A Control System for a Small Autonomous Sailing Vessel

Tobias Ferl and Stephen Hills
Department of Engineering, Electrical Engineering
United States Coast Guard Academy

As a yearlong undergraduate project, we are developing a 1.2-meter autonomous sailboat for a trans-Atlantic attempt, from New England to Ireland, in the summer of 2020. The control system for the sailboat requires sensors for wind and location, a solar rechargeable power system, a low-powered microcontroller, and mechanical actuators for sail and rudder control. In addition to the hardware design, we are also developing custom software for autonomous navigation and control of an experimental wing-sail. Initial testing of a hardware mockup had positive results, however rudder response times were too slow. We found that the rudder was delayed as it waited for the complex navigation subroutine to execute. Our solution uses proto-threading to mimic parallel processing on a sequential processor, allowing us to run multiple subroutines at the same time. This spring, we will install the full, improved control system in the sailboat's hull and perform testing on the water.

Corresponding Author: Tobias Ferl, tobias.s.ferl@uscga.edu

Introduction

Autonomous systems are increasingly prevalent in the maritime industry with autopilots, dynamic positioning systems, automated cargo handling, and monitoring systems improving efficiency, reliability, and safety of transportation at sea. Automation streamlines shipboard operations, sometimes reducing merchant vessel crew sizes from upwards of 40 to as few as 15 people [1]. With clear benefits, maritime automation has grown interest at the undergraduate level.

The Microtransat Challenge seeks to improve the state of maritime automation through competition. The goal is to complete a trans-Atlantic crossing with an unmanned, wind-powered vessel up to 2.4 meters in length. While the first unmanned sailing vessel successfully completed the course in August 2018, no completely autonomous vessel has finished [2].

As the leading federal agency overseeing maritime industry, the Coast Guard has an operational interest in fostering and understanding automated technologies and their applications. The United States Coast Guard Academy has worked in the field of automation and autonomous systems for some time, including autonomous aerial vehicles, ground vehicles, and one prior autonomous sailing vessel (ASV) project, but has yet to attempt the Microtransat.

We are currently in the process of creating an ASV, though a yearlong undergraduate directed study, to attempt the Microtransat Challenge. The design tasks are unique and have already bridged numerous engineering disciplines including systems engineering, electrical, mechanical, and marine engineering. Specific topics include controllers, control algorithms, and power systems.

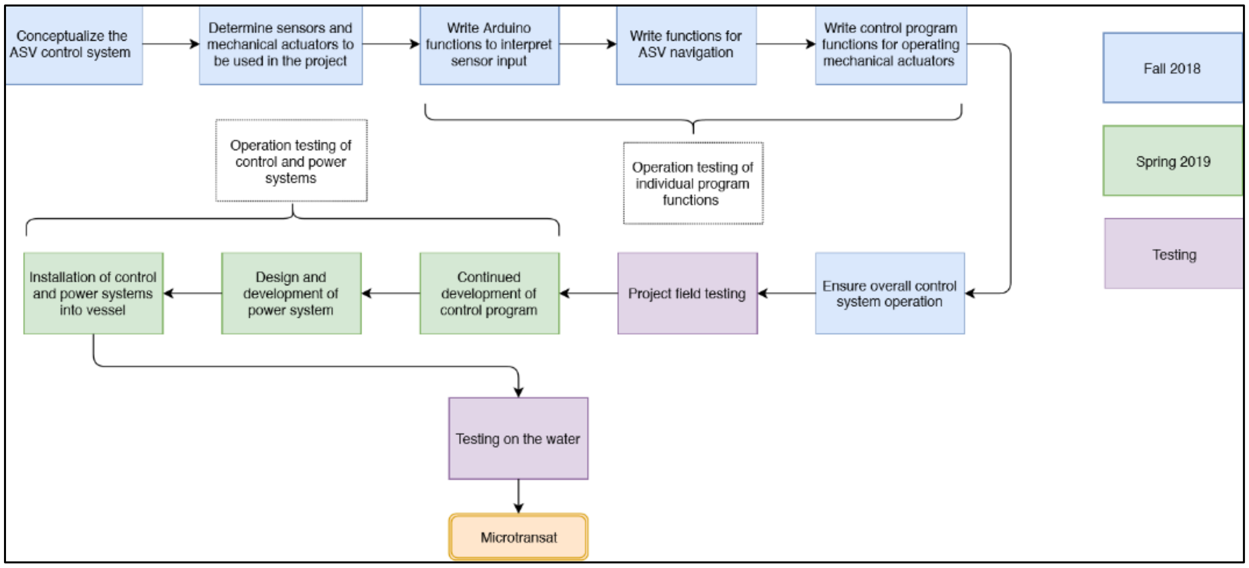
Conceptualize the ASV control system
Determine sensors and mechanical actuators to be used in the project
Write Arduino functions to interpret sensor input
Write functions for ASV navigation
Write control program functions for operating mechanical actuators
Fall 2018
Spring 2019
Testing
Operation testing of control and power systems
Operation testing of individual program functions
Installation of control and power systems into vessel
Design and development of power system
Continued development of control program
Project field testing
Ensure overall control system operation
Testing on the water
Microtransat

