

1-2014

Senior Project Design of a Two Meter Autonomous Sailboat

Ben Williamsz

Julia Kane

Richard J. Hartnett

Peter F. Swaszek

University of Rhode Island, swaszek@uri.edu

Follow this and additional works at: https://digitalcommons.uri.edu/ele_facpubs

Citation/Publisher Attribution

Williamsz, B., Kane, J., Hartnett, R.J., Swaszek, P.F., "Senior Project Design of a Two Meter Autonomous Sailboat," *Proceedings of the 2014 International Technical Meeting of The Institute of Navigation*, San Diego, California, January 2014, pp. 594-600.

Available at: <https://www.ion.org/publications/abstract.cfm?articleID=11530>

This Conference Proceeding is brought to you by the University of Rhode Island. It has been accepted for inclusion in Electrical, Computer, and Biomedical Engineering Faculty Publications by an authorized administrator of DigitalCommons@URI. For more information, please contact digitalcommons-group@uri.edu. For permission to reuse copyrighted content, contact the author directly.

Senior Project Design of a Two Meter Autonomous Sailboat

The University of Rhode Island Faculty have made this article openly available.
Please let us know how Open Access to this research benefits you.

This is a pre-publication author manuscript of the final, published article.

Terms of Use

This article is made available under the terms and conditions applicable towards Open Access Policy Articles, as set forth in our [Terms of Use](#).

Senior Project Design of a Two Meter Autonomous Sailboat

Ben Williamsz, *U.S. Coast Guard Academy*
Julia Kane, *U.S. Coast Guard Academy*
Richard J. Hartnett, *U.S. Coast Guard Academy*
Peter F. Swaszek, *University of Rhode Island*

BIOGRAPHIES

Ben Williamsz is a 2013 graduate of the U.S. Coast Guard Academy, and majored in Electrical Engineering. He is currently an Ensign in the U.S. Coast Guard, serving aboard USCGC Harriet Lane in Portsmouth, VA.

Julia Kane is a 2013 graduate of the U.S. Coast Guard Academy, and majored in Electrical Engineering. ENS Kane received the ION sponsored student award for best navigation-related senior thesis, which documented the project efforts that ENS Williamsz and ENS Kane pursued together in the autonomous sailboat project. She is currently serving aboard USCGC Kikui in Honolulu, HI.

Richard J. Hartnett is a Professor in Electrical and Computer Engineering at the U.S. Coast Guard Academy in New London, CT, having retired as an O-6 from the USCG in the summer of 2009. He received his BSEE degree from the U.S. Coast Guard Academy, the MSEE degree from Purdue University, and his Ph.D. in EE from the University of Rhode Island. His research interests include efficient digital filtering methods, improved receiver signal processing techniques for electronic navigation systems, and autonomous vehicle design.

Peter F. Swaszek is a Professor in the Department of Electrical, Computer, and Biomedical Engineering at the University of Rhode Island. His research interests are in statistical signal processing with a focus on digital communications and electronic navigation systems. He spent the 2007-08 academic year on sabbatical at the U.S. Coast Guard Academy working on a variety of RF navigation systems.

ABSTRACT

Here we describe an Electrical Engineering senior design project that was performed at the U.S. Coast Guard Academy (USCGA) in New London, CT, during the 2012-2013 academic year. The objective of this project

was to design and prototype a fully autonomous sailboat capable of competing in the annual SailBot competition in the 2-meter class. SailBot is a North American competition for robotic sailboats, and competition is held at the 1-meter, 2-meter, and “open class” (up to 4-meters in length). While competition includes manual as well as autonomous events, our primary focus this year has been to design a vessel that is capable of autonomous navigation around a race course. A secondary focus (also useful for testing) has been to include capability for manual navigation and sail trim. The system employs Matlab™ scripts, an Airmar weather sensor, and various servos (rudder and sail), to perform autonomous (or manual) navigation to a given GPS waypoint. This project encompasses all aspects of the engineering design process, from physically altering the platform to designing and testing a digital rudder controller. This year’s two-person team focused on adding autonomous functionality and robust controller design, while increasing the both hardware and software reliability.

INTRODUCTION

The SailBot project began several years ago as a joint effort between several Naval Architecture/Marine Engineering students and several Electrical Engineering students at the U.S. Coast Guard Academy. Initial project requirements were to perform a complete design of a 2-meter vessel that could be controlled (both rudder and sail trim) via radio control, and sail reliably for a period of several hours. Since that time, USCGA’s SailBot has evolved to be a sleek 2-meter custom hull, constructed with carbon fiber over a foam core. The sail plan is that of a traditional sloop-rig with a panel-cut Mylar-over-Kevlar main sail, and a self-tacking jib on a carbon fiber mast.

