

2.7.2 Boat Construction

Although the boat was a COTS (Commercial Off The Shelf) RC boat, a few issues were encountered during the mechanical construction:

- Many of the attachment points for various lines broke, requiring replacement.
- The keel board was very flexible and would bend substantially when the boat was tilted on its side.
- The number of spreaders for the shrouds were excessive (due to the boat being a scale replica).
- The small size of all of the components made them hard to work with.
- The spool for the sail winch tended to get tangled whenever it was unwound without having tension on the sheets.

2.7.3 Boat Electronics and Sensors

Without being properly waterproofed, some of the electronics may have become unreliable due to dampness. In particular, the IMU that was used (BNO055) periodically malfunctioned. The main control board, a BeagleboneBlack, also malfunctioned during the competition.

Each component that was unreliable is listed below with a description of the situation in which malfunctions occurred.

GPS GP-635T, had issues getting satellite communications from inside of the electronics box.

IMU BNO055, 9-axis IMU, would randomly fail when it encountered water.

Wind Sensor This was a custom built wind vane mounted on a US-Digital analog encoder. It was mounted on top of the mast and worked reliably enough that it was never the main point of failure.

Servos The servos functioned, but there were serious concerns about water intrusion. These specific servos also didn't provide any position feedback

Xbee An Xbee was attached to provide communications with the shore. This was never extensively tested or used.

Beaglebone Black This acted as the main controller; There were some inconveniences due to 3.3V vs. 5V logic levels, but it worked fine until used in Kingston, where there were issues with water.

Arduino An arduino was always attached (to provide a 5V interface for the BBB). It functioned but was inconvenient to work with due to the lack of any ability to reload code in real time.

2.7.4 Boat Software

Although the software appeared to function, it was never extensively tested.

2.8 Existing Sailbot Entries

8 teams, including WPI, competed in the 2016 IRSR. Below, we list some of the notable features of each teams boats:

US Naval Academy This was the best performing boat, with a 2 meter hull and 3 separate sets of sails for different wind conditions. It had a relatively old control system that had been used on multiple previous boats, and no particularly advanced sensors or electronics. The hull and keel were highly developed, but both were standard and didn't have any unique features.

Queens University The most notable feature on this boat was the use of a canting keel. There were apparently some general issues with the complexity and power involved in the mechanism and it is unclear the extent to which the canting keel actually helped.

Aberystwyth University This team used a "MaxiMOOP" hull, which is a simple and cheap design that is advocated strongly by Paul Miller (the USNA professor).

Olin College This team ran with an approximately 4 meter sailfish as the base for all of the electronics. The boat overall was somewhat slow, although it was capable of completing all the challenges. It had issues with waterproofing. This boat also dominated the payload challenge.

Virginia Tech This team was the only team to attempt the vision challenge. They had a boat similar in style to the USNA, although it performed at a much lower standard.

The USNA was the only team to score points in all the challenges (except for the Obstacle Detection, which was technically optional, as it could be replaced with the payload challenge). They were the only team to earn points on the Navigation challenge and were the only team to earn more than 3 points on the long distance challenge [17].

2.9 Sailbot Controls Literature

While the field is not particularly large, there is a body of work surrounding the control of robotic sailboats. Essentially all the existing work assumes cat-rigged boats with a rudder and no other actuators, although Xiao and Jouffroy performed their calculations for a movable mass steering system in addition to the standard rudder in [26].

In [21], Saoud et. al. presented first a 6-DOF model of the rigid-body dynamics of the boat, followed up by a controller design based on simplified 3-DOF (i.e., x, y, and heading) dynamics. Their sail trim controller set sail trim based on maximum forward thrust from the aerodynamic forces on the sail. Their rudder controller was a PI-controller which assumes that the rudder always has sufficient authority to turn the boat.

This is followed up by Saoud et. al. in [22] in which a navigator was developed which allows the boat to navigate around obstacles and go into regions with better winds. Furthermore, the rudder heading controller was modified to account for the leeway (sideways motion) of the boat, meaning that the boat would actually sail in the direction that was desired, rather than just being *pointed* in the desired direction.

However, both of Saoud's solutions had a couple of potential shortcomings:

- Rather than optimizing sail trim for maximum forwards speed or maximum headway, they were optimized for the aerodynamic forces on the sails. This doesn't always translate purely into actually moving. In practice, the two are highly correlated, but not necessarily identical.
- As is explicitly called out in [21], the rudder controller does not account for rudder stall, or the limited amount of torque available to the rudder. This could lead to the controller accidentally stalling the rudder in certain situations, or with the controller attempting to tack when it does not have enough turning authority

Jouffroy and Xiao investigated just heading control (no sail trim) in [26]. 4-DOF dynamics were developed (x, y, heading, and heel), as well as accounting for sail luff in tacking.

2.10 NMEA 2000

Electrically, much of the system will be based around a NMEA 2000 bus. See [6] for an overview of NMEA 2000 from NMEA. See CANBoat [1] for a reverse engineering of the NMEA 2000 standard. Both sources inform the remainder of this section.

NMEA 2000 is based on a 5-conductor backbone, consisting of:

- A grounded shield
- Ground and 12V for power
- CAN-H and CAN-L for the CAN bus

The core part of NMEA 2000 is the CAN bus; all communications run along it and most of the NMEA 2000 standard (that will be relevant to us) consists of regulating those communications. NMEA 2000 is based on the existing SAE-J1939 standard, which is used in automotive applications, and as such has many similarities.

Structure of a NMEA 2000 message

NMEA 2000 messages are based on a standard CAN message, which itself consists of:

- A 29-bit ID; the smaller the value of the ID, the higher priority the CAN message takes when arbitration occurs on the bus [2].
- A 4-bit Data Length Code (DLC), with a value from 0-8.
- A data field, up to 8 bytes long (length specified by DLC).
- Various other padding, reserved bits, check-sums, and acknowledgment bits. These aspects are handled by NMEA compatible hardware and don't influence the higher level use of the bus.

What NMEA 2000 does is specify a standard format for the contents of the ID and, depending on that ID, provides a specification for the format of the data field.

The CAN ID is filled as follows:

Table 2.2: NMEA 2000 CAN ID Bytes

Bits	Field Name	Description
26–28	Priority	Allows user to set message priority
25	Reserved	
24	Upper PGN Bit	Highest order bit for the PGN
16–23	Data ID Byte A	High-order byte of the Parameter Group Number (PGN)
8–15	Data ID Byte B	Either lower-order byte of PGN or destination address for non-global messages
0–7	Source address	

If the Data ID Byte A is less than 240, than it is a PDU1 format message, and so byte B is a destination address and the upper PGN bit is ignored.

Although they have not yet been discussed, this brings in two important concepts with regards to both NMEA 2000 and J1939:

Parameter Group Number (PGN) The PGN is a 17-bit number that defines the format of the message. NMEA maintains a list of PGNs and the corresponding formats as part of the NMEA 2000 standard. A typical PGN might be something indicating a wind information message, which corresponds to a well defined format for the message data. Any PGNs that are expected to have a destination address will, by necessity, have a lower byte of 0 and so be multiples of 256.

[2] A note on arbitration: When a CAN message is being sent out on the bus and two devices happen to start sending at the same time, arbitration will occur. The first device *not* to assert a dominant bit (a 0) loses arbitration and waits until the other device has finished its message before trying again.

Address Although our code is not fully NMEA compliant in this regard, there is an arbitration process for addresses on the bus which corresponds to a particular set of PGNs which are used by devices joining the bus to discover what addresses are available and to claim an open address.

Because CAN only allows for 8-byte messages, NMEA 2000 allows for multi-packet messages. A multi-packet message will have the first byte of the data segment contain a frame counter, with the second byte of the first data packet containing the total number of bytes of actual data in the message. The last 5 bits of the frame counter are used to indicate which packet within a message the current packet is; the first 3 bits are used to distinguish between messages (so that if a device sends two multi-packet messages in rapid succession, they won't have overlapping frame counters; one will index off of 0x00 and the other off of 0x20/0x40/0x60/0x80/0xA0/0xC0/0xE0).

The data are then interpreted according to the PGN, where the formats are well-defined by NMEA 2000. While we do not have direct access to the NMEA 2000 standard, the CANBoat project has reverse-engineered many of the common messages (including many of the wind/position data that we are dealing with in the Airmar 220WX), any we define new PGNs for our own use between the SCAMPs.

3. Software, Controls, and Electronics

3.1 Software Platform

Most of the main code for the sailboat was written in C++ and runs on a BeagleBone Black running Debian Linux. The BeagleBone connects to the boat systems via the NMEA 2000 CAN bus and connects to the shore, rigid wing sail, and any other auxiliary systems via WiFi. All of the boat code can be found on GitHub, see [5].

3.1.1 BeagleBone Black

The BeagleBone Black is a small, credit-card sized computer with a 1GHz AM335x ARM Cortex-A8. It has various I/O pins and, most importantly for our efforts, has on-board Ethernet and CAN. There are several revisions of the BeagleBone Black. We used the most recent one (the Rev C). The main difference between the Rev C and Rev B was an increase in space to the built in persistent eMMC. The Rev B's main change from the various Rev A versions was a change the exact model of the processor [14].

For the CAN, the AM335x does have CAN built in but still needs a transceiver chip to actually connect to the CAN bus. For this, we used a Waveshare Beaglebone RS485/CAN Cape. The only portion of this that we actually used was the MCP2551 transceiver chip [15].

BeagleBone Black Setup

In order to set up the BeagleBone, there were 4 main steps that we took:

1. Installed a new image of Debian Jessie on a microSD card to use as the main filesystem and Operating System for the BeagleBone
2. Updated the C++ libraries to match the versions being used by our cross-compilers
3. Configured the Ethernet network by modifying the `/etc/network/interfaces` file
4. Installed the SocketCAN [3] libraries and ensured that the CAN interface was brought up on boot

3.1.2 WiFi

As mentioned in section 3.4.2, we use a Bullet M Titanium for the WiFi radio [16]. This is configured as an Access Point with WEP security. The purpose of the security is primarily to prevent accidental connections to the network while avoiding having to deal with any hardware that may not support more secure methods, such as the various WPA versions[1].

In order to configure the Bullet, we:

1. Reset the radio to factory settings by pressing and holding the reset button for 8 seconds [25]
2. Navigated to the configuration page, which is at 192.168.1.20
3. In the first menu, disabled "AirMax" [24]
4. Set the channel width to 20 Mhz (the default 40 Mhz does not work for most other radios)
5. In the network settings, set to bridge mode and configured for a static IP address

[1] For the uninitiated, WEP is an old WiFi security standard that is now considered obsolete and almost entirely unsecure [19]

6. Powered the radio with 24V using Power over Ethernet, by splicing out the brown and blue twisted pairs (corresponding to pins 7/8 and 4/5 respectively) and putting ground on the brown (7/8) and 24V on the blue (4/5). Note that, in 10/100 Ethernet as is being used in this application, these wires would not normally be used and so there is no need to worry about maintaining a connection through the entire cable as there might be with gigabit Ethernet.

3.1.3 Auxiliary Systems (RPi)

While we did not complete any computer vision tasks as part of this MQP, we did do some initial setup with a Raspberry Pi Zero W to make it possible for future work to introduce computer vision to the boat.

To do this, all we did was flash a version of Raspbian onto a microSD card, setup the network to connect to the WiFi, and set up a cron job to run on reboot and record video that we could then examine later.

3.1.4 Cross-compilation and Build System

In order to easily compile code for the BeagleBone while also being able to run and debug code on our personal machines, we needed to set up a build system and cross-compiler to build code for the BeagleBone.

For the build system, we chose to use Bazel, a build system created by Google [13]. For our purposes, the main features of Bazel are:

1. Supports C++ well
2. Creates highly reproducible builds and can be set up to easily download and build dependencies without requiring developers to have exact versions of various libraries on their system
3. Relatively clean build file syntax–it is easy to add new dependencies, add new build targets, and generally make modifications to the build files

The main drawback of using Bazel is that it has not existed for as long as some other build systems and so does not have as large a body of online resources and documentation as more traditional systems, such as cmake, do.

The greatest issues in getting Bazel to work were encountered in getting the cross-compiler to work. This was mostly an issue of getting the right files and right configurations in the right places, and the final configurations can be seen by viewing our code [5].

3.1.5 BeagleBone Scripts

In order to make running the code on the BeagleBone easier, we have several scripts for convenience:

startup.sh A file that is set up to run on boot. This initializes the CAN interface and starts up all the code, including setting the name of the logfile to correspond with the current clock time.

bringup-can.sh A short script to initialize the CAN interface on the Beaglebone

killall.sh A scrip to stop all the sailbot related processes on the boat and clean up the resources that they use on the system

3.1.6 Unit Testing

In order to facilitate unit testing of portions of the code which are easily testable (i.e., not the lowest level interactions with the hardware), we use the gtest libraries [2] and Bazel's cc_test built target type. All the unit tests are then run on a clean build in a clean Travis-CI system that triggers a build whenever a commit is made to the github repository.

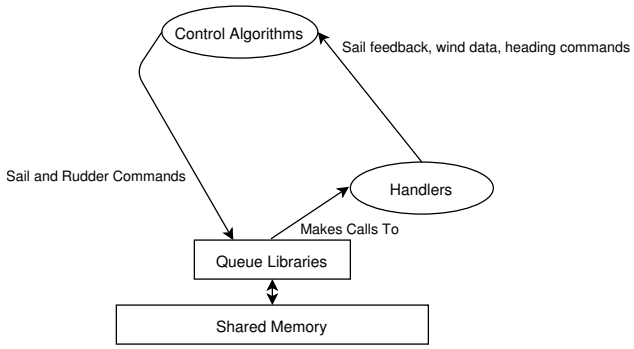


Figure 3.1: Basic structure of the low level control node

3.2 Software Architecture

The BeagleBone software consists of various processes which communicate with one another using a publisher/subscriber style.

For an example, consider the basic controller node in Figure 3.1, which takes a heading command and the boat state data and spits out rudder and sail commands.

The main code is a set of control algorithms that runs in the main loop for the code. This code makes calls to our queue libraries which then actually put the sail and rudder commands into shared memory. These commands are then dealt with by another node. On the receiving end, the queue libraries keep an eye on the shared memory and whenever new information (wind data, sail position data, heading commands) appears it calls some handlers in the code which then actually set the values of some member variables in the code. These member variables are accessed in the main loop and used to calculate the new sail/rudder commands.

To see all the nodes and what queues they each use to receive and send information, see Figure 3.2. Of note in Figure 3.2 are:

- The "everything" bubble, which is used to represent how the logger and UI server send and receive data from all the various processes. In practice, the only messages that the UI typically sends out onto the queues are waypoints which are sent to the tacking code.
- The rectangular boxes, which correspond to various libraries that interact with the world outside of our code.

3.2.1 Interprocess Communication (IPC)

In order to have real-time performance in our systems, we had to have real-time interprocess communications. In order to accomplish this, we use the Boost Interprocess libraries [9], taking the IPC queues that they provide and making alterations in order to make it so that, rather than the queue only allowing each message to be received by a single process, each message would be seen by every subscribing process. The boost IPC queues use shared memory in order to communicate, meaning that the communications should be as real-time as the linux system libraries and RAM access.

However, the shared memory queues are not in and of themselves enough to provide a clean interface. In order to transfer data in a useful, structured manner, we used Google protobufs [10].