Figure 1 Flow chart of project tasks

Project Overview

Work on the ASV project began during the Fall 2018 semester and will continue through at least May 2019 (Figure 1). Our primary Fall semester goal was to illustrate the project's viability by creating a working "mockup" of the control system. Dry land testing with the mockup on a small cart allowed us to simulate course deviations, wind shifts, and other environmental stimuli while observing system responses (Figure 2).

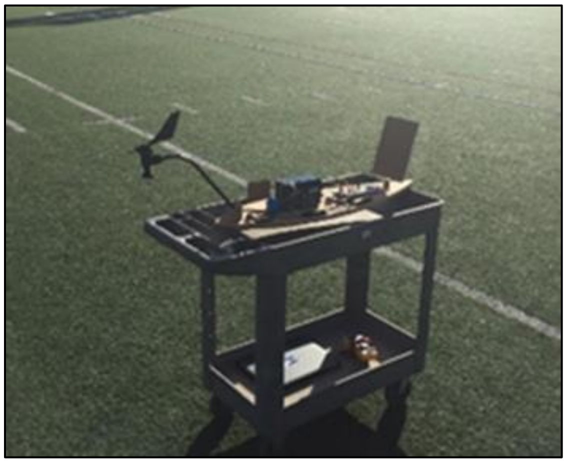

Figure 2 Mockup on cart for testing

The result of the Fall semester is shown in Figure 3. Simple cardboard indicators attached to the servos illustrate rudder and wing-sail movements.

Rather than a traditional soft sail, we decided to experiment with a rigid wing for our project. High performance sailing vessels, such as America's Cup yachts, have pioneered the use of rigid wings [3]. These airfoils may have potential advantages for small vessels as well. The Naval Academy found the wing-sail to be promising for ASV performance across a variety of wind angles, strengths, and environmental conditions [4].

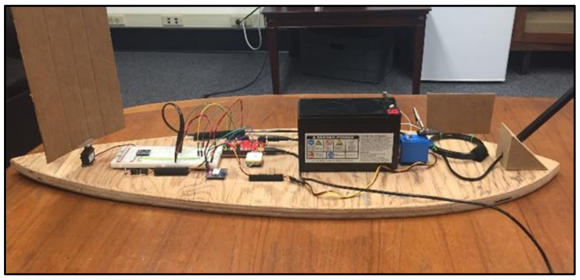

Figure 3 Final product of the Fall 2018 semester

Work in the Spring 2019 semester has shifted focus from concept validation to optimizing the 2018 design in preparation for on the water testing and an attempt at a trans-Atlantic crossing.

Preliminary Control System Design

Hardware Architecture

The ASV's control system hardware is straightforward. The vessel must sense wind direction, determine its position to calculate a desired heading, and compare its desired and current headings to determine required course changes. These requirements lead to the ASV's hardware architecture (Figure 4). A Davis 6410 Vantage Pro2 anemometer senses wind speed and direction, although we are looking for a more rugged component to replace it. The Waveshare NEO-7M-C GPS provides the ASV with positioning data. Documentation available for both the Waveshare and the Davis anemometer made them excellent candidates for our control system's development [5] [6]. Our heading sensor was the Adafruit LSM303, which includes an accelerometer and magnetometer that can yield tilt-compensated magnetic compass headings.