This year we focused on system identification and autonomous controller design, such that the vessel is now capable of fully autonomous sailing over a desired series of waypoints, over a 3km race course in open water. In

addition, the vessel sends telemetry data while underway, so monitoring of navigation and controller performance is relatively easy. Finally, manual control can be invoked at any point by toggling one switch on an RC controller.

The sails and rudder for SailBot are actuated by two separate servos that are controlled via Matlab™ scripts running on a portable computer inside the 2-meter vessel. Data such as vessel heading, relative and true wind velocities, and GPS/WAAS positional information are provided by a single onboard sensor (Airmar PB200), and this NMEA-0183 data is transmitted to our onboard portable computer (and put into a MySQL database), for use by the sailing tactics algorithm, the sail trim algorithm, and the proportional/integral controller for the rudder. All data acquisition, control, and sailing tactics algorithms run at a 10Hz sampling rate.

In addition to a technical description of USCGA's SailBot vessel, we describe the process for system identification of the discrete-time step invariant equivalent transfer function of USCGA's SailBot, and we describe the resulting transfer function (of rudder input to rate of turn output). We present Simulink™ simulations of the platform, and proportional/integral (PI) rudder controller performance. For the presentation itself, we show video clips of the vessel in action, in various wind conditions, with heading and sail control being performed autonomously. We also discuss future directions, including controller adaptation in varying wind conditions and sea states, plus sensor integration employing Kalman filters.

DESIGN REQUIREMENTS AND CONSTRAINTS

Design requirements for each class of SailBot are delineated in the SailBot competition guidelines. These requirements are broken down into two categories: the physical requirements for the boat, and the performance requirements for the competition.

The physical requirements restrict the dimensions of the competing vessels. For the 2-meter class, the waterline length must not exceed 2 meters, the beam must not exceed 3 meters, and the draft must not exceed 1.5 meters. Additionally, the total height of the vessel may not exceed 5 meters. USCGA's SailBot fits well within these dimensions.

The competition itself is divided into four challenges: fleet racing, autonomous racing, station keeping and endurance racing. In order to compete successfully, a vessel must be able to navigate all of these challenges under full autonomous mode (except during the fleet racing). Point penalties are awarded for taking the vessel out of autonomous mode. From these individual challenges, our specific design requirements were:

- The vessel must navigate to a given set of pre-defined GPS waypoints.
- The vessel must, if necessary, tack and jibe to reach these waypoints.
- The vessel must be able to navigate under remote control at any time.
- The vessel must have enough endurance and watertight integrity to finish a 3km race in open water.
- The vessel must send and receive telemetry data while underway.

In addition to the above requirements, we built on previous iterations of SailBot. Design decisions from these previous iterations persist in the current version of Sailbot.

DESIGN APPROACH: PHYSICAL PLATFORM AND SOFTWARE

Although the design requirements of the SailBot competition are specific, they leave room for creativity in implementation. We prioritized the performance characteristics that the SailBot competition requires in order to make numerous design decisions about the physical platform and the software.

Airmar Sensor: The previous iteration of SailBot had the Airmar weather sensor (Figure 1) mounted on the deck directly underneath the main sail, as an improvement over mounting on top of the mast. Unfortunately the wind data was inaccurate due to the close proximity to the turbulent water, and due to the redirected laminar air flow from the sails.



Figure 1 – Airmar PB-200 GPS/WAAS weather sensor.

We determined that leaving the Airmar weather sensor mounted on the deck was not an acceptable option, so we considered three options: shifting back to a mast-top mount, mounting it on a small mast on the bow of the boat, and mounting it on a small mast on the stern of the boat. Returning the sensor to mast-top location was a concern because of its weight and the potential negative impact to the boat's dynamic stability. Adding weight so high would cause a large moment when the boat is heeled over. Mounting the sensor on a small mast on the bow of

the boat proved unsuitable as well. There is not a lot of reserve buoyancy in the bow, so when SailBot is sailing downwind in a breeze of over 10 knots, it tends to plow through the waves and submerge the bow. Adding more weight would only worsen this problem.

In the final design of SailBot, the Airmar sensor is located on a small mast on the stern of the vessel (Figure 2). The sensor placement is high enough above the waterline, and far enough away from the sails to provide reasonably accurate wind speed/direction data. This location also provides better GPS satellite constellation visibility.