The protobufs allow us to define message structures per the google protobuf format. The protobuf libraries then handle serializing and reading the messages from raw memory. The primary advantage of using these over using raw C structs and just writing them to memory, is that this way it is feasible to add

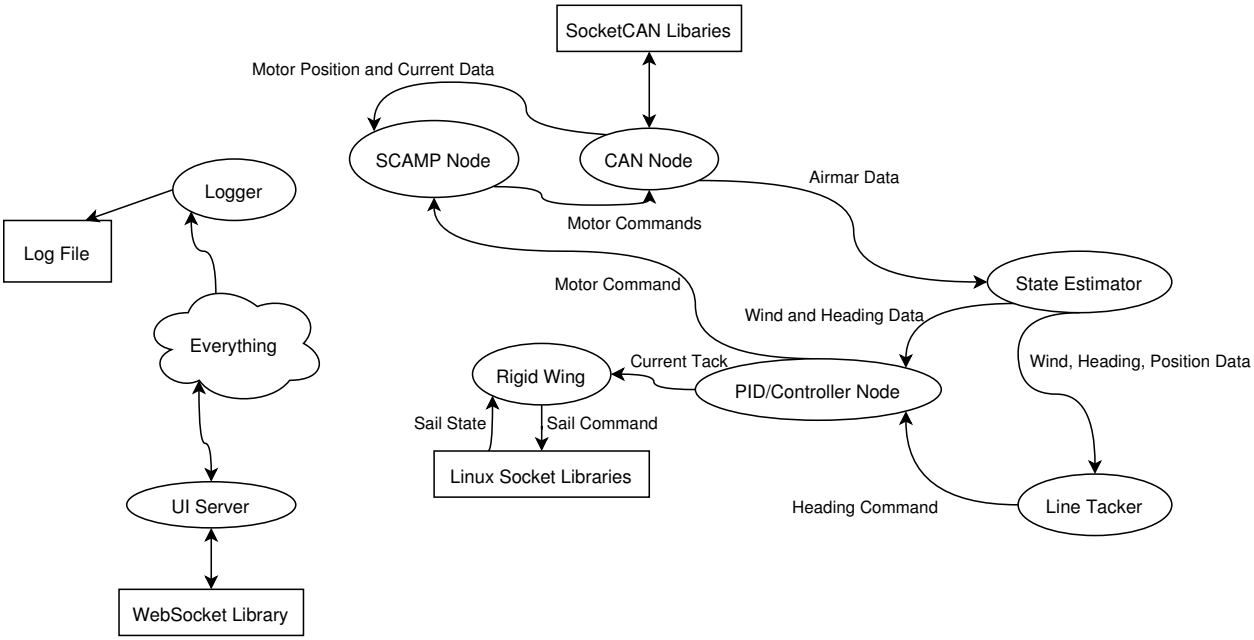


Figure 3.2: Overall Code Structure

and alter the message types at a later date while still having the original messages be readable. Using a raw C struct just written to memory would tend to be extremely fragile and error-prone.

Finally, in order to make logging easier and more robust, we make a master protobuf structure, which contains a single member for each individual queue, as well as a single timestamp field. Because each field is optional, there is relatively little wasted space due to this structure, while it allows us to use the same, single message type for each instance in the logs. It also allows us to detect which queue was sent on the log by querying the protobuf library to determine which member(s) are set.

While we have not done a thorough analysis to check for real-time behavior of the protobuf library, we do use pre-allocated pools of memory (referred to as "arenas" by the protobuf library documentation) to make it so that there should not be any calls to malloc during the main execution of the program.

Test Mode

In order to speed things up while testing and avoid using shared memory, we also provide a secondary queue system whereby, if the user specifies the correct flag and all the nodes running are running as part of the same process (so IPC is not needed), the shared memory queues are simulated by standard library containers and functions.

3.2.2 Logging

There are two sorts of logs that we have while the code is running:

- Logs of all the messages sent on the queues, stored by simply writing the serialized messages to a file on disk
- Logs which are essentially just print statements in the code, with varying priority levels and switches to turn them on/off when debugging or running in real life

Queue Logging

In order to log all the queues, we run a logger process that subscribes to every single queue. When it receives a message, it takes the raw serialized data and appends it to a write buffer. This buffer is then written to a file on disc whenever it can. This process runs at a lower priority than all other sailbot processes, as it does not need to be real-time and disc I/O can be extremely time consuming.

Printf Logging

This makes use of the glog libraries, allowing multiple priority levels and some convenient macros, such as things for checking errno, and a "FATAL" log-level for aborting the program with a message.

These libraries are not real-time. As such, all log statements should either be marked as debugging and so not run during the deployed execution of the code, or the statements must be in portions of the code where a break from real-time behavior is reasonable, for instance when an error occurs and it should be noted.

3.2.3 Log Replay

By simply reading the logged queues and sending them out again we can simulate the state of all the messages on the boat after the fact. We can then, for instance, run a single process (e.g., the state estimator) and ensure that its output, when given the real data, matches what we would expect.

3.2.4 Timing and Nodes

Each process running on the boat is considered a "node" and, for the most part, they all have similar requirements:

- Some regular, timed loop (e.g., 100Hz)
- The ability to easily send/receive queue messages
- A way to cleanly exit when SIGINT (Ctrl-C) is received
- The ability to spoof a clock to behave appropriately when performing tests on a personal machine

As such, there is a Node class which provides various utility functions for all of these. There is some complication inherent in creating a fake clock, as various synchronization mechanisms must be employed to ensure that no piece of the code gets stuck or gets ahead of another and to avoid deadlocks, all while still allowing the code to cleanly exist should it receive a SIGINT. The fake clock only works for single processes, although it does allow multiple nodes running in different threads within the process. The fake clock and test queues are commonly used in conjunction with one another to create a single process that runs all the necessary nodes to simulate the boat.

3.2.5 CAN (Controller Area Network)

For CAN, as mentioned in section 3.1.1, we use SocketCAN [3]. This allows us to treat the CAN interface as if it is a socket and use standard system library calls on it. Most of the configuration we do is to set it so that the calls to `read` are blocking, but timeout after a second. This prevents a situation where if the process receives a request to stop (a SIGINT) while waiting to receive new data, it will not block forever.

Additionally, when sending messages we do set a bit in the can id that is passed to the SocketCAN library to indicate that we are using Extended CAN Frames, as required by NMEA 2000, rather than "Standard" size frames.

When a CAN message is received, it is sent out on a queue that is named after its PGN (e.g., wind data, pgn 130306 is sent out on queue `can130306`). If a process other than the main CAN node itself sends a

Figure 3.3: The main UI view

message on the CAN queue with the `outgoing` bit set to true, then the CAN node will place the message in the proper NMEA 2000 format onto the CAN bus.

3.2.6 Rigid Wing

The communication with the Rigid Wing Sail is done through a socket that we serve from a node on the BeagleBone. See the Rigid Wing-Sailbot interface document in Appendix A for the full details of the communications. All that this node does is take messages from the appropriate queue and push those out over the socket to the rigid wing, while also publishing messages from the rigid wing out onto the queues.

3.2.7 UI

At a high level, we use WebSockets (specifically the uWs library [4]) to allow anyone on the same network as the BeagleBone to read and write from all the queues on the BeagleBone.

While there is essentially no security, this does mean that it is easy to write a webpage using Javascript to debug the boat, or to write a Python interface to read from a joystick and send appropriate motor commands to the boat from a laptop on shore.

Server

The server code running on the BeagleBone is a low-priority process which does the following:

- Subscribes to every queue
- Stores the most recent message for each queue
- Runs a uWs websocket server
- Parses the JSON messages from the websocket to either send queue information out to the websocket or to take a queue from the websocket and put it on the shared memory queues

Webpages

The webpages are run via Javascript and HTML, using the Javascript Websocket libraries for communications. We have various utility functions to make it easy to add fields to an HTML page that either display the current status of a queue or can be used to send values out on the queue, as well as some other functions doing thing such as easily creating handlers to listen for values on specific queues.

For the stick-diagrams of the boat, we use SVG and update the SVG values directly through the Javascript.

JSON Message format

The format of all JSON messages is as an object where each key is a queue name and the value corresponding to the key is either the message corresponding to that key (as can be parsed by Protobuf's JSON parsing mechanisms) or null. If null, then it means that the client is requesting the value of that queue.

The server will also reply to all requests with at a minimum a special key `time` which contains the current system time.

3.2.8 Controls Architecture

There are 4 primary nodes in the pathway for going from waypoints and sensor data to raw commands for the servo and winch (or rigid wing):

SCAMP Node This is a low level node that takes in winch voltages, rudder angles, and CAN messages from the SCAMP and sends out CAN messages for the SCAMP as well as publishing the current rudder and winch position. The SCAMP node also manages the mode for the motors, preventing any command from going through if the boat is disabled.

State Estimator This takes the various CAN messages sent by the Airmar and extracts/transforms the relevant data so that it is easily usable by us. This includes, for example, changing the frame of reference from compass headings (North is 0, angles increase in the Clockwise/Easterly direction) to our normal reference system (East is 0, angles increase in the Counter-Clockwise/Northerly direction).

Simple Controller This handles the PID control for the winch, reading the wind values from the state estimator and the pot values from the SCAMP to determine a voltage to send to the winch motor. It also handles taking the current goal heading and current boat speed and setting the rudder angle appropriately.

Line Tacker This takes the list of waypoints and state information (the wind and current boat heading) and determines what the goal heading should be.

3.2.9 Simulator Architecture

For the simulator, we need something that essentially spoofs everything that goes through the main CAN node, as well as something to spoof the rigid wing.

This is accomplished by having two main categories:

1. A physics simulation of the boat that does all the heavy lifting for the calculations
2. A couple of nodes that act as fake sensors, taking the true state of the system from the physics engine and spitting out values in the same way that one of the sensors would, including adding in some noise

There are a couple of physics setups that we have, all of which can replace each other without issue. The main difference is that one physics engine is substantially simpler, whereas the other is, in theory, higher fidelity, but in practice is just harder to debug and doesn't work entirely correctly.

3.3 Controls and Simulation

3.3.1 Simulation

Physics

For the physics simulation, the initial pass was based upon the physics presented in [21], but for ease of debugging and because we do not have good enough data to create a higher fidelity simulation, we simplified things substantially. We do use the same coordinate systems and reference frames as presented there.

Overall, we assume that forces on the boat originate from the following:

- Gravity on the center of mass
- Buoyancy of the boat
- Aerodynamic lift and drag on the:
  - Sail
  - Rudder
  - Keel
- Some damping terms from the hydrodynamic forces on the hull
- Gravity on the movable ballast (if the movable ballast is enabled)

We calculate all the moments about an arbitrary point on the deck along the line that goes through the center of the boat fore-aft. The x-axis points forwards in the boat, the y-axis points to port and the z-axis points upwards.

For our dynamics, we only worry about the boat's movement in the x, y, roll, and yaw axes. For the rotation vectors, when moving from the inertial frame to the boat frame, we first translate by x and y and then rotate for yaw about the z-axis and then rotate for heel about the boat's x-axis.

The only actuators that we assume that we have are the rudder, sail, and ballast angles. For all the actuators, an angle of zero corresponds to being all the way in and in line with the boat's x-axis. Positive angles correspond to the body of the actuator being on the starboard side of the boat (i.e., a positive sail angle means that the boat is on a port tack).

For the buoyancy term, we assume a constant point in the boat's reference frame as the center of buoyancy and apply a constant upwards force equal to the weight of the boat.

For the hull damping term, we assume damping about the roll and yaw axes proportional to the angular velocities about each axis.

Finally, for the aerodynamic forces we assume that the sail, rudder, and keel all behave *similarly*, albeit with different constants. We approximate the coefficients of lift and drag at various angles of attack based on the original values from the rigid wing group's sail and tweaking stuff to see what functioned reasonable. We ended up with the drag coefficient as being proportional to n^{α^2} for some angle of attack α and some constant n. The lift coefficient we approximate as increasing linearly until the angle of attack with the max lift coefficient and then having it decrease linearly to zero from there.

Faking Sensors

The noise for the sensors is just introduced by adding zero-mean gaussian noise to the output values. Other than that, it is just a matter of ensuring that the output values of the fake sensors are in the same reference frames and scales as the real sensors provide.

3.3.2 Controls

The final controls algorithms we used a relatively simple. We tried investigating some of the algorithms described in section 2.9, but never developed to the point where we could productively debug them on the real boat. The algorithms described here are the ones that were running on the boat as of the writing of this report.

Sail and Rudder

For the sail, we have a simple proportional controller that drives the sail based on the current apparent wind angle. We denote the apparent wind angle by $a_w \in [-\pi, \pi]$, where $a_w = 0$ if the boat is pointed straight upwind and $a_w = \pi/2$ if the boat is on a beam reach on the port tack, the current sail angle by δ_s, the goal sail angle by r_s, and the sail winch voltage by V_s, with K_s a tuning constant. Note that there are physical limitations on the values of V_s:

$$\begin{aligned} r_s &= a_w/2 \\ V_s &= K(r_s - \delta_s) \end{aligned}$$

For the rudder, we have the same essential idea, except that we add in some consideration for the current speed of the boat. For the rudder, we have a goal yaw ψ_g, a current yaw ψ, a goal rudder position δ_r, the current speed of the boat $v_s = ||v||$, with a function for constraining δ_s, $\delta_{max}(v_s)$ and constants K_r, K_{vr}, $\delta_{r,slow}$, $\delta_{r,fast}$:

$$\begin{aligned} \delta_s &= -K_r(\psi_g - \psi)\frac{K_{vr}}{\max(v_s, K_{vr})} \\ |\delta_s| &\leq \delta_{max}(v_s) \\ \delta_{max}(v_s) &= \begin{cases} \delta_{r,slow} & v_s \leq K_{vr} \\ \delta_{r,fast} & v_s > K_{vr} \end{cases} \end{aligned}$$

Effectively, this means that at low speeds, the rudder will not be used too much and so won't act as a brake on the boat, and at high speeds the boat doesn't need to use the rudder as much because it has more authority anyways.

Tacking

The tacking code takes in a set of waypoints and determines what the current goal heading to pass to the rudder controller above should be. For these purposes, we ignore the nature of the spherical coordinates we get from latitude and longitude and treat the boat as if it is on a flat plane.

Broadly speaking, what the tacking code does is:

- If the boat can go straight to the heading, it does
- If boat must go upwind to reach it's target and can't go straight there, then consider the line straight through the two waypoints that the boat is traveling between and then if the boat is too far (per some tunable parameter) from that line, then the boat will turn to whichever tack will bring the boat back close to the line. So long as the boat is close enough to the line, it will just go on whichever close-hauled heading is closest to its current heading

However, the above approach alone will fail under two circumstances:

1. Inconsistent winds, where the boat may end up rapidly changing tacks despite not changing its heading much, and as a result the goal heading may change rapidly, causing the rudder in turn to change a great deal, slowing the boat down while accomplishing little else

2. Low boat speeds, as if the boat has too little speed when it attempts to tack, then it will fail to complete the tack and end up dead in irons

As of this writing, we are still working on addressing these and testing the solutions properly.

3.4 Electrical Architecture

The electrical architecture of the boat includes power distribution, communications, and various components of the boat. In this section, we go into detail explaining the functionality of most of the components and the process for designing custom components. An overview of the electrical architecture is shown below.