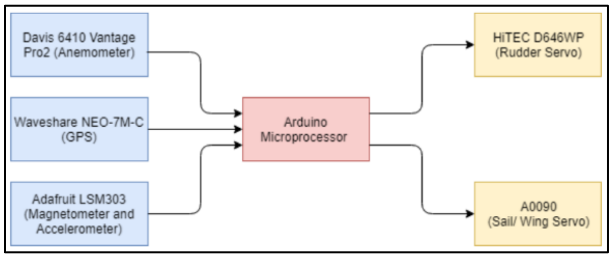
Davis 6410 Vantage Pro2 (Anemometer)
Waveshare NEO-7M-C (GPS)
Adafruit LSM303 (Magnetometer and Accelerometer)
Arduino Microprocessor
HiTEC D646WP (Rudder Servo)
A0090 (Sail/ Wing Servo)

Figure 4 Block diagram of ASV control system hardware components

The ASV's microprocessor is a variation of the Arduino Uno. Two of the board's analog input pins are utilized for the LSM303 as an I2C (Inter-Integrated Circuit) connection, one analog input is reserved for the anemometer, and two dual-purpose input/output (I/O) pins for the GPS (Figure 5). Two other I/O pins deliver the required pulse width modulation (PWM) for the rudder and wing servos (Figure 5).

The last components of the control system hardware are mechanical actuators for wing-sail and rudder movement. During initial development two different servos were utilized for these actuators: an A0090 micro sized servo that was included in the Arduino's starter kit, and a HiTEC D646WP.

Component selection involved substantial research to ensure their interoperability. Additionally, we were required to read technical documentation, troubleshoot, and make component substitutions when devices did not meet project requirements.

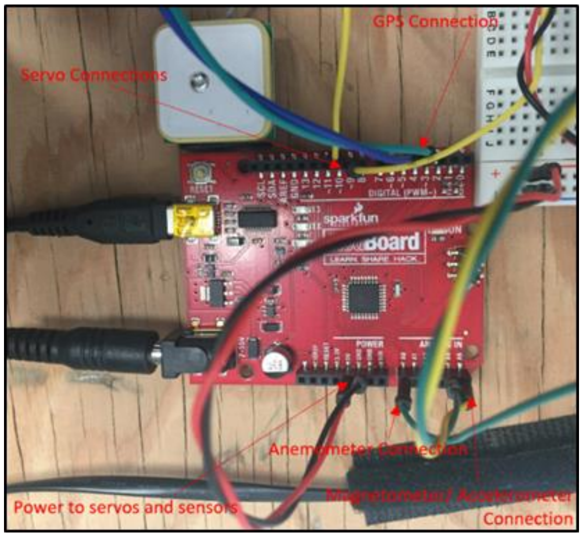
GPS Connection
Servo Connections
Anemometer Connection
Power to servos and sensors
Magnetometer/ Accelerometer Connection

Figure 5 Arduino microprocessor with connections labeled

Control Code

Developing the algorithms and controllers for the project was the greatest technical challenge for the first semester. Figure 6 illustrates the general flow chart of the control code by December 2018. Separate functions interpret data from the sensors. Arduino is an open source platform with documentation and programing "libraries" of useful functions readily available for work with a wide variety of sensors. While we decided to focus the project on automation and systems, it should be mentioned that optimizing I2C, analog, and digital connections offers a project extension into signal processing.

Once the ASV's location is determined from the GPS, the control program compares this position to the next desired position for a desired heading calculation. That heading is compared to the corrected heading from the LSM303 to determine a required course change. If no course change is necessary, the ASV will simply trim its wing-sail and continue on.

If a turn is required, the control code references the anemometer's input to determine what type of turn is appropriate. Sailing vessels cannot sail directly into the wind but must sail at angles for their sail or wing to provide the desired, driving force. Our algorithm balances maximizing the distance sailed in the correct direction with minimizing the number and complexity of sailing maneuvers if the desired heading is too close to the wind. Any turn may require one of two different sailing maneuvers (turn type), as shown in Figure 6.

After determining the type of turn, the corresponding turn algorithm adjusts both the rudder and wing-sail angles to complete the maneuver. A discrete-time, proportional controller was used for the rudder. The apparent wind angle was mapped to the position of the wing-sail using a linear relationship.

Occasionally, sailing vessels will get stuck in a position where the boat is pointing directly into the wind. Since the sail or wing does not provide forward motion in this scenario, normal control inputs do not work. We created a specific algorithm (the "irons" routine) to detect and escape this situation and return the vessel to a sailable course.