Figure 2 – Redesigned Airmar sensor support.

Sheeting system: Earlier versions of our SailBot vessel (up to Spring 2012) used a single pulley system to adjust both the main sail and the jib. This was inadequate because the layout did not allow SailBot to let its sails out fully, significantly reducing the number possible points of sail. In order to use the full range of sail trims, we redesigned the deck layout to more closely resemble a traditional sailboat, as shown in Figure 3.

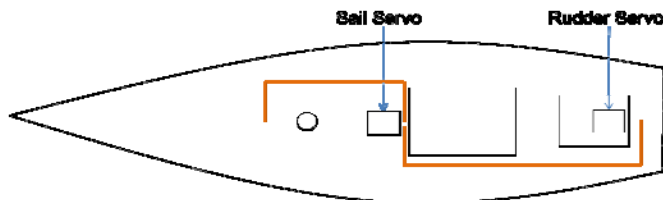


Figure 3 - SailBot deck layout (top view).

Using the new layout, SailBot is able to use every sail trim, from close-hauled to a downwind run.

One final problem remained, however, in that the sheets would frequently foul around the sail servo sheave, particularly when the sails were luffing and the sheets were not under tension. In order to solve this problem, we designed a new sail servo sheave with a much larger diameter and deeper grooves so the sheets no longer foul themselves around the sheave. We “printed” several copies of our new design from our 3-d printer, and this new design worked well.

Hatch problems: One of the primary problems with last year’s iteration of SailBot was the lack of watertight

integrity. Water easily leaked through the inspection porthole covers on the deck, which screwed on and off to allow access to the electronics within the hull. We considered just replacing the circular inspection port covers, but instead we created our own design and used heavy-duty neoprene gasket material and fiberglass to seal rectangular holes in the deck. The covers are easy to remove and replace, and they allow easy access to the electronics.

Rigging adjustments: During the initial remote control tests of SailBot there was no tension on the leech of the jib. When SailBot was sailing anything lower than a beam reach, the jib would luff, making it essentially useless. In order to remedy this problem, we considered two different types of vang: a traditional vang, mounted to the deck, and an inverted vang, attached to the forestay. A traditional, deck-mounted boom vang was not practical because there was such little clearance between the jib boom and the deck. Instead SailBot relies on an inverted vang, which uses the forestay to push the boom down and provide leech tension (Figure 4.)

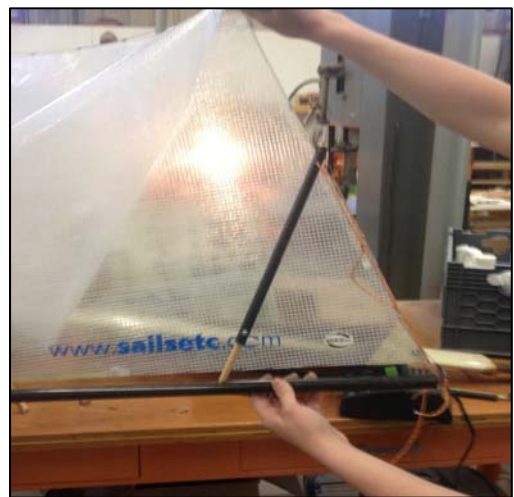


Figure 4 – Inverted vang modification.

Another adjustment that we made to the rigging was a removable 25% reef in the main sail. With the full rig, SailBot was significantly overpowered in anything over 10 knots of breeze. Now we are able to test SailBot in a greater range of weather conditions.

Software (Data Storage): One of the essential components of SailBot’s software is a central data repository. While there were several different options for storing the necessary data, this year we chose to use a MySQL database (as opposed to flat text file structures or the Matlab workspace environment itself), in order to allow multiple scripts to access the data simultaneously. Using a database structure allowed multiple processes to access data at once and provided for easy sorting and analysis. We chose MySQL specifically because it is

open-source software and the MySQL Workbench environment offers a simple and easily understandable user interface.

Software (Modularity): For a marine system, SailBot is a reasonably high dynamic platform, and requires a reasonably high sampling rate in order to do an adequate job of data acquisition to support digital control of heading and sail trim. In order to improve sampling rate, we chose to separate SailBot’s software functionality into three distinct modules (as opposed to using a single sequential Matlab script). SailBot relies on Windows XP’s thread management to divide processor power and execute all three code modules simultaneously (Figure 5), and we are now able to run the data acquisition and control algorithms at a 10Hz sampling rate. This sample rate has proved to be sufficient for our design.