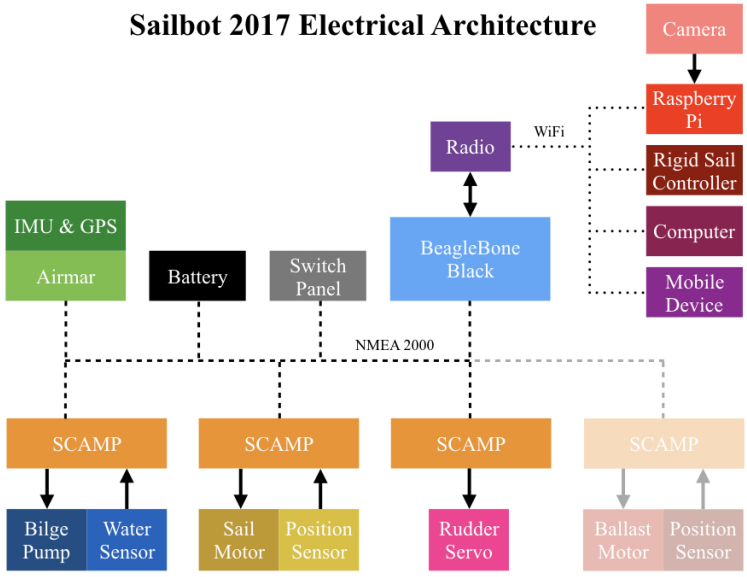


Figure 3.4: Electrical Architecture

3.4.1 NMEA 2000

The boat has a variety of sensors and actuators, all requiring power and some for of communication. To clean up the electrical system and reduce shorts and wiring errors, having a single unified communication and power channel is desired. The NMEA 2000 standard fulfills these requirements and has some features desirable for our design [6].

1. It interfaces directly with one of the most important components on the boat: the Airmar 220WX

2. It has a CAN bus with allows relatively high bandwidth communications between any 2 components

3. The connectors are standardized and designed to be water resistant, reducing our concerns of water related electrical problems [6].

In addition to the NMEA 2000 cable itself (which carries 12V), we also ran larger gauge power wires for higher current components like the winch motor.

3.4.2 Components

BeagleBone Black

As described in section 3.1.1, the BeagleBone Black is the heart of the controls system. It interfaces with the NMEA 2000 and requires 5V to operate.

AirMar

The Airmar 220WX is the main sensor for wind direction and speed. It also incluse a GPS receiver and an IMU for detecting roll, pitch, and yaw. The Airmar 220WX interfaces directly with the NMEA 2000 bus and has a suite of standardized messages that it broadcasts informing other devices of the wind speed and GPS data [12].

5V Regulator

A 5V regulator was used to step the voltage down for certain components including the rudder servo. Because the rudder servo requires a nontrivial amount of current (averaging around 0.5 Amperes, peaking at 2-3A), we used a Buck converter rather than a linear regulator which would have been extremely inefficient given that it had to step approximately 12V down to 5V, and so would have to disperse on the order of 4W $(0.5A * (12V - 5V))$ on average.

Battery

We used a variety of batteries throughout testing. At first we used a readily available 12V 2.2Ah LiPo quad-copter battery. In a variety of conditions with both sails, this battery lasts about 2.0-2.5 hours. The main battery that will be used for the competition is a 12V 10Ah NiMH battery. The NiMH chemistry is more forgiving than LiPo in higher current applications similar to what we expect with the winch and rudder. It also requires less protection and is more robust than the LiPo [7]. This battery is expected to run for around 8 hours depending on weather conditions.

Switch Panel

There is a switch panel for quick access to power and component status. The panel features a main power breaker connecting the battery to the rest of the electrical system. It also has a disable/enable switch for quickly stopping motor operation without interrupting running software. There are a number of status LEDs indicating communications with the BeagleBone Black and each SCAMP (explanation below). The panel also has an LCD to display various information such as battery voltage, Wifi status, and way-point data.

WiFi Radio

The boat uses a Bullet M Titanium Wifi Radio for all Wifi communications. This radio is powered via POE (Power Over Ethernet) and connects directly to the BeagleBone Black. This radio was chosen over a more typical radio because it is relatively high power/gain and allows us to attach an antenna with a high enough gain such that, when sailing the boat it provides around 200 yards of range when connected to a standard laptop computer. This range may be increased by using another more powerful or sensitive radio/antenna instead of a standard laptop antenna [25].

3.4.3 SCAMP (Sensing and Controlling Atmega Messenger and Processor)

The SCAMP is one of the most important components on the boat. It is the interface between the BeagleBone Black and many of the sensors and all of the motors.

Research Into Existing Options

After choosing the NMEA 2000 standard for most of the communications on-board the boat, we researched speed controllers for NMEA 2000. One component that is similar to what we need is the Talon SRX [11]. It communicates over CAN and has some additional functionality including PID control and sensor inputs. Unfortunately, it isn't NMEA 2000 compatible as it runs at a different bit rate (1 Mbps instead of 250 Kbps [20]). More research into the topic showed no results as NMEA 2000 is mostly designed for sensing and displaying information in marine applications rather than low-level motor control. The solution was to design our own NMEA 2000 speed controller.

Hardware/Electrical Requirements

The SCAMP has a few key requirements and some additional ones that allow it to be used in place of many other COTS (consumer of the shelf) parts. The 2 main requirements are:

- Driving the highest current motor (2-3A continuous, 15A peak)
- Receiving and sending CAN messages

Other features that would be nice to include:

- Digital inputs/outputs
- Analog inputs/outputs
- PWM output for controlling a servo
- Quadrature encoder support

Design Process

To start the design process we first experimented with an Arduino Uno and commercially available CAN chips and shields. This was essentially a proof of concept. Next we started creating the electrical schematic for the board. The Atmega328P was selected as the main processor because it is very popular and allows us to use the Arduino Libraries for quick development. We then added the CAN chips and the necessary hardware to allow those to function. The next big feature to add was a motor driver. There were a few options for this, but ultimately we settled on the VNH2SP30 motor driver. It met our requirements of 2-3A continuous and 15A peak and also had a variety of other features [18]. These included:

- Small, surface mounted design (0.625in by 0.625in)
- Thermal shutoff
- Current sense output
- Simple electrical interface

After the motor driver was finalized, we added the rest of the components. Many of these were necessary to fulfill the electrical requirements for some of the components. Others were added to make debugging easier and board performance overall more reliable.

- 3.3V & 5V Voltage regulators
- 5 LED status lights
- Reset button for Atmega328P
- ISP interface for programming
- 0.1 uF capacitors for noise protection
- Pins for all inputs & outputs
- High current pins for motor output

Once the schematic was complete, the layout of components and route tracing was completed. The boards were ordered, assembled, and tested. This process was completed a second time both to fix issues with the first version and to add some additional functionality.

In addition to the boards, we also designed housings to make the boards water-resistant, and heat sinks to allow the heat generated by the motor drivers to be distributed.

SCAMP 1

The first version of the SCAMP overall was a success. The final board was 3.625in by 2.150in. It had some problems, most notably the TX/RX lines for the CAN chips were reversed. This was easy to correct with some additional wires though. The boards were relatively large and had a variety of through hole components to make assembly more easy. The Atmega328p was also removable to aid in initial programming. Below is an rendering of the board.

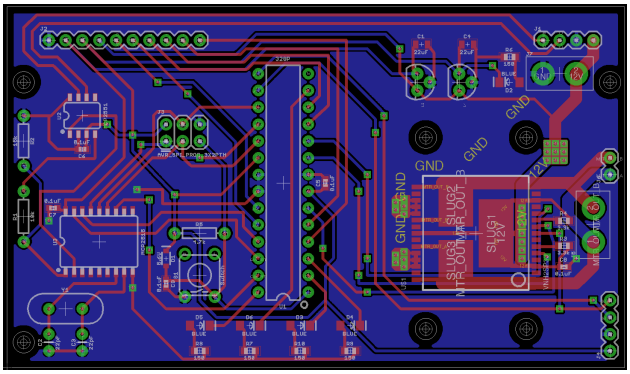


Figure 3.5: SCAMP 1 Schematic

In order to handle the heat produced by the motor driver, we designed and machined a heat-sink (pictured below on an assembled board). The wires for fixing the Tx/Rx issue are also visible. Additionally, since we were still in the testing phase at the time of the image, the SCAMP is resting inside of a Tupperware container for waterproofing.

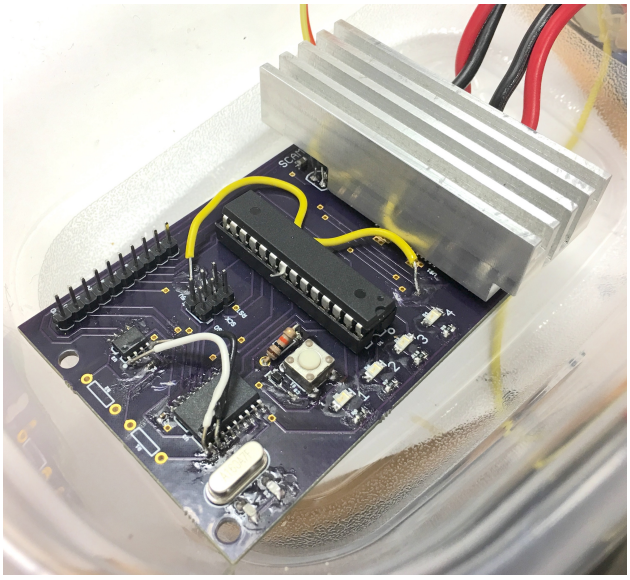


Figure 3.6: SCAMP 1 Board with Heat-Sink

SCAMP 2

After producing SCAMP 1, there were some issues to fix and some additional functionality to add. With this version, there are many improvements including more surface-mount components and a smaller design. For a full description of the board, see Appendix C.

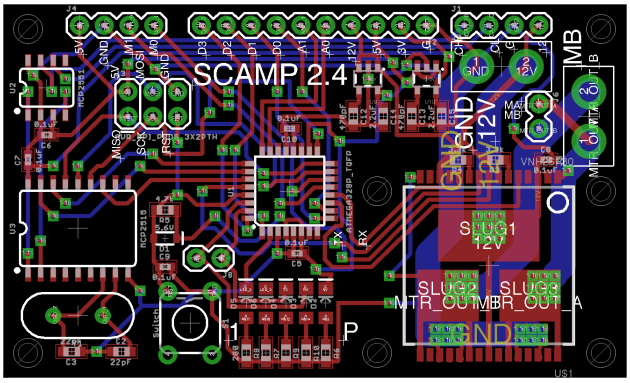


Figure 3.7: SCAMP 2 Schematic

The second version is 53% smaller than the first version. This allowed us to make the housing much smaller while making the heat sink even bigger with an additional fin.

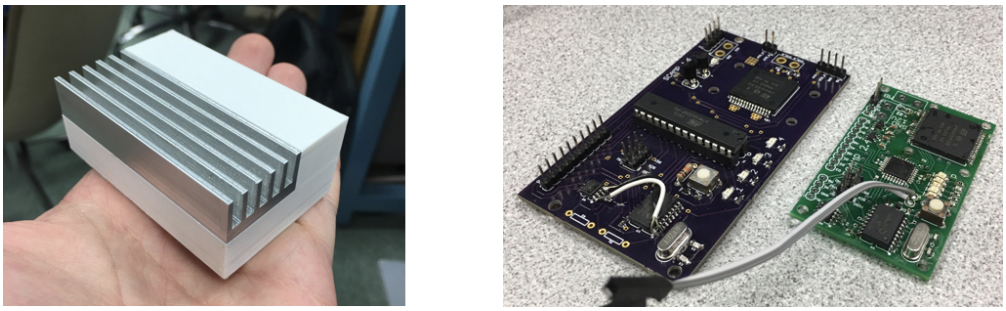


Figure 3.8: SCAMP 2 Housing and Heat-Sink, SCAMP Size Comparison

3.5 Controls Testing Methodology

In order to test the actual autonomy of the boat, we went out to the water and progressively tested the various levels of boat electrical and software functionality:

1. Testing the boat under radio control, to ensure that the electrical system worked and that the boat could physically sail.
2. Running the boat by giving it basic heading commands, and then confirming whether or not the rudder and winch behaved appropriately.
3. Giving the boat GPS waypoints to go to to try to determine whether or not the tacking code was behaving properly.

All the while, we also tested on each stage with both the rigid wing sail and the more traditional windsurfer sail.

4. Mechanical Design, Analysis, and Manufacturing

In order to have a platform to test all of the software, controls, and electronics described in section 3, a 2 meter boat had to be designed and manufactured. The following sections will describe all of the major systems of the boat and all of the design, analysis, and manufacturing work that went into the boat mechanically. Refer to Figure 4.1 and Appendix D for complete pictures of the boat.

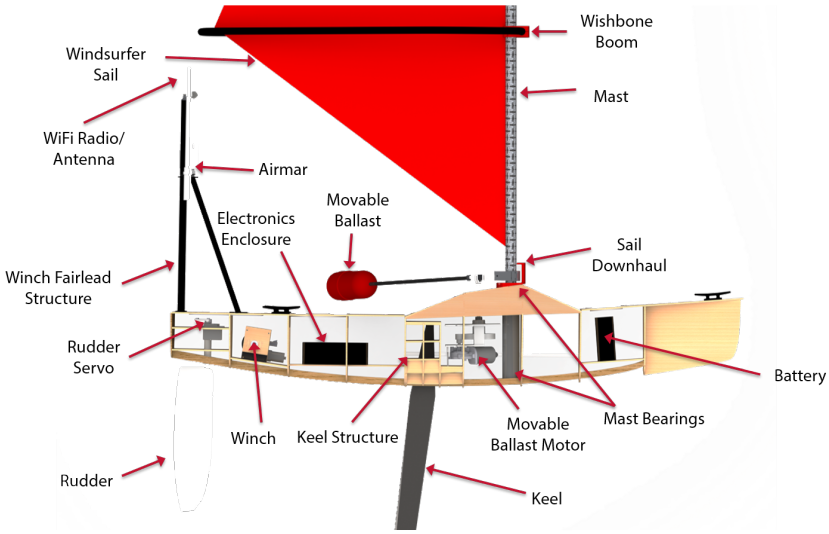


Figure 4.1: Detailed View of Full Boat

4.1 Hull Design

The first major component of the boat is the hull. Based on the advice from previous sailbot teams and especially the USNA, a monohull design was decided on early in the project. There was concern about the ability to turn and tack with a multi-hull system and it was believed that the decrease in stability would be a reasonable trade-off for increased maneuverability, transportation, and ease of manufacturing. As for the design of the hull, a donated hull form was used due to its availability and quality. Since none of the members of the MQP team are Mechanical Engineering majors and the donated hull is of already proven design, we decided to use the existing hull form. The hull form produced a hull of exactly 2 meters in length with a very narrow beam for minimal drag. Figures 4.2 and 4.3 show a Solidworks model of the hull shape that was used.