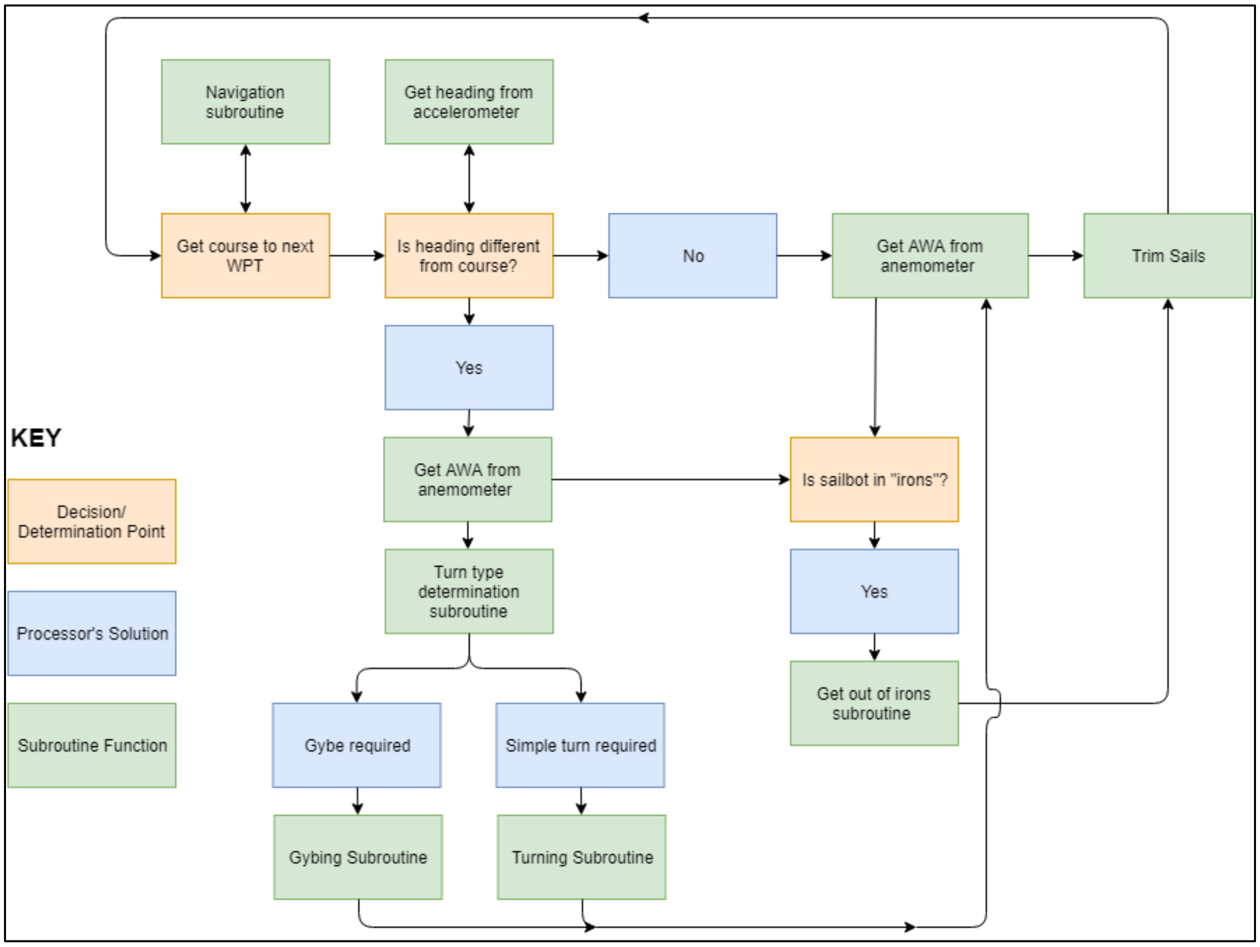

Figure 6 Flowchart of the ASV control as of December 2019

Results

We conducted two “field” tests with the ASV control system in 2018, utilizing the mock-up mounted on a wheeled cart. Both tests put the control system through turns, simulated wind shifts, waypoint increments, and desired course changes.

The test results validated the logic for all aspects of the control algorithm but often the response came far too slowly. Often, the rudder would move too little then pause for up to 30 seconds and overcorrect.

Troubleshooting showed that the slow execution was the result of sequential processing of the complex computations, calculations, and data processing for the navigation subroutines. Code for parsing and interpreting the GPS and LSM303 inputs, and a navigation function involving many trigonometric computations, are a substantial workload for the Arduino’s single core processor. The control code by itself occupies over half of the Arduino’s useable program memory.

Modified Control Code

Implementation of Protothreads

To address issues with the execution speed of the control program, we have implemented protothreads to mimic parallel processing on the Arduino’s single core processor. This technique pulls certain tasks from the control code, and still executes them sequentially, but in between the execution of other tasks. Arduino’s “TimedAction” library provides a user-friendly approach for proto-threading portions of Arduino code and defining the time interval over which the threat is executed.