Current Software

- Windows XP SP3
- 3 Matlab scripts running concurrently
 - Data Acquisition
 - Navigation Algorithm
 - Rudder/Sail Trim Controller
- MySQL Database
 - High throughput (10Hz)
 - Universal access
 - Hosted on onboard computer
 - Accessible to shore-side computer via 802.11 network








Figure 5 – Software description summary.

Software (Rudder Controller Design):

Our first step in designing SailBot’s rudder controller was to perform a “system identification,” in order to characterize the open-loop dynamics of the SailBot platform. Our intent was to obtain a discrete-time step invariant equivalent of the open-loop continuous time SailBot transfer function, with rudder angle as input, and rate of turn as output. Here the original idea was to provide SailBot with random rudder commands, measure the actual rudder angle, and measure rate of turn at a 10Hz rate, under sail. In theory that idea would work, however we felt that sailing in random directions at random points of sail would produce inconsistent results. Instead, we chose to implement a simple proportional controller, with proportionality constant of $K_p=0.4$, and identify the open-loop system dynamics from within a closed loop system (Figure 6), which produced data shown in Figure 7. This indirect method of measuring rate of turn for a given rudder command worked very well, and we used least-squares approximation methods to obtain a transfer function (rudder angle input to heading

output). Using methods from [1], the final step invariant equivalent transfer function (rudder angle input to heading output) for the SailBot platform was identified to be

$$H(z) = \frac{0.0118 - 0.0169z^{-1} + 0.0102z^{-2} + 0.0021z^{-3} + 0.0068z^{-4}}{1 - 1.9921z^{-1} + 0.9921z^{-2}}$$

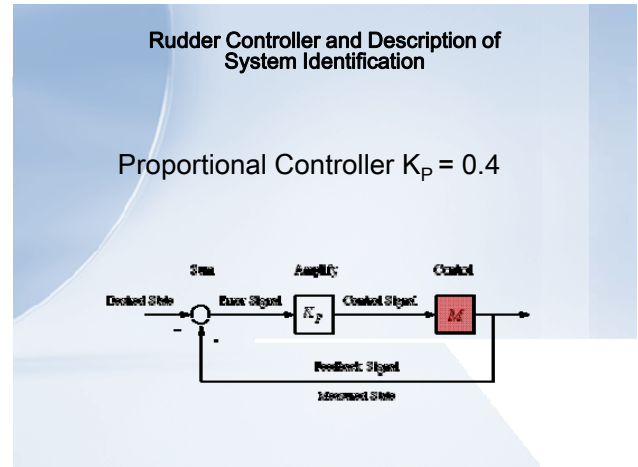


Figure 6 – Simple proportional controller used for system identification.

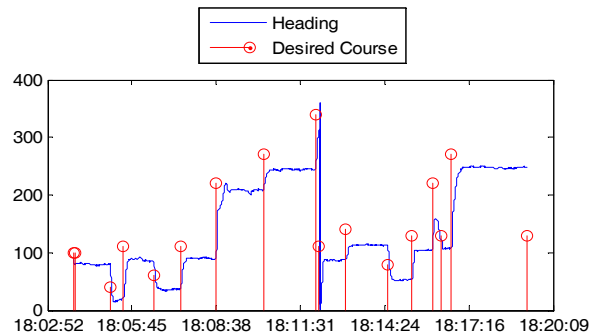


Figure 7 – Data used for system identification.

By running Simulink™ simulations with the transfer function above, we noticed that there was steady state error, and overshoot of less than 10 percent. We decided that a proportional-integral (PI) controller would be most suitable for SailBot, to eliminate the problem of steady-state heading error. Using root-locus techniques and Simulink™ simulations, we arrived at final values for proportional (K_p) and integral (K_i) coefficients of 0.5 and 0.002, respectively, for our PI heading controller. This controller provided zero steady-state heading error, and overshoot of less than 10 percent.

Software (Sail Controller Design): In the rudder controller it was relatively simple to implement a closed-loop controller since it is possible to get feedback in the form of heading error. With the sail controller, however,

we had no way to get feedback relative to boom position, so we chose to implement an open-loop controller with four discrete states. SailBot's sail controller algorithm determines what its sail trim should be by comparing the heading to the true wind direction. One of four sail trim states is then chosen, and the software then sends the appropriate command to the sail winch servo.