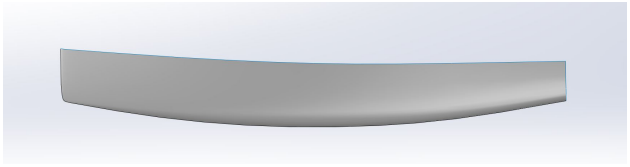


Figure 4.2: Side View of the Hull Surface Shape

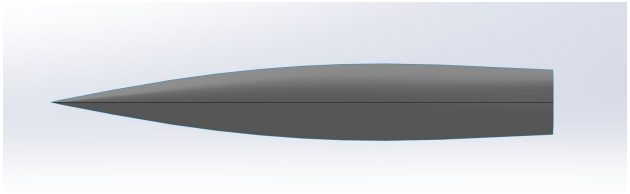


Figure 4.3: Top View of the Hull Surface Shape

4.1.1 Hull Fiberglassing

In order to make the hull from the donated hull form, a conventional layered fiberglassing technique was taken. With the help from members of the WPI Robotics Club, the hull was laid up using layers of fiberglass cloth added one after the other. At a minimum, every part of the boat has 4 layers of fiberglass, but some areas such as the bow and transom have many more for added rigidity. Before beginning to lay-up the hull there was a sizable amount of fiberglassing practice which is described more in depth in section 4.6.

4.1.2 Internal Ribbing Structure

Since the hull was merely constructed of layered fiberglass with no filler agent to provide rigidity, an internal ribbing structure was designed, produced, and fiberglassed into the boat to provide rigidity as well as attach the deck onto the boat. This ribbing structure features a series of lasercut plywood bulkheads located approximately 8" apart throughout the length of the boat. Within each of the bulkhead sections, areas were dedicated to each of the major boat support structures and system. For example, there was a section dedicated to the mast tube, keel structure, rudder, electronics, battery, ect. See the following sections for a detailed explanation on each of these systems. Figures 4.4 and 4.5 show the final ribbing structure.

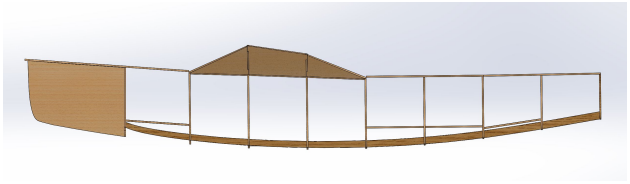


Figure 4.4: CAD View of Bulkhead Support Structure

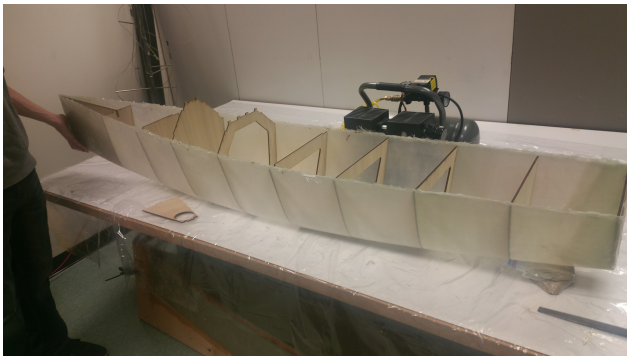


Figure 4.5: Ribbing Structure Dry-Fitted into the Boat

4.1.3 Deck

Going hand in hand with the ribbing structure, the deck was designed and made to provide rigidity to the boat. The deck features a series of lasercut plywood pieces held together with approximately two layers of fiberglass on each side. In addition, the deck includes a stepped area near where the mast is located in order to better support the possible moments produced by the sail. For example, below are some calculations to show the difference in thrust loads that would need to be supported with a stepped deck area vs. a non-stepped deck.

Calculated max righting moment based on keel righting moment at 90 deg heel:

$$M_{max} = 52in * 30lbs \qquad = 1560in * lbs$$

Calculated thrust loads with and without a deck step

$$F_{Non-Stepped} = 1560in * lbs/9.5in \qquad = 164.2lbs$$
$$F_{Stepped} = 1560in * lbs/14in \qquad = 111.4lbs$$

As the calculations show, without the little step in the deck, the thrust loads on the supports increase by almost 50%.

Finally, the deck was attached to the hull using layers of fiberglass and fiberglass filler mix. First the deck was dry-fit on to the boat and sanded to make the it fit well with minimal gaps. Then, the gaps between the deck plates and the hull were filled with a mixture of resin and low density filler. Lastly, fiberglass was laid over the top of the filler in order to provide a watertight seal between the hull and the deck and to better smooth out the attachment between the deck and the hull.

Figure 4.6: Completed Deck Temporarily Fit on the Boat

4.1.4 Access Hatches and Deck Accessories

Given that this boat was made as a prototype system which required constant customization, a multitude of of access hatches were added into the deck to provide access to the major actuators, electronics, and battery, and to allow for water leakage monitoring. In total, there were 4 hatches added to the boat all with a distinct purposes as shown below.

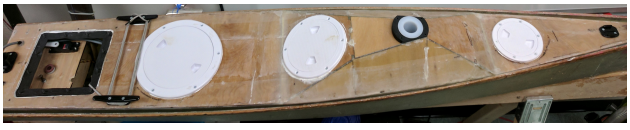


Figure 4.7: Access Hatches Installed on the Boat

Additionally, as seen in the figure, mainly COTS (commercial off the shelf) hatches were used for their ease of installation and proven effectiveness. However, in the case of the aft most hatch, a custom hatch needed to be design and manufactured due to the lack of appropriately sized hatches that were available. This hatch was made from a series of lexan plates and rubber sealant in order to seal against the deck prevent water from entering. The lexan plates were chosen to make it easier to view systems inside the boat.

4.1.5 Bilge System

Since every boat leaks to some degree, it was decided that having a bilge pump would be helpful in situations such as the endurance challenge where the boat must be on the water for 6-8 hours. However, we did not expect a very large amount of water accumulation in the boat so we went with one of the smallest water pumps available. The chosen pump is rated for continuous duty cycle operation allowing the boat to function fine in longer events such as the endurance challenge. The specific pump that we decided on is a 12V DC 240 L/Hr submersible pump. This pump fits all of the criteria listed below. The pump can be seen in figure 4.8

- A constant duty cycle
- Use in freshwater and saltwater environments
- Small form factor
- 12V DC power source

Figure 4.8: Water Pump used for the Bilge System

For controlling the pump, there are two methods of sensing water/moisture that were investigated. The first was a series of float switches placed throughout the boat in order to get an approximation on the depth of the water. However, with testing we found that there was very little water accumulating in the boat which was not sufficient to trigger a float switch. As a result, we instead explored a moisture sensor, which allowed for easier placement and quicker water detection for more efficient use of the bilge pump system.

4.2 Keel Righting Ballast

The main form of righting ballast on the boat is in the form of lead ballast located on the bottom of the keel. This ballast serves as a method of opposing the component of lift from the sail which is perpendicular to the hull. If the lift is not counter balanced, the boat can heal excessively and possibly capsize.

4.2.1 Keel and Ballast

The keel selected for the boat is a keel that was donated by the USNA Sailbot team from last year's competition. This was selected for it's proven success with the hull design. Again since this project was mainly a focus of the controls of the sailboat and there was limited mechanical resources, using an existing keel allowed for much better focus on other portions of the boat. The donated keel is approximately 5 feet long with a 27 lbs lead shot bulb on the bottom.

4.2.2 Keel Structure

Due to the tremendous righting moments that the keel is able to provide, a structure was designed to fit and attach the keel to the boat as well as distribute the forces throughout the fiberglass hull. This structure

was made mainly of lasercut 1/4" plywood sections with patches of fiberglass in order to add rigidity to the structure and attach the pieces to the hull. The structure features a main fiberglass pocket for the keel and then a series of plates which distribute the loads from the pocket onto the hull. When creating this fiberglassed pocket, there was some difficulty in ensuring the strength and watertight sealing of the connection between the pocket and the boat. A first attempt proved unsuccessful as it had a major leak and did not seem fit for supporting the keel moments, however, a second attempt proved much more successful. This was accomplished by 3D-printing a model of the top portion of the keel and then fiberglassing the pocket directly inside of the boat. Shown in figure 4.9 is a picture of this 3D-printed tool used for fiberglassing in the pocket.

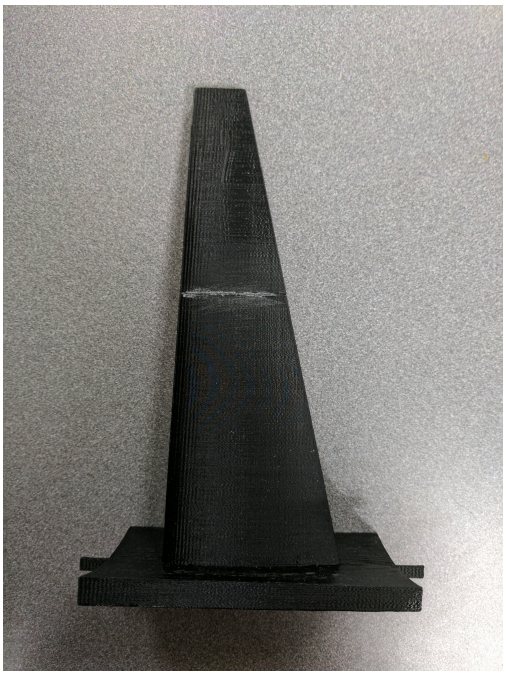


Figure 4.9: 3D-printed Keel Mold used to Fiberglass Structure Pocket

This allowed for a much better bond between the two fiberglass structures since layers of fiberglass where able to be connected to the hull both on the inside and outside. Finally, at the top of the structure, a plate with a hole was added in order to rigidly support the weight of the keel by the single 1/4-20 tapped hole in top of the keel. Below in Figure 4.10 is a comparison view of the CAD model of the keel structure and the actual structure on the boat.

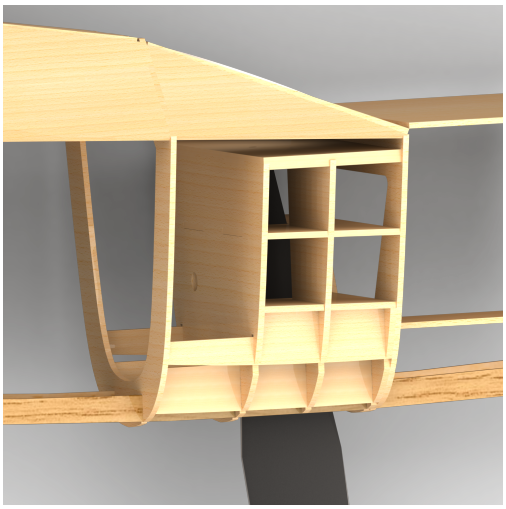


Figure 4.10: Keel Structure CAD vs. Actual Comparison View

4.2.3 Keel/Centerboard Improvements

As mentioned later on in the testing section, initial testing showed that the boat was not very balanced in one of the sail configurations due to the center of effort and center of lateral resistance being located too close to one another. As a result, an extension of the keel was designed and implemented which moves the center of lateral resistance aft and also allows the keel to generate lift at lower speeds. Figure 4.11 shows this extension which was added to the keel.

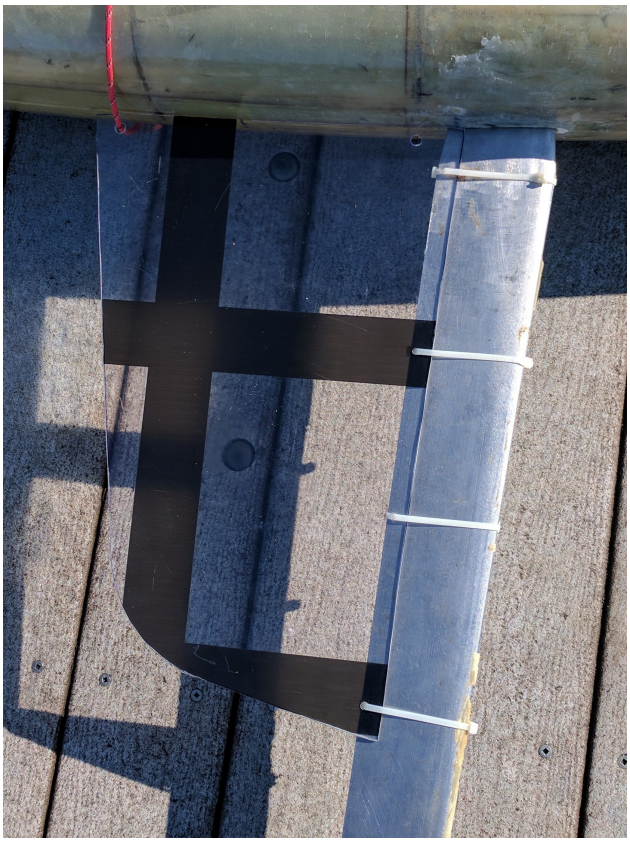


Figure 4.11: Prototype Keel Extension Piece

4.3 Sail Mechanical Systems

There were two main sails that were chosen for use on the boat. The first sail was a self-trimming Rigid Wing Sail as mentioned in section 2.5. The second is a back-up sail system which features a winch-trimmed windsurfer sail. These two sails were designed together to have a similar interface with the boat for ease of switching between the two sails.

4.3.1 Mast Interface and Structure

The first part to each of the sails is how they are attached and supported on the boat. This is done by a small structure and a mast tube. The mast tube is a piece of 2" SCH 40 PVC pipe. This tube is fiberglassed in the boat at the bottom and the top in order to adequately support the loads that are experienced in each of those locations. Within this tube is a series of thrust bearings that allow each of the sails to freely rotate while supporting the sail moments at each of the connections. Additionally, there is a vertical retention system that ensures each of the sail configurations will not come out of the boat in case of capsizing.

4.3.2 Rigid Wing Sail

The rigid wing sail was a sail design by a second MQP team to be the main propulsion method for the boat. This sail drove most of the decisions for how the boat would mechanically interface with each of the two sails. In order to ensure that the separately designed boat and sail would interface properly, a document

was created to explain the mechanical interfacing between the boat and the rigid wing sail which can be found in Appendix B. This document not only describes the mechanical interfacing structures as summarized in section 4.3.1, but also the space, weight and size allocations given to each project to ensure that the final product would function as designed and fit within the rules of the sailbot competition (section 2.2).

4.3.3 Windsurfer Sail

Since the main propulsion system of the boat was a parallel developed MQP project, a back-up sail system was designed and made in order to ensure that a method of propulsion would still be available regardless of the progress or success of the other MQP. The type of sail that was decide on is a windsurfer sail for its simplicity and similarity in interfacing with the Rigid-wing sail design. This sail required a few additional systems in order to function effectively.

Sail Rigging

One of the main advantages of the windsurfer sail over a conventional stayed sail is that it requires minimal rigging. However, since this sail is not conventional for a sailboat like this, there were a few modifications that had to be made in order to ensure that the sail could be rigged-up effectively. Figure 4.12 shows all of the parts to rigging up the windsurfer sail.

Figure 4.12: Rigging of the Windsurfer Sail

One of the main components to the rigging that had to be designed and manufactured was a downhaul block which provides tension on the sail and generates the necessary bend to the mast. Throughout the

design of this component, it was unknown exactly how much downhaul force would be required to get the appropriate rigging, but it was known that it would likely be on the order of 200 - 300 lbs. The general design of this component is a two part assembly that clamps onto the exterior of the windsurfer mast which has a series of pulleys and a jam cleat in order to tension the downhaul cable. Figure 4.13 shows a picture of this downhaul block that was made.