Not everything needs to be proto-threaded as we still want certain actions to occur in a specified order. Functions for reading the anemometer, GPS, and LSM303 inputs, and the navigation function, have been proto-threaded. The threads update global variables, accessible anywhere in the program, for wind angle, position, desired heading, and heading. The major advantage to proto-threading these complicated functions is that it prevents their unnecessary, and slow, execution in every iteration of the control code’s loop. Instead, required variables are updated on frequency intervals that make sense, and that do not hamper control code execution speed.

Code Consolidation

The initial structure of the control code was bulky and hardly efficient. Through subsequent developments to the control code, we have eliminated calls to functions which use loops for specific turn types. Instead, after the turn type determination is made, the required course change and rudder angle are determined and executed. The angle of the wing-sail is then adjusted for the current wind angle, and the control code moves onto its next iteration.

Field testing has illustrated that this method provides the same basic responses as the specific turning functions. With the bulky turning functions eliminated, and the control code simplified, execution becomes faster and more seamless.

The implementation of proto-threading and code consolidation has had a noticeable improvement on system response times. Field testing now verifies minimal lag in responses (approximately 500ms) for both rudder and wing-sail movement in response to stimuli.

Controller Development

Since the wing-sail position is directly controlled by a servo, its simple mapping of apparent wind angle to desired wing position is appropriate and allows for optimum aerodynamic efficiency.

The same is not true of the rudder. The simplest type of control is a proportional controller, which sets rudder angle based solely on the magnitude of the course deviation. Proportional control is unresponsive to different rates of turn and causes practical issues with the ASV overshooting and oscillating around its desired course.

Instead, the rudder requires a proportional-integral-derivative (PID) controller. The proportional part of the rudder's PID controller still responds to the difference between current and desired vessel heading. The integral term takes previous heading differences into account. For example, if a rudder angle has not produced any course change, the integral term may increase the rudder angle until the appropriate response is achieved. The derivative term corrects for the heading difference's instantaneous rate of change, preventing overshoot and oscillation around the desired course.

Creating a PID controller through Arduino code is a challenge. The Arduino forces a discrete time system. Instead of a traditional, continuous-time PID controller, we are required to use difference equation estimations of the integral and derivative terms [7] (Equations 1 and 2). These difference equations are realizable with Arduino code.

$$y(t) = \frac{dx(t)}{dt} \approx y[n] = \frac{1}{T}(x[n] - x[n-1]) \qquad (1)$$

$$y(t) = \int_{-\infty}^{t} x(\tau)d\tau \approx y[n] - y[n-1] = Tx[n] \qquad (2)$$

T is the sampling period of the heading sensor, $y(t)$, is the rudder angle, and $x(t)$ is the heading difference. $x[n]$ and $y[n]$ are the discrete instances of these functions as represented on the Arduino.

As a clear and useful extension into control systems modeling, we intend to use Simulink and MATLAB software (Figure 7) to help optimize our discrete-time PID controller's coefficients as we gather experimental data on the ASV's heading controller via on the water testing.

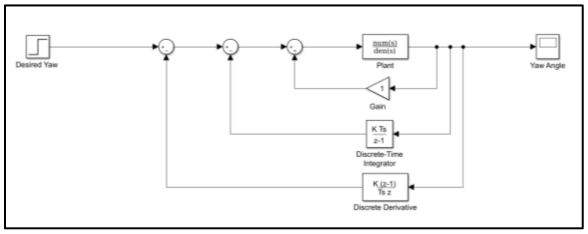

Figure 7 Early Simulink model of PID rudder controller

Power System Design

In addition to the control system, we had to design a solar rechargeable power system capable of sustaining ASV control system operations for the duration of a trans-Atlantic voyage. Our search for power system components was driven by power-budget calculations, which consider the expected loads, frequency of operation, and current draw of every component on the vessel. Our power system design consists of a Richsolar, 30 Watt, 12V, monocrystalline solar panel, a Genasun GV-5, maximum power point tracking (MPPT), custom solar charge converter, and five Bioennopower, 11.1V, 5 amp-hour batteries. A preliminary system block diagram is shown in Figure 9.

We will use measurements during actual operating conditions to verify the capacity of our designed power system prior to installation in the ASV.