Software (Navigation Algorithm): In order for SailBot to compete in the autonomous portions of the competition, it needs to be able to sail autonomously to a selected waypoint (latitude/longitude). Our navigation algorithm uses true wind direction and current GPS location to determine the optimal course to a waypoint, taking into account the fact that no sailing vessel is capable of sailing directly into the wind. In designing this algorithm, we used the increment and iterate method, beginning with a very simple algorithm, testing it, and then adding increased functionality.

Our first step was to make sure that SailBot could compare its current location with the waypoint coordinates and determine the correct course on a beam reach. Once we were satisfied that our main navigation algorithm was providing the correct desired course to the waypoint to the rudder controller, we moved on to the sailing tactics.

SailBot cannot sail within 55° of the true wind direction. Although it is capable of trimming the sails to sail directly downwind, we elected to not sail within 30° of straight downwind because we were worried that small shifts in breeze could cause SailBot to tack unintentionally. When the course directly to the waypoint was too close to the wind direction or too close to a downwind run, the logic in SailBot's navigation algorithm determines when it is appropriate to tack or jibe. It does this by comparing the course to the waypoint with the wind direction. Whenever the course to the waypoint is not within 55° of the wind direction or within 30° of a downwind run (the "go zone"), SailBot will steer directly to the waypoint. When the course is within the previously stated angles, Sailbot will simply continue on the same tack, sailing at the closest angle possible that does not put it into the "no-go zone." As soon as it can sail directly to the waypoint on the opposite tack, SailBot will either execute a tack or a jibe. We tested SailBot multiple times, forcing it to sail at all points of sail, to pre-determined waypoints under autonomous control, and the algorithms performed flawlessly.

RESULTS

Our final SailBot vessel design is shown in Figure 8. Discussion of results will be described in separate sections on the rudder controller, sail controller, and the navigation computation algorithm.



Figure 8 – SailBot final design.

Rudder Controller: Our final rudder controller is a proportional-integral design as described in the design process. The final version of the controller uses a proportionality coefficient of .5 and an integral coefficient of .002. Figure 9 shows actual results after rounding a waypoint, showing the actual heading response to a desired course change of roughly 170 degrees. Note that we were able to eliminate steady state error, and there are no oscillations in the controller response.

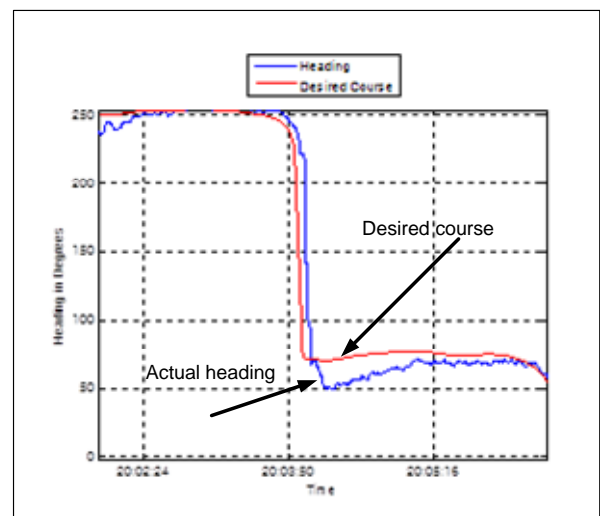


Figure 9 – SailBot rudder controller performance showing response to a desired course change of 170 deg.

Sail Controller: SailBot’s sail controller is a basic open-loop system that resolves relative wind direction into one of four discrete sail trims, corresponding to each point of sail. Recognizing that we do not want to trim the sail based on noisy measurements of the true wind, the sail trim algorithm uses a low pass filtered “true wind” (Figure 10) as the single input. This simple algorithm provides a reliable way of trimming the sail for up and downwind courses, and while it is likely a suboptimal algorithm, it performs flawlessly and allows for fully autonomous sailing.

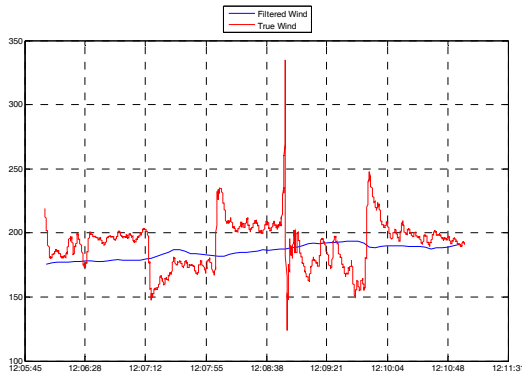


Figure 10 – SailBot’s true wind and low pass filtered wind measurements.