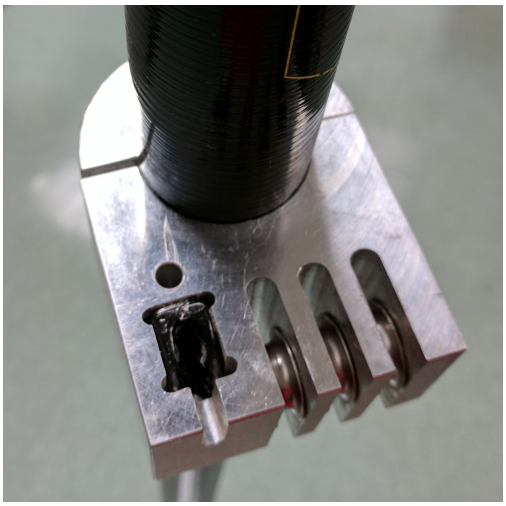


Figure 4.13: Downhaul Block used in the Rigging of the Sail

The rest of the parts to the windsurfer sail are conventional windsurfer parts including the cloth sail, wishbone boom, and RDM mast. Finally, the windsurfer sail is actuated based upon a main sheet line which is run to the wishbone boom.

Winch

In order to control the main sheet, a custom winch was designed and manufactured. The first part to the winch design was motor selection and calculations. Below are the calculations for the max power that would be required from the winch.

$$\begin{aligned} P_{max} &= F_{max} * V \\ F_{max} &= 40lbs \\ V &= 60"/10sec \\ P_{max} &= 27watts \end{aligned}$$

There were many initial designs including different motors for the winch, but in the end a window motor was decided on for its anti-backdrive characteristics, water-resistant sealed design, and adequate output power characteristics. After selecting a motor, a drum size had to be calculated in order to provide sufficient output toque of the winch in worst case situations. Calculations for the winch drum size can be found below:

$$\begin{aligned} \tau_{PeakPower} &= 4.3N * m \\ \omega_{PeakPower} &= 42RPM \\ R_{drum} &= \tau_{PeakPower}/F_{max} \\ R_{drum} &= 0.95" \end{aligned}$$

With the drum size decided on, the rest of the winch system and it's mounting to the boat was designed. This included mounting for the motor, mounting for the position-control potentiometer, and management of the main sheet in the form of a winch cover/ divider system. Figure 4.14 shows a picture of the complete winch system in its final configuration.

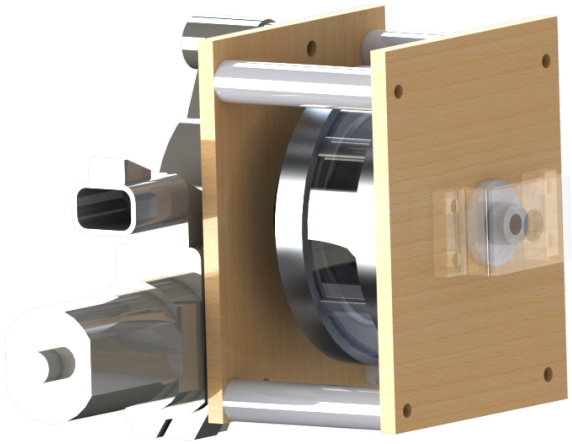


Figure 4.14: CAD Render of the Winch

Winch Structure

When designing the winch system for the boat, there was a concern of having adequate torque to trim the sail in close-haul situations. When running the main sheet cable to a wishbone boom in this close-haul situation, there is a very severe angle that the boom is pulled at which requires significantly more torque then normal operation. As a result, a structure for managing the main sheet and allowing for a more optimal sheet tension angle was designed. In order to adequately design the structure and ensure that it would hold up in the worst case situations, a Solidworks FEA was run on the structure to determine proper materials and sizings. Below in Figure 4.15 is one of many FEA tests that was run on the structure design.

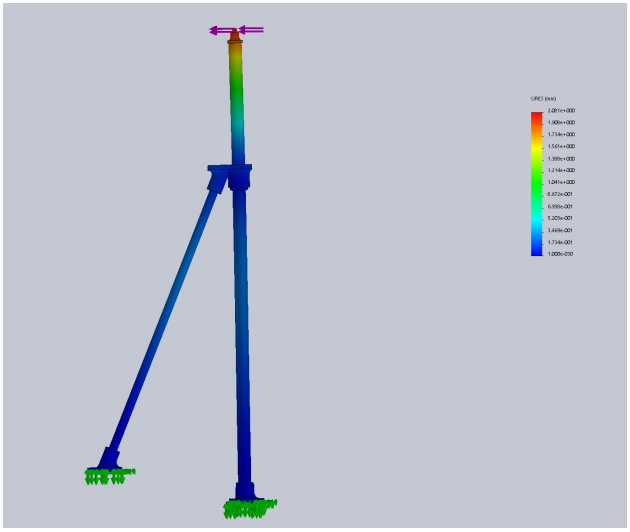


Figure 4.15: Finite Element Analysis of Winch Structure System

In addition to managing and supporting the main sheet cable, this structure provides a good place to put some of the boat's electronics and sensors (specifically the Airmar and the WiFi radio and antenna). First off, the Airmar was mounted here because it provides a fairly unobstructed place for measuring the wind direction and getting a GPS fix. In addition, since it is often thought that the most important place to have accurate wind sensor data is when traveling downwind, the sensor is placed on the aft of the boat to ensure unobstructed wind direction data in a downwind configuration. Secondly, this is also an optimal location for the WiFi antenna since the WiFi range is improved when there is a direct line of sight to the antenna.

Figure 4.16: Final Main Sheet Structure Design with Sensor Mounting

Overall, this structure consisted of a 4 legged design with a series of custom made fairleads and a single main sheet pulley for routing the main sheet to the wishbone boom at a more optimal tension angle in a close-haul situation. Figure 4.16 shows the final structure.

4.4 Rudder Actuation

The second main point of actuation on the boat is the rudder. This system is critical to the boats functionality and reliability because without control over the rudder the boat has little to no control over its heading. As a result, it was important to ensure that the rudder system was thoroughly tested and proven robust.

4.4.1 System Requirements and Calculations

For the rudder system, like many of the other boat systems, existing parts were used to decrease the number of new parts and components that needed to be designed. Just like the keel and the hull form, the USNA had donated a rudder last year to the project which was used as the rudder for this system. This rudder is shown in Figure 4.17.

Figure 4.17: Rudder Donated by the USNA and Used on the Boat

The main portion of design and mechanical implementation work that surrounded the rudder was finding an appropriate servo and creating a robust method of connecting the servo to the rudder. Since calculating the necessary torque to adequately turn the sailboat is not a very straightforward calculation, we instead calculated the torques that would be required to move the rudder to different positions at varying speeds and then searched for an appropriately powered servo from there. Below in table 4.1 is a list of some of the calculated torques.

Table 4.1: Rudder torque requirements at different boat speeds and rudder angles

Rudder Angle Range (degrees)	**Boat Speed (knots)**	**Torque Requirement (oz-in)**
± 35	1	51.9
± 25	1	35.4
± 25	3	318.7
± 15	3	197.0
± 15	5	547.2

As a result, a high-torque waterproof servo was selected in order to control the rudder.

4.4.2 Design and Improvements

In order to obtain the maximum reliability that was desired out of the rudder system, the method of actuation was kept very simple with an adjustable linkage system. This linkage is shown below in figure 4.18

Figure 4.18: Final Rudder Servo and Control Linkage

Additionally, to add to the robustness of the system, there were some improvements that were made throughout the testing process. The first change, and most substantial one, was swapping out the driving servo due to what is believed to be a lack of torque from the original one. Another change that was made to the servo system was experimentation with the linkage lengths to increase control over the rudder. Since the rudder only requires a maximum of 45 degrees of motion to either side, the linkage was altered to offer more torque to the rudder at the expense of less range of motion.

4.5 Movable Ballast System

A secondary form of righting ballast was intended to be used on the boat in the form of a electronically controlled movable ballast system. The purpose of the movable ballast system was to reduce the weight needed to act as the righting moment of the boat. Because the position of the movable ballast system could be actively controlled, and because the arm could be kept at or near horizontal with regards to the force of gravity, the weight on the movable ballast system would be used more efficiently than that on the keel. The plan was to install the movable ballast system on the boat, determine the most optimal weight for the system to use, and reduce the ballast on the keel accordingly.

4.5.1 Design

The first step of designing the system was determining the method by which the ballast would be moved within the boat. As mentioned, a previous MQP had been performed on this very topic, the Sailboat Stabilization System [8]. It was determined that the method used would not be optimal for the purpose of this MQP for two reasons, the mechanism would likely be too heavy, and it would not be able to provide the amount of leverage needed. Instead it was determined that the design was to be a composed of a weight mounted on a rotating arm located above the deck. The arm would be horizontal to the deck when pointed directly aft and would decline in a linear fashion as it rotated in either the port or starboard direction to a maximum declination of 35°. The arm would rotate a maximum of 180°with the arm pointing directly aft as its center. The arm's declination would be controlled by a ramp that it rested upon through its rotational travel. The position of the arm would be measured using a rotary potentiometer.

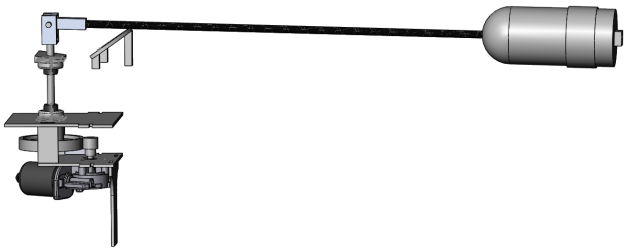


Figure 4.19: CAD of Movable Ballast System

4.5.2 System Requirements and Calculations

Determining the maximum righting torque that system would need to be able to deliver was the initial step of its design. This torque would determine the power requirements of the motor and drive train to move the arm. It would also determine the maximum amount of force that the arm, ramp, deck, and overall system would need to be able to handle. The torque was calculated with a few assumptions: that the heel of the boat should be designed not to exceed 35°, that the maximum torque that the movable ballast system would ever possibly need to output was equal to the righting moment of the keel supplied by the USNA Sailbot team at 35°, and that the arm of the system would be roughly 1 meter in length.

To calculate the maximum power of the motor to drive the arm, it was estimated that it should be able to traverse its 180°movement in as little as 5 seconds (6 rpm). Other factors to consider for choice of the motor were water resistance, nominal torque, nominal rpm, inability to be backdriven, required voltage, weight, and cost. It was preferable that the motor's nominal rpm be quite low to reduce the amount of gearing that would need to be used to result in a reasonable speed (˜6 rpm). Another important feature is the inability to be backdriven as this would greatly reduce the energy requirement for the system. The chosen motor was an AME 218 series windshield wiper motor. This motor features a built in gearbox with a screw drive so that it cannot be backdriven, is resistant to water, runs at 12 volts, and has a no load speed of only 116 rpm. Given the nominal torque of 98.235 in*lbs, a gear ratio of around 5:1 was required. There were a few drawbacks to this motor however: its strange, asymmetrical shape, the unusually sized 11 mm diameter output shaft, and its low efficiency.

Gears were chosen to reduce the output speed and increase the output torque of the motor. A single-stage gear system was preferred because of simplicity. Using the estimated power rating of the system, it was determined that 16 pitch, 20°pressure angle, 0.75 inch face width gears would suffice. A 15 tooth gear and 72 tooth gear form the drivetrain with the 15 tooth gear acting as the motor pinion. While the power rating of the 15 tooth gear was just barely more than needed, the 72 tooth gear was far stronger, and heavier, than necessary. Using finite element analysis measures were taken to determine how much material could be removed from the 72 tooth gear while remaining strong enough. Machining was performed on the gear to remove 60% of the weight but reducing the safety factor could allow more weight to be removed.

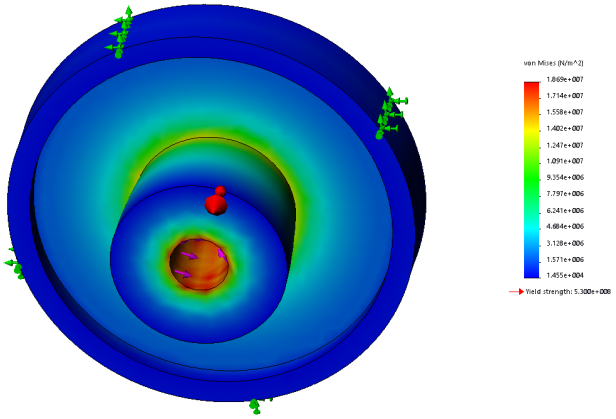


Figure 4.20: FEA of 72 Tooth Gear

Given the maximum expected righting torque to be produced and a maximum length of 1 meters, finite element analysis was done to determine the material and size that should be used. Of particular consideration were also the characteristics of corrosion resistance and stiffness. Corrosion resistance was important given the arm's location above the deck of the boat and expected exposure of saltwater. Because of the arm's long length, and the location of the sail just above it, the arm needed to be particularly stiff to avoid interference.

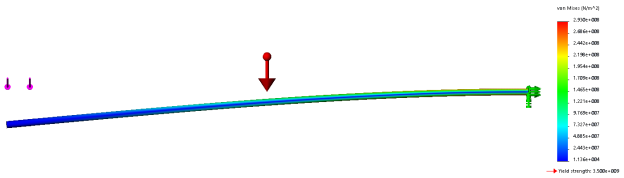


Figure 4.21: FEA of 1 meter Carbon Rod

Ultimately, the final design for the arm was a 0.5 inch diameter, unidirectional, extruded, solid carbon fiber rod. The reason for the use of a solid rod was due to the initial plan of attaching the rod to other parts using screws or pins. However, when it became clear that no convenient machining facilities would allow machining carbon fiber the plan was scrapped for the use of Scotch Weld DP460, an epoxy, instead.

4.6 Manufacturing Techniques and Processes

Throughout the project there was a variety of manufacturing techniques that were used in order to produce the various boat systems that were described in the previous sections. The following sections will go into detail as to how these techniques were used throughout the project.

4.6.1 Fiberglassing

The main manufacturing technique that was used during this project was fiberglassing due to its common use in the marine environment for hull, deck, etc. manufacturing. Since this technique was new to all of the project members, there was a sharp learning curve that required practice as part of the project. We started by practicing using many female mold shapes before working with many of the real boat components. These molds included pizza pans, baking pans, and some of the more complex parts of the boat hull such as the transom and the bow. This practice took many iterations to improve our skills to a point of actually producing real boat components. During the practice, we experimented with the number of layers of fiberglass cloth to

use, the amount of resin/hardener to use, the best methods for applying and spreading the resin/hardener, and the best methods for eliminating air bubbles.

As mentioned in previous sections, fiberglass was used in many places throughout the boat including the hull, deck reinforcing and waterproofing, and structure reinforcing. The fiberglass allowed us to make strong and light structures molded into almost any shape in a relatively easy manner. All of the fiberglass was done using West Systems hardener and resin and 6 oz fiberglass cloth.

In addition, for many of the structural components and the fiberglass bonding that was performed, we used a variety of fiberglass fillers. Some of the fillers that we experimented with West Systems 406 adhesive filler and 404 high density filler. These fillers provided a method for bonding fiberglass components, filling air gaps for a waterproof seal, and filleting connection joints for a stronger seal.