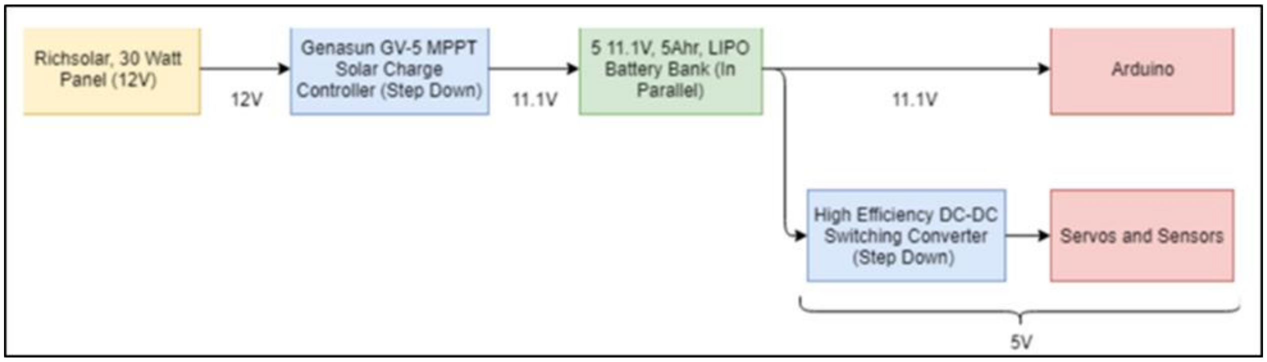
Richsolar, 30 Watt Panel (12V)
12V
Genasun GV-5 MPPT Solar Charge Controller (Step Down)
11.1V
5 11.1V, 5Ahr, LIPO Battery Bank (In Parallel)
11.1V
Arduino
High Efficiency DC-DC Switching Converter (Step Down)
Servos and Sensors
5V

Figure 9 Power system block diagram

Conclusions

Future Work

This is an accessible project that has components requiring several engineering disciplines. Subsequent semesters of our project work may choose to examine signal processing and filtering of signals from the ASV's sensors, applying different modelling techniques to optimize controllers, and implementing different algorithms for craft control. Currently, the LSM303's accelerometer is only utilized for magnetometer corrections. Future controller iterations may combine acceleration with GPS and compass data to optimize rudder control.

Application to Engineering Education

Few projects offer so much potential at the undergraduate level. The project incorporates creative design processes, troubleshooting, and innovative solutions to multidisciplinary challenges. While the Coast Guard Academy has approached ASV development through its Electrical Engineering and Naval Architecture Departments, other programs may adopt ASV development as Mechanical, Systems, or Controls Engineering projects.

Compared to many collegiate research projects, ASV development is very accessible. The project's cooperative and cross-disciplinary nature contributes to self-paced student work, study, and discovery. Most programs can put forward the modest budget needed for the project. To date our power and control system components have totaled less than $1000. The "MaxiMOOP" hull our Naval Architecture Department is constructing for the ASV was designed to be a cost affordable platform and plans are available online (the hyperlink is halfway down the page and is in blue text reading "file") [8].

References

1. Ding, Song, Han Duanfeng, and Boshi Zhang: Impact of Automation to Maritime Technology. International Conference on Computer and Information Application (2012).
2. The Micotransat Challenge. About the Microtransat. Retrieved from The Micotransat Challenge: https://www.microtransat.org/. Accessed 06 December 2018.
3. Holtrop, John. Rigid Wing Sails. Retrieved from Epoxyworks: https://www.epoxyworks.com/index.php/rigid-wing-sails/. Accessed 25 March 2019.
4. Miller, P., et. al: An Alternative Wing Sail Concept for Small Autonomous Sailing Craft. United States Naval Academy (2017).
5. Cactus.io. How to Hookup Davis Anemometer to Arduino. Retrieved from cactus.io: http://cactus.io/hookups/weather/anemometer/davis/hookup-arduino-to-davis-anemometer. Accessed 04 December 2018.
6. Arduino. How to Interface GPS Module (NEO-6m) with Arduino. Retrieved from Arduino Project Hub: https://create.arduino.cc/projecthub/ruchir1674/how-to-interface-gps-module-neo-6m-with-arduino-8f90ad. Accessed 04 December 2018.
7. B.P. Lathi and Roger Green. Linear Systems and Signals. Oxford University Press (2018).
8. Sailbot. MaxiMOOP. Retrieved from: https://www.sailbot.org/maximoop/. Accessed 23 February 2019.