Navigation Algorithm: Sailbot’s navigation algorithm is a basic “bearing-to-waypoint” optimized solution. It operates by performing real-time calculations of the vessel’s current bearing to the desired GPS waypoint, and matching the desired heading to it as closely as possible.

SailBot’s tacking and jibing angles are pre-set to 55 and 150 degrees off of the true wind direction, respectively. When the bearing to the waypoint falls within these angles (i.e., when SailBot cannot sail directly to the waypoint), the navigation algorithm keeps SailBot either as close to or as far from the wind as possible until the waypoint bearing is outside the tacking or jibing angle on the opposite tack. Figure 11 shows the results of an upwind test of the navigation algorithm (winds out of 177 degrees true), and demonstrates SailBot’s ability to tack itself upwind to a waypoint. At the same time, Figure 11 also highlights a limitation of not calculating environmental state variables such as set and drift (vector velocity for current), since the final leg towards the “finish” point is bowed toward the south, presumably because of water current.

Despite the fact that SailBot will reliably navigate to any waypoint, the bearing-to-waypoint optimization that we implemented is not optimal.

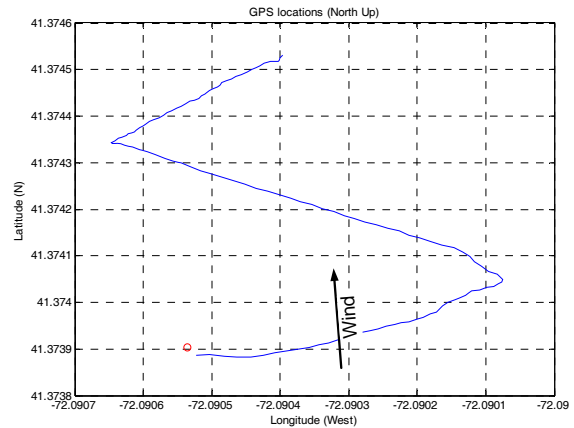


Figure 11 - Autonomous navigation upwind.

FUTURE WORK

Software: Although SailBot is now fully capable of autonomous navigation, there is considerable room for further development and optimization. The primary focus will be to increase the efficiency and robustness of the autonomous navigation algorithm.

More specifically, in order to avoid hitting marks on the challenge course, a function should be implemented to automatically navigate SailBot around marks rather than navigating directly to them. One suggestion for solving this problem is to automatically navigate to imaginary GPS waypoints a given radius from the desired waypoint.

In addition, SailBot’s current navigation algorithm does not calculate various state variables that are essential for optimizing a route to a given waypoint. Performing a mathematical estimate of state variables such as current, crab angle, and rudder offset will help future versions of SailBot sail more efficiently.

We also recognize that collision detection and avoidance are important features for any autonomous vehicle, and it would be desirable to load applicable chart data into the navigation algorithm to avoid marked obstacles and shoal water.

Hardware: Physical improvements can be made to SailBot to make it both faster and more controllable. One such change would be to increase the size and efficiency of the rudder. By decreasing the chord length and increasing the overall draft of the rudder, SailBot’s control algorithms could more reliably sail a given heading. In addition, this new rudder would improve the sailing characteristics in high-wind conditions.

CONCLUSIONS

SailBot is now a robust platform, capable of completing a competition challenge. Valuable lessons were learned about real-world, real-time digital control of a continuous time system.

ACKNOWLEDGMENTS

We would like to thank all EE faculty at USCGA for their dedicated support in this project. We would also like to thank ETCS Ken McKinley, who helped us overcome multiple challenges with our onboard electronics. This project would also not have been possible without the support of Mr. Jack Neades, the offshore sailing coach at the U.S. Coast Guard Academy. Mr. Neades spent countless hours of his personal time helping us to improve SailBot's physical platform, and giving us advice on how to improve the performance of the vessel. Thanks also to the entire waterfront staff for allowing us to use their facilities to work on and test our project, and for their help and expertise.

DISCLAIMER AND NOTE

The views expressed herein are those of the authors and are not to be construed as official or reflecting the views of the U.S. Coast Guard or Department of Homeland Security.

REFERENCES

- [1] Phillips, C. and Nagle, T., *Digital Control System Analysis and Design*, 3rd edition, Prentice Hall, Inc., New York, 1995.