4.6.2 Laser Cutting

The next form of manufacturing parts that we used for the project was laser cutting of plywood. This technique was used for most of the interior boat structure for its ease of use and quick part turnaround. Using this manufacturing technique we were able to make wood parts of specific shapes and size very easily that would conform to the inside of the boat.

4.6.3 Manual and CNC Machining

The final main method for manufacturing that we used was manual and CNC machining for metal and plastic parts. Some of these parts included the downhaul block, custom hatch cover, fairleads, etc.

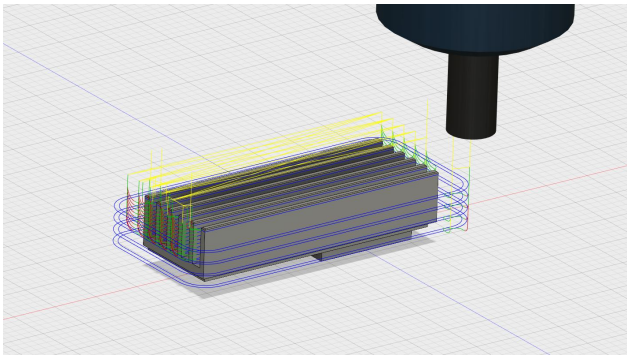


Figure 4.22: CNC Machining Computer-Aided Manufacturing Program

4.7 Testing Methodology

Progressive testing was performed to ensure each of the systems would function effectively when the boat was completed and ready for full system testing. Some of these tests include waterproofing tests, keel structure tests, and overall boat durability tests.

4.7.1 Waterproof Testing

In order to ensure that the fiberglassing that was done at each stage of the project was well done and formed an adequate water tight seal, there was some waterproofing testing done throughout the build process. After the hull fiberglassing, the keel structure fiberglassing, and finally after attaching the deck to the boat, we performed specific testing to find possible leaks and repair them as necessary.

Figure 4.23: Hull Leak Test Showing a Small Leak

4.7.2 Mechanical Durability and Reliability Testing

There were two main methods of testing many of the mechanical systems on the boat. First, there were targeted tests to ensure that each of the boat structures and systems functional well on there own. For example, after fiberglassing the keel structure into the boat to its full capacity, there was specific testing to ensure the structure held up and could support the keel at excessive angles of heel. Figure 4.24 shows testing that was performed to verify the keel structure.

Figure 4.24: Keel Structure Testing Showing the Boat Heeled

Secondly, there was reliability testing which solely included giving the boat time to sail on the water. This testing would often run in parallel with controls and autonomous testing. Throughout this testing, we were not only able to ensure that each of the systems work like they were supposed to, but also that they worked reliably and consistently.

5. Results

Overall the boat performed well, sailing from way-point to way-point with minimal human adjustment required. The boat was able to adjust the rudder and winch to set the heading and trim appropriately.

5.1 Boat Sailing Performance

In testing to ensure that the boat was actually able to sail (regardless of whether it was running autonomously or under radio control), we observed several things.

5.1.1 Original Keel & Windsurfer Sail

In general the boat performs reasonably well with the original keel and sail. Once the boat accumulates some speed, it is controllable and able to switch tacks and headings without a problem. At higher speeds though, the boat has a considerable amount of weather-helm (a tendency to turn upwind). This is due to the placement and size of the keel relative to the sail. The keel is too far forward with respect to the sail and too narrow to generate enough lift to compensate for the sail. This leads to excessive heeling and weather-helm.

5.1.2 Jib & Keel Extension

With the addition of a jib and a keel extension, the boat's performance improved slightly. The jib moves the center of effort of the sails forward and reduces weather-helm, while the keel extension both moves the center of effort of the keel back while also reducing the leeway (sideways movement of the boat) at low speeds. Although there is more drag with the extended keel, it makes the boat more controllable at low speeds.

5.1.3 Windsurfer Sail vs Rigid Wing

When testing we compared the traditional style windsurfer sail to the rigid wing. In general, the windsurfer sail works better at lower wind speeds. This is related to the problem described above with weather-helm. At high wind speeds, the sail forces the boat to point upwind as well as introducing too much heel angle due to the sheer area of the sail. The weather-helm makes it hard to control the heading of the boat while the heel exacerbates the weather-helm and reduces the effective authority of the rudder by angling it upwards. At lower speeds, the rudder has enough authority to change the heading and start generating lift.

The rigid wing had the opposite problem as the windsurfer sail. In our testing, we almost exclusively used just the lower half of the rigid wing. This is because the upper half was too heavy and caused the boat to tip to one side even with little to no wind. Since the area of the wing was almost always reduced, the lift was also reduced. At wind speeds below 10 knots, the boat struggled to accumulate any speed. Above 10 knots however, the boat performed well. Since the rigid wing is 100% freely rotating, it "windvanes" to the proper position. It is also much more efficient than the traditional sail. On the other hand, when the boat is traveling at low speeds and the rigid wing sail is being used, when the boat turns downwind the center of effort of the sail moves forward and increases the boat's lee-helm (tendency to turn downwind). When combined with the low-speed of the boat (which reduces rudder authority) and the relatively low drag of the rigid wing (which is advantageous when going upwind but counterproductive when going downwind), this means that the boat has trouble controlling its heading when going downwind with the rigid wing in low wind conditions.

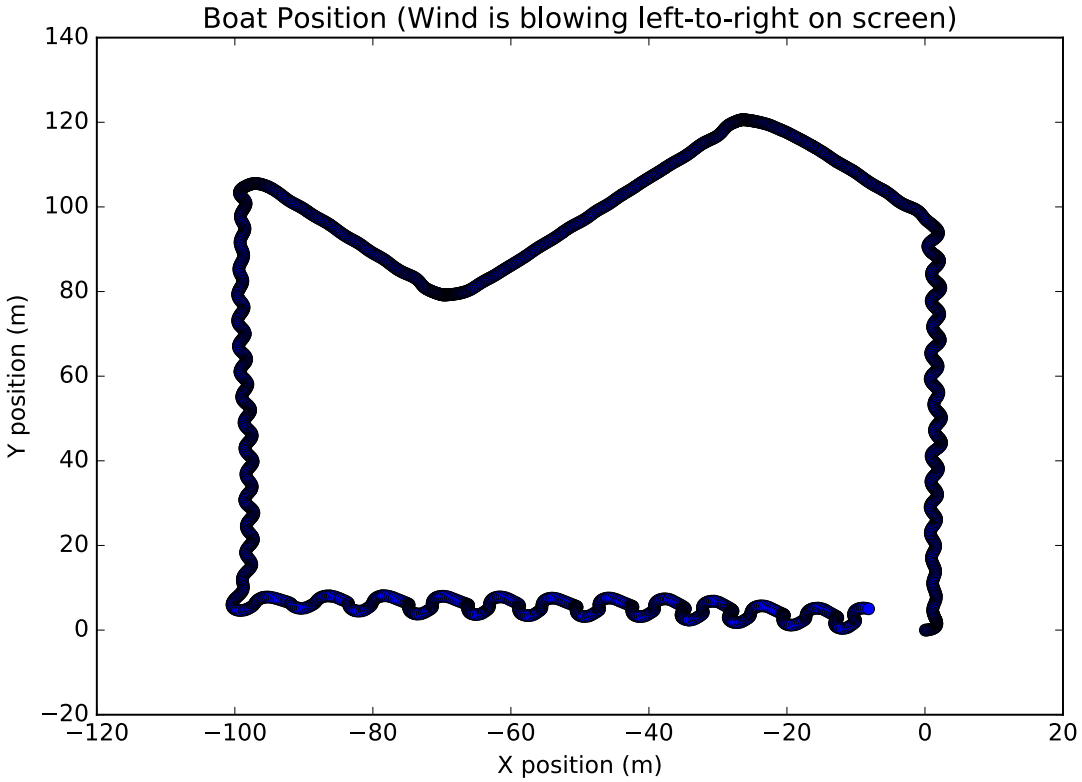


Figure 5.1: Boat Traversing a Square in Simulation

5.2 Controls

5.2.1 Simulation Results

Broadly speaking, the simulator functioned and allowed us to do the initial development of the various control algorithms.

In Figure 5.1 we can see the boat going around a square in simulation, validating that the basic algorithms do fundamentally work. However, there are several issues with this:

- The simulation generally assumed ideal wind conditions, and we did not try out sufficiently varied wind conditions in the simulator. This was compounded by the fact that we did not know what sorts of wind conditions to reasonably expect in real life and so were unable to easily try them out.
- Even a perfect simulator would still need to have the parameters tuned to reflect reality; we did not get a chance to do this beyond basic sanity checks.

To illustrate the final point, we can point to the downwind and reaching legs of Figure 5.1 (the bottom, left, and right legs). On all of these legs the boat experienced noticeable oscillations. In the real world, these oscillations did not occur. However, in the simulation, this likely occurred due to poorly tuned damping parameters or incorrect assumptions about the rudder hydrodynamics, the boat oscillates.

Unfortunately, this made it hard to reliably test algorithms in the simulation, as we could not be sure whether issues were due to real, fundamental issues with the algorithms or inaccuracies in the simulator.

5.2.2 Real Life Results

General note on graphs in this section: Many of the graphs contain some small text; however, they are all included in vectorized formats and zooming in on the graphs should make everything readable.

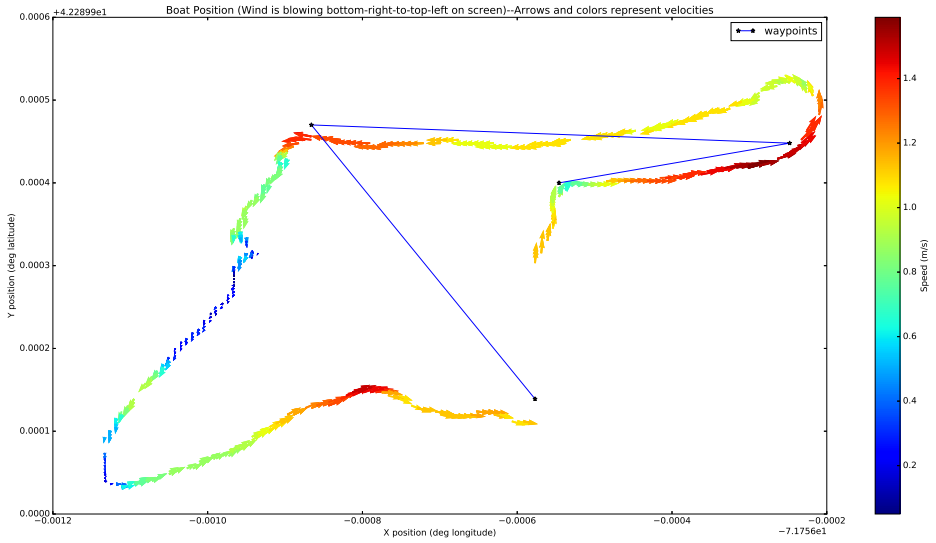


Figure 5.2: Boat Following a Triangular Course

In Figure 5.2, we see the boat going to 4 waypoints, starting with a downwind leg, then a pair of beam reaches and an upwind leg. Although it is not made clear in the graph, it should be noted that while the boat was going upwind we attempted to change the waypoints a couple of times to try and get it out of irons. The boat did operate wholly autonomous for the first three waypoints and we never manually specified anything below the level of the waypoints.

The primary takeaway from just Figure 5.2 is that the boat can:

- sail on all points of sail, including tacking and gybing
- autonomously sail between waypoints on all points of sail
- autonomously gybe and change heading, but struggles to tack reliably

Figures 5.3, 5.4, 5.5, and E.1 correspond to the same course as shown in Figure 5.2, with the X axes being time and the various lines corresponding to relevant boat information. Figures 5.3, 5.4, and 5.5 are all zoomed-in portions of Figure E.1, as Figure E.1 is somewhat difficult to parse without zooming in and so is only included in the appendix.

Some notes on the various lines presented in the graphs:

Heading The direction that the boat is currently *pointed*–not necessarily the actual direction of travel if the boat is moving sideways.

Apparent Wind The angle of the apparent wind on the boat. If zero, the boat is travelling upwind; if positive the boat is traveling on a port tack with the wind coming over its port side.

Rudder The current rudder command to the servo–we do do not have direction feedback from the servo so we simply have to assume that it is doing the correct thing. 0 angle corresponds to the rudder being straight, and positive values would cause the boat to turn to starboard.

Heel The angle at which the boat is heeled over, with 0 being no heel and positive heel meaning that the top of the sail is over the starboard side of the boat.

Leeway The difference between the direction of the velocity vector for the boat and the heading of the boat. If zero, that means that the boat is traveling in the direction that it is pointed. Positive values mean that the boat is drifting slightly to the starboard.

Boat Speed The magnitude of the velocity vector of the boat.

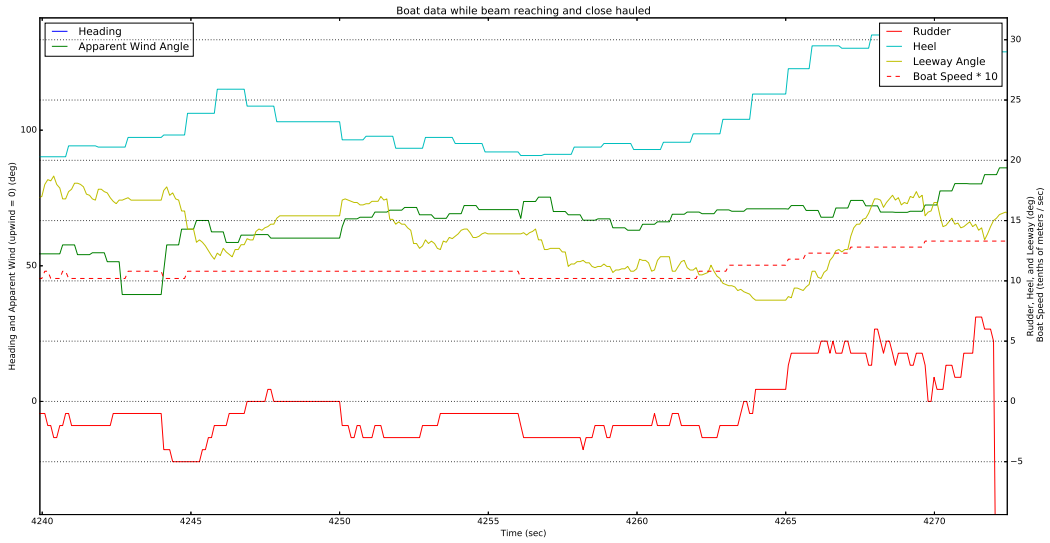


Figure 5.3: Beam Reach Portion of Figure E.1

In Figure 5.3 we see the boat's behavior along the main beam reach (the top segment of Figure 5.2). The primary notes of interest are that:

- The heel angle is a manageable 20-30 degrees, about what we originally wanted.
- It requires some constant rudder offset to go straight but it isn't unduly much (the boat is going straight for the first around 3/4 of the segment and the rudder stays between 0 and -10 degrees).
- The leeway angle is a significant but not undue 10-20 degrees.
- The speed of the boat is steady at around 1m/s (2 knots).

Figure 5.4 shows the close hauled portion of the boat's travel (the left-most leg in Figure 5.2). The following observations can be made:

- The boat retains approximately the same heel angle while it was beam reaching
- The rudder (and, by extension, the boat as a whole) oscillates significantly while going close-hauled
- The boat is sailing substantially closer to the wind than one would typically expect, with an apparent wind angle of 20-30 degrees (the boat itself is sailing at about 50 degrees from the true wind; 40-50 degrees would be a typical angle for the apparent wind when going upwind)

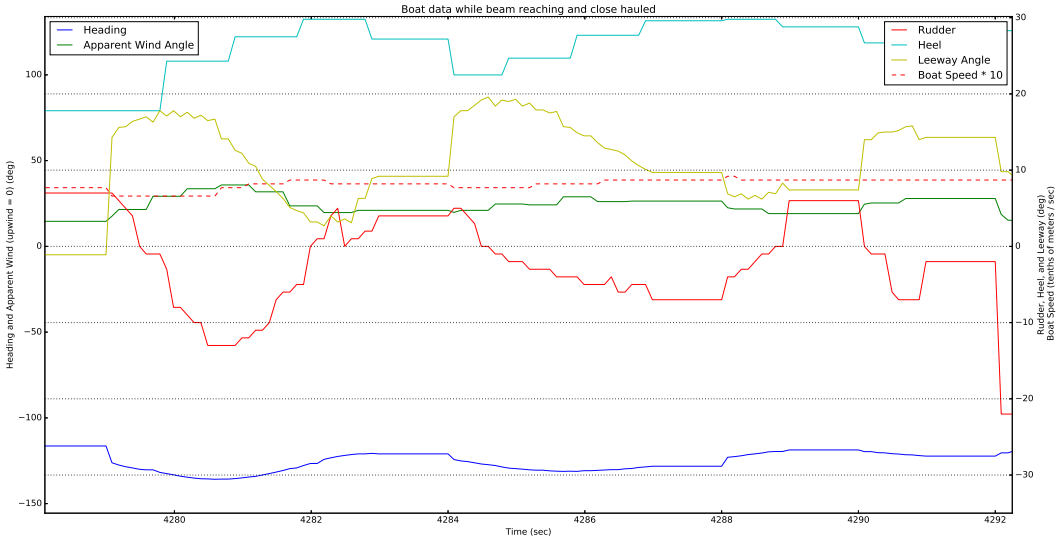


Figure 5.4: Close Hauled Portion of Figure E.1

The final two points seem to be a combination of the weather helm of the boat (which leads the boat to turn upwind, followed by the rudder over-correcting, followed by the boat turning upwind again, and so on and so forth), combine with some poor tuning of the tacking algorithms. In this particular case, the tacking algorithms were driving the boat to sail 45 degrees from the true wind, rather than the apparent wind, meaning that the boat was sailing closer to the wind than would typically be expected. The boat was still controllable, but struggled (as evidenced by the rudder oscillations and the boat's failure to tack for several minutes).

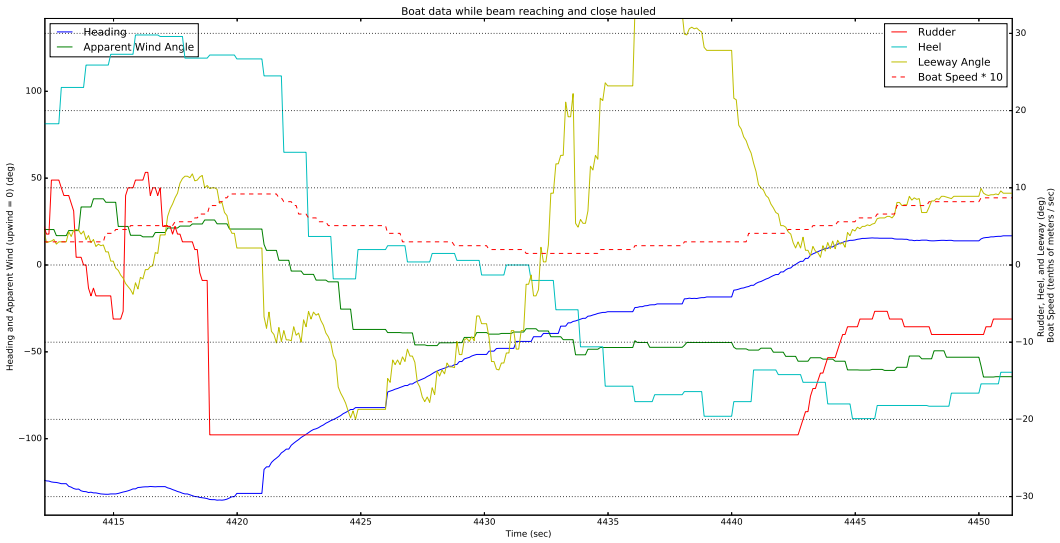


Figure 5.5: Tacking Portion of Figure E.1

Finally, in Figure 5.5 we see the boat's behavior when it is finally able to tack, moving the heading from -120 degrees through the wind to +30 degrees.

- The rudder starts out moving around at low angles, which appears to let the boat start building up speed.
- The rudder then spends most of the rest of the time fully over (it takes the real system a moment to react– the rudder is not moving instantaneously).
- The boat's speed drops drastically while it attempts to complete the tack, although the boat does continue turning without too much issue.

For the most part, the simple algorithms functioned to control the boat. However, in some low-speed conditions the algorithms would tend to maneuver the boat in ways that do not work. On a normal sailboat, the sailor will let the boat build up speed before attempting to maneuver in low-wind conditions. However, in the autonomous conditions, the boat would tend to use the rudder when going too slow (causing the rudder to act as a brake) or attempt to tack and get stuck in irons.

5.3 Overall Autonomy

In general, the boat could successfully navigate between preset waypoints. Under some weather conditions, it struggled more than others, namely with inconsistent and particularly low-speed wind, when the various control algorithms could not always handle the low-speed conditions and keep the boat out of irons.

6. Recommendations and Conclusion

6.1 Recommendations

6.1.1 Finish movable ballast

While all parts of the movable ballast system have been acquired, due to the need to begin testing of the boat as soon as possible, installation of the system has not been completed. Furthermore, due to the need to remove portions of the deck to install the system, and its relative inaccessibility once installed, it would be prudent to first construct and test the system outside of the boat. Also, in order to achieve its purpose of reducing the total weight of the boat, the keel will need to be modified accordingly.

6.1.2 Boat balance

As mentioned in section 5.1, the centers of aerodynamic effort of the various sails and the keel were not aligned as well as would have been desirable to get a well-balanced boat.

In the future, either more care should be put in from the start to ensure that the boat is balanced or some system should be included to make it relatively easy to adjust the positions of the sail(s) and keel so that major modifications are not required to fix any imbalance in the boat.

6.1.3 Controls/Software Recommendations

- Tune simulator to better match reality (see section 5.2.1)
- Develop more robust control algorithms to better handle various corner cases, including low speed and erratic wind conditions
- Thoroughly test all code for real-time performance to ensure that high CPU usage by, for instance, the UI process will not cause issues with the process sending out critical CAN messages
- Consider replacing the WebSocket library with some more lightweight alternative–building the uWebSocket library requires multiple unneeded dependencies (mostly OpenSSL) that are hard to build and require inconvenient workarounds to make compile properly
- Increase unit test coverage, to ensure greater code reliability and preemptively catch bugs
- Further develop log replay infrastructure to make it easy to examine particular spots in the logs, including features such as pausing, rewind, trimming of the logs, and utilities to go to particular times based on the GPS time rather than system time

6.2 Conclusion

We successfully built and programmed a boat to autonomously complete most of the challenges specified in the IRSC. We did not fully complete and test the movable ballast system that we designed. The rigid wing sail was successfully incorporated into the boat and was easy to swap out with our backup sail.

Bibliography

[1] CANBoat. https://github.com/canboat/canboat.

[2] Google Test. https://github.com/google/googletest.

[3] SocketCAN. https://github.com/linux-can/can-utils.

[4] uWebSockets. https://github.com/uWebSockets/uWebSockets.

[5] WPI Sailbot Code. https://github.com/wpisailbot/boat.

[6] NMEA 2000 Explained - *The Latest Word*. Paper, NMEA, 1999.

[7] Nickel Metal Hydride (NiMH) Handbook and Application Manual. http://data.energizer.com/pdfs/nickelmetalhydride_appman.pdf, 2010.

[8] Sailboat Stabilization System. https://web.wpi.edu/Pubs/E-project/Available/E-project-043014-175629/unrestricted/Sailboat_Stabilization_System_MQP.pdf, 2014.

[9] Boost C++ Library. http://www.boost.org/, 2016.

[10] Protocol Buffers. https://developers.google.com/protocol-buffers/, 2016.

[11] Talon SRX. http://www.ctr-electronics.com/talon-srx.html, 2016.

[12] 220WX WeatherStation Instrument. http://www.airmar.com/productdescription.html?id=153, 2017.

[13] Bazel Build System. https://bazel.io, 2017.

[14] Beaglebone black wiki. http://elinux.org/Beagleboard:BeagleBoneBlack, 2017.

[15] Beaglebone RS485 CAN CAPE. http://www.waveshare.com/rs485-can-cape.htm, 2017.

[16] Bullet M Titanium. https://www.ubnt.com/airmax/bulletm/, 2017.

[17] International Robotic Sailing Competition website. http://sailbot.org, 2017.

[18] VNH2SP30-E - AUTOMOTIVE FULLY INTEGRATED H-BRIDGE MOTOR DRIVER. http://www.st.com/en/automotive-analog-and-power/vnh2sp30-e.html, 2017.

[19] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: the insecurity of 802.11. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 180–189. ACM, 2001.

[20] Cross The Road Electronics. *Talon SRX User's Guide*, 2017.

[21] Hadi Saoud, Minh-Duc Hua, Frédéric Plumet, and Faïz Ben Amar. Modeling and control design of a robotic sailboat. In *Robotic Sailing 2013*, pages 95–110. Springer, 2014.

[22] Hadi Saoud, Minh-Duc Hua, Frédéric Plumet, and Faiz Ben Amar. Routing and course control of an autonomous sailboat. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2015.

[23] Daniel Singer, Kelsey Regan, and Dean Schifilliti. The Robotic Automated Wingsail. *Worcester Polytechnic Institute*, 2017.

[24] Ubiquiti Networks. *airOS User Guide*, 2016.

[25] Ubiquiti Networks. *Bullet M Titanium*, 2016.

[26] Lin Xiao and Jerome Jouffroy. Modeling and nonlinear heading control of sailing yachts. *IEEE Journal of Oceanic Engineering*, 39(2):256–268, 2014.

A. Rigid Wing Communications Document

Below is the Rigid Wing Communications Document

Sailbot 2017 Rigid Wing Communications Document

Tucker Martin - 3/14/17 - Version 1.2

Introduction

The goal of this document is to capture all of the relevant information required for effective communication between the Sailboat and Rigid Wing. This introduction provides some background to the problem and notable issues or concerns. Later sections explain in full detail the implemented solution.

Background

Sailbot is an ongoing MQP with the goal of completing a series of challenges using a variety of sensors, actuators, and control algorithms. One of the key actuators of the sailboat is the sail. There are two options for the sail to be used: A traditional soft sail with a wishbone boom, and a Rigid Wing sail developed by another MQP team.

Rigid Wing Sail

The rigid wing was designed as a totally independent system, and relies on wireless communication for all control and sensing between the boat and the sail itself. The chosen solution to be implemented is Wifi (IEEE 802.11n). There is a controller in the wing itself that is capable of interfacing with a wireless module (ESP8266). Serial communication between the main sail controller and this module allows the controller to send and receive messages from the sailboat hull.

Hardware

This section describes the current hardware used by both the rigid wing and the main sailboat with regards to wireless communication.

Sailboat

The sailboat has a BeagleBone Black (BBB) for its primary processor. It runs eLinux (Embedded Linux), more specifically Ubuntu 14.04.

The sailboat also has a wireless Radio (Access Point with DHCP server) that communicates with the BBB. This radio broadcasts a wireless network (802.11n) with the following information:

SSID:	sailbot
Password:	Passphrase123
Wireless IP Range:	192.168.10.20-192.168.10.50

Rigid Wing

The rigid wing controller is a Teensy 3.5 running at 3.3V. The wireless module is an ESP 8266 also running at 3.3V. The module is currently configured to use Serial4, and is wired with the following port information:

Teensy Pin	**ESP 8266 Pin**
GND	2 (GND)
3.3V	7 (VCC)
3.3V	3 (CH_PD)
31 (RX4)	1 (TX)
32 (TX4)	8 (RX)

Rigid Wing States

There are a few situations that exist with regards to the heading, heel, motion, and intent of the sailboat and wing. These states have been distilled down and are represented here.

0. No Lift

This is the default state and the state that the wing should start in. In this state, the trim tab is set to generate zero lift. This state is switched to whenever communication is lost. Once communication is regained, the state can be updated via new incoming messages.

1. Heel Limited Lift - Starboard Tack

The wing is trimmed to generate maximum lift with wind coming from the starboard side of the boat. The heel angle is provided, and as the heel passes a given threshold the sail will let off the power until the heel angle is stabilized.

2. Heel Limited Lift - Port Tack

The wing is trimmed to generate maximum lift with wind coming from the port side of the boat. The heel angle is provided, and as the heel passes a given threshold the sail will let off the power until the heel angle is stabilized.

3. Maximum Drag - Starboard Bias

The sail is set to have maximum drag, articulated all the way to the starboard side of the boat.

4. Maximum Drag - Port Bias

The sail is set to have maximum drag, articulated all the way to the port side of the boat.

5. Manual Control

The sail trim is set by the manual control value sent.

Message Structure

To convey information about sensor data and the current state, messages are exchanged between the sailboat and the rigid wing. There is only one type of message sent by the sailboat to the rigid wing, and another type of message from wing to sailboat. These messages are described below.

Message to the Wing

The message from the sailboat to the wing has 4 values, all represented as non-negative integers with a constant number of digits. These values are listed below. The message is simply a string of characters starting and ending with square brackets, with the integers separated by spaces.

Here is an example message: "[1 076 20 050]"
This message reads: The wing should be in state 1 (Heel Limited Lift - Starboard Tack), with a

current heel angle of 14 (90 - 76) degrees to port, the maximum heel angle is set to 20 degrees from vertical, and the manual control value is 50 (no lift, not used since the state is not manual control)

Message Components:

1) **State** (1 digit)
   The current state of the system as described in the previous section.
   Values: {0, 1, 2, 3, 4, 5}
2) **Heel Angle** (3 digits)
   The current heel angle of the boat as measured by the on-board IMU.
   Values: [0, 180] number of degrees as measured from the port horizon
   0 = sail horizontal on port side of boat
   90 = sail is vertical
   180 = sail is horizontal on starboard side
   60 = sail is leaning at a 30 degree angle towards the port side
3) **Max Heel Angle** (2 digits)
   When in state 1 or 1, this is the maximum heel angle the sail will allow before it starts to reduce the thrust generated.
   Values: [5, 60] number of degrees from vertical (absolute)
4) **Desired Manual Position** (3 digits)
   When in state 7, this is the desired position of the trim tab
   Values: [0, 120] range of servo positions
   0 = full trim to starboard (stall)
   120 = full trim to port (stall)
   60 = centered (no lift)
   35 = trimmed to starboard (some lift generated)

Message to the Sailboat

The message from the wing to the sailboat has 3 values, all integers. It follows the same formatting as the over message type and it is described below.

1) **Apparent Wind Direction** (3 digits)
   The current direction of the wind (relative to the sail)
   Values: [0, 359] Degrees
2) **Servo Position** (3 digits)
   The current position of the servo
   Values: [0, 120] Range of the servo
   0 = full trim to starboard (stall)
   120 = full trim to port (stall)
   60 = centered (no lift)
   35 = trimmed to starboard (some lift generated)
3) **Voltage** (3 digits)
   The current voltage of the sail battery (mV) times some gain factor N
   Values: [0, 999]

Implementation

To send the messages, a TCP connection is used. Once the ESP and BBB are both connected to the AP, the ESP should attempt to make a TCP connection to the BBB on port 3333. Once the connection is made, the BBB will start sending messages at the rate of about 10 per second. For every message received, the ESP should respond with a message of its own, with a minimum of 1 message per second (assuming possible delays due to processing power).

When booted, the Teensy (wing controller) should look for the sailbot access point. It continues searching indefinitely until the right access point is found. It should display the correct LED message while waiting (see chart below).

Once the Teensy connected to the access point, it should start attempting to connect to the specified port (3333). Once a TCP connection is established, it should first wait for a message to arrive. After the first message arrives, a grace period of 1 second is set. If no message is received in 1 second or if the TCP connection is broken, the Teensy will disconnect and try to reconnect to the port. This is repeated until a successful connection is made. Immediately after communication is lost, the wing should reduce thrust to zero.

Here is the chart of signal types that the Teensy should indicate when it is trying to connect to the BBB.

Condition	**Blinking type**
Turned off	Off
Attempting to find AP	Blinking Short On, Long Off
Connected to AP, waiting for TCP	Blinking Short On, Short Off
Connected to TCP and receiving messages	On

B. Rigid Wing Mechanical Interface Document

Below is the Rigid Wing Mechanical Interface Document

Sailbot 2017 Rigid Wing Mechanical Interface Document

Nick Gigliotti

February 10, 2017

1 Introduction

This document will layout the mechanical interface between the Rigid-Wing Sail and the Sailbot main boat. This will serve as a reference for both teams in order to ensure that the two MQPs function seamlessly with one another.

1.1 Units & Standardization

In order to ensure a successful integration of the two projects and allow for easy collaboration, a common set of units and conventions has been decide on. As a result, both projects will be using the imperial measurement system in the design and manufacturing process.

2 Physical Interface

As per the requirements of the Rigid-Wing design, the wing will be attached to the boat to allow for 360° free rotation. In addition, the wing will be free standing and solely supported at the bottom portion of the mast. The rest of this section will specifically layout the dimensions of the interfacing parts and components and link to many of the COTS (consumer off the shelf) components that will be used.

2.1 Mast

The mast that will be used on the rigid-wing is the Naish 2016 Sport RDM Mast 430. This mast is a 100% fiberglass mast with an approximate ID of 1.25" and OD of 1.5", however, the mast is slightly tapered with slight variations in diameter throughout the entire length. For reference, the mast can be found at the here:
`http://www.naishsails.com/product/rdm-sport-430/`

2.2 Mast Tube

In order to contain the mast within the boat, a mast tube will be fiber-glassed into the boat in which the wing mast will be easy inserted into. This tube is a section of 2" SCH 40 PVC pipe with an approximate ID of 2.06" and OD of 2.30". Below is a drawing of this part in figure 1 for reference.

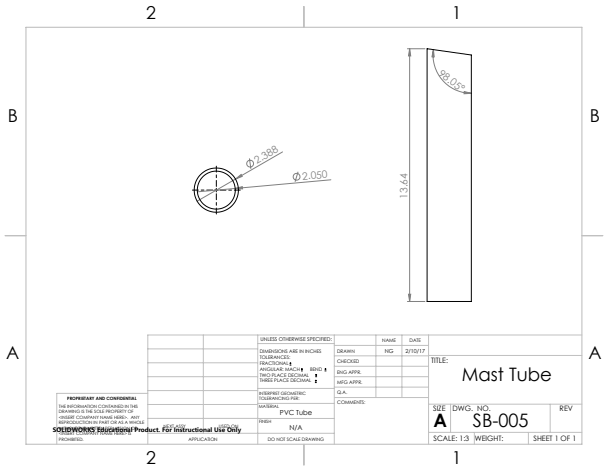


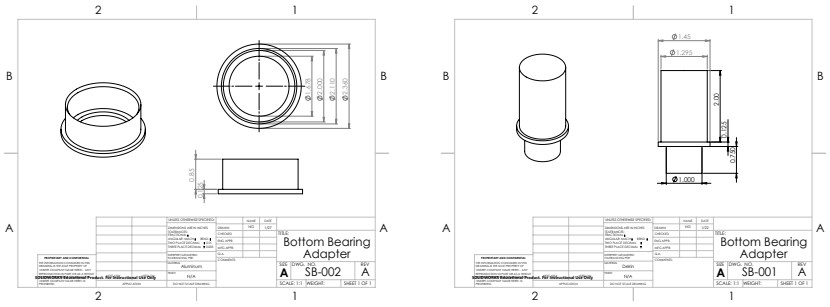
Figure 1: Drawing of Mast Tube made of PVC tube

2.3 Bearings

Since the sail must freely rotate and is powered by the wind, a series of two bearings are used to interface between the mast and the tube which is attached to the boat. Due to the large variations in wind speeds that the boat may experience, these bearings must have low friction, but high damping. Below is a detailed description for each of the two bearings.

2.3.1 Bottom Bearing

The bottom bearing is an off the shelf ball bearing from KMS (part number: AR16DR-1-G). This bearing has an ID of 1" and an OD of 2". In order to interface between this bearing and the mast, an adapter plug will be made out of delrin which will fit firmly into the ID of the mast and the ID of the bearing. Similarly, the bearing is attached to the mast tube with an adapter on the OD of the bearing. This adapter is pressed into the mast tube and is not removable, however, the bearing is loosely fit into this adapter to allow for removal of the bearing. Drawings of each of the these parts are shown below in figures 2a & 2b.



(a) Drawing of bottom bearing OD adapter (b) Drawing of bottom bearing ID adapter

Figure 2: Drawings of Bottom Bearing Parts

2.3.2 Top Bearing

The top bearing is a machined delrin bushing that fits around the OD of the mast. This bearing is custom-made to fit onto the mast and provide low-friction rotation. In addition, the mast is sanded smooth in the area of the bearing to provide a better bearing surface for the delrin. A drawing of this bushing is shown below in figure 3.

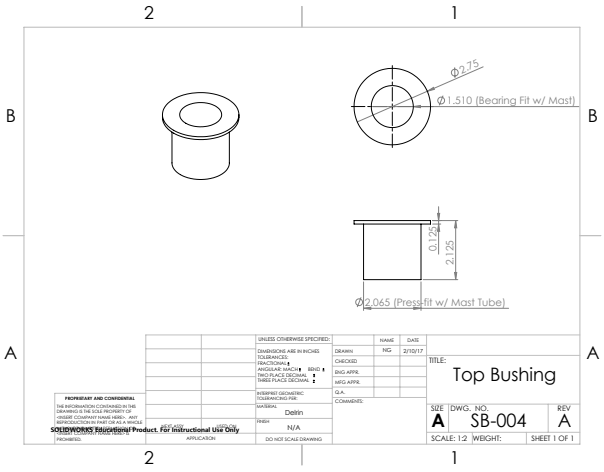
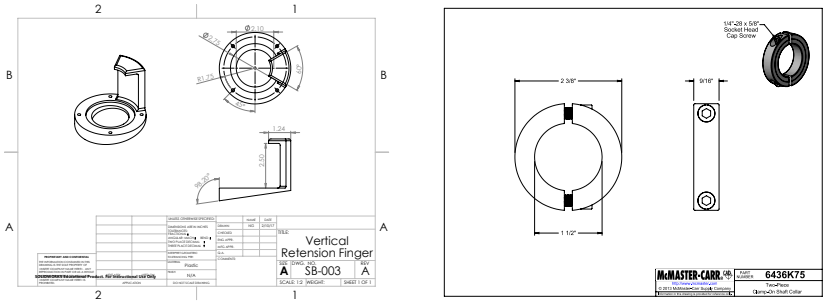


Figure 3: Drawing of top delrin bushing

2.4 Vertical Capture System

In the case of unpredictable conditions and abnormal boat behavior, a system of vertically retaining the mast and wing-sail in the boat will exist. This system will consist of a shaft collar fixed to the mast and a L-shaped finger mounted to the deck. In normal operation, the shaft collar and retention part will not be in contact, but if the boat were to capsize the shaft collar would hit the retention part and prevent the mast from coming out. Figure 4 provides drawings of these parts and show their intended use.



(a) Drawing of Vertical Retention Part (b) Drawing of Shaft collar attached to the mast

Figure 4: Drawings of mast retention parts

3 Clearance & Space Allocation

3.1 Height Restriction

Due to competition rules for the IRSR, the boat may not exceed a height of 5m from the bottom of the boat (including keel and ballast) to the very top of the sail. In order to ensure that the boat will fall within the bounds of this rule, some height allowances have been allocated to each of the systems. Table 1 and figure 5 detail these allowances.

Table 1: Height Allowances for each of boat components

Name	**Diagram Label**	**Allocated Height (in.)**
Keel	A	54
Hull	B	10
Deck Clearance	C	8
Wing-Sail	D	124
Total (<5m / 196.86")	N/A	196

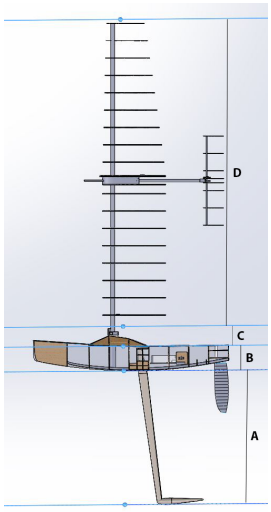


Figure 5: Diagram of Height Allowance Measurements

3.2 Deck Clearance

In order to allow for additional components to be placed onto of the deck of the boat, a clearance distance was established early on in the project to allow both projects to proceed with a parallel design stage. This deck clearance is set at 8" above the main portion of the deck. Figure 6 shows exactly where this clearance is set at.

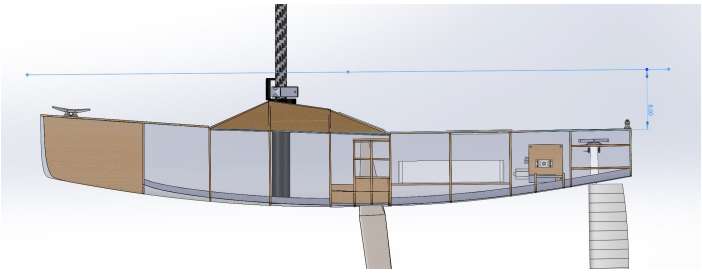


Figure 6: Vertical Deck Clearance Plane

C. SCAMP

Below is the technical document for the SCAMP (Sensing and Controlling Atmega Messenger and Processor)

Sailbot 2017 SCAMP Document

Tucker Martin - 3/14/17 - Version 1.2

Introduction

The goal of this document is to capture all of the relevant information regarding the development of the SCAMP, or Sensing and Controlling Atmega Messenger and Processor. The SCAMP was developed by the Sailbot 2017 MQP at WPI to fill a specified need: A speed controller capable of reading and writing messages to the NMEA 2000 communication channel. The SCAMP also has additional functionality including analog and digital inputs and outputs, and the ability to perform PID control.

Design Requirements

The SCAMP was designed to fulfill the necessary design requirements listed below.

- Driving the highest current motor (2-3A continuous, 15A peak)
- Receiving and sending CAN messages

Some additional features were also included to allow the replacement of other boards and COTS components:

- Digital inputs/outputs
- Analog inputs/outputs
- PWM output for controlling a servo
- Quadrature encoder support

Schematic

Below is a schematic for SCAMP 2.4, the latest iteration.

D. Full Boat Render

Below is a render of the full boat.

E. Full Results Graph

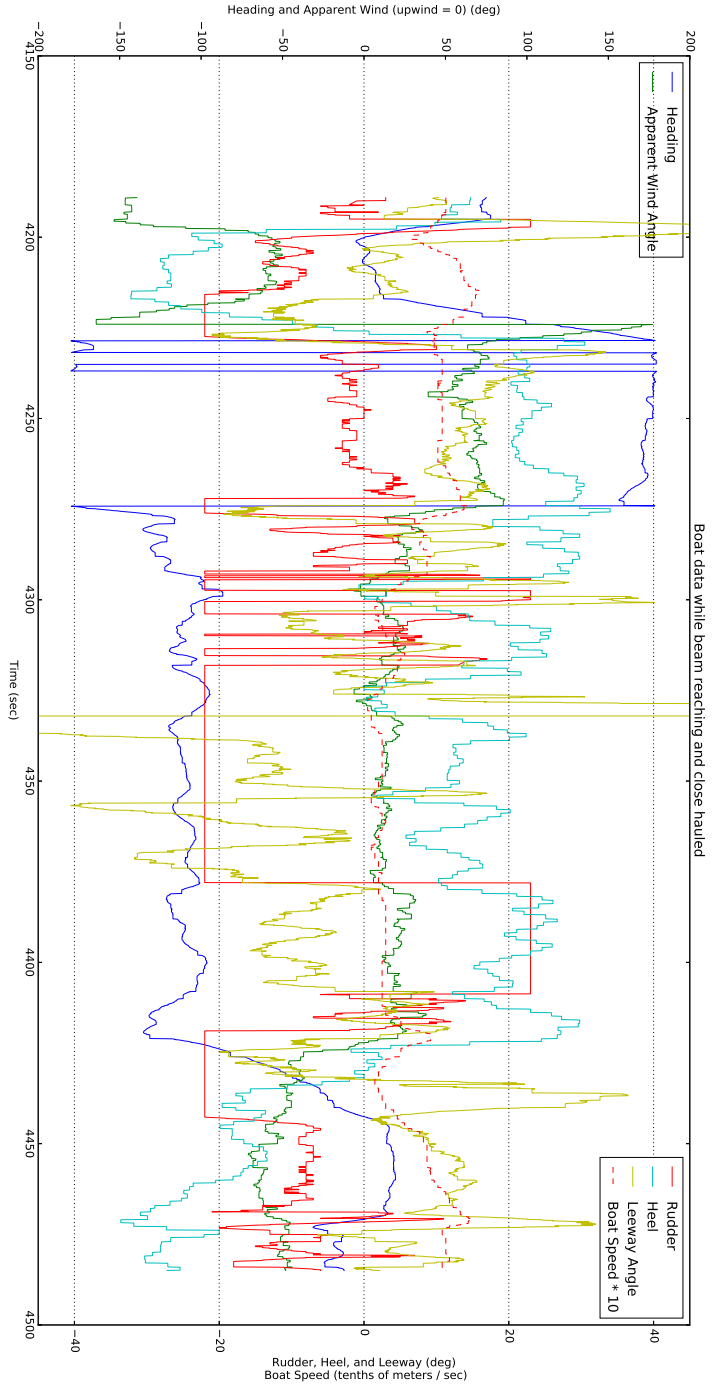


Figure E.1: Various Boat Data from Same Course as Figure 5.2