5-2019

Autonomous Boat Control Software Design Using Model-Based Systems Engineering

Noah Nelson

Autonomous Boat Control Software Design Using Model-Based Systems Engineering

Noah Nelson

Bachelor of Science in Electrical Engineering Honors Thesis

University of Arkansas-Fayetteville

April 2019

Statement of Originality

I, Noah Nelson, the author of the paper entitled “Autonomous Boat Control Software Design Using Model-Based Systems Engineering”, do hereby certify that this paper is my original work except where due reference is made.

Acknowledgements

Thanks to Aleczander Jackson of No Magic, Inc. for providing resources and guidance in the use of Cameo Systems Modeler and also for organizing parts of the model when necessary; Liam Carr, Justin Jackson, and Sloan Becker for working alongside the author as part of the Boat Hardware Control Platform Team to develop the overall autonomous boat system that the control software discussed in this paper is a part of; Nicholas Blair, Baker Jones, Kaleb Crow, Alex Cutsinger, Brian Jones, Daniel Hartley, Kendrick Steven, Jason Dixon, John Jordan, and Michael Pollard for their prior work on the autonomous boat system project, upon which much of the current work on the project was built; Robert Saunders for providing advice, guidance, and mentorship to the team; Daniel Decipulo and Dr. Matthew Patitz, for providing assistance in setting up and configuring the boat’s software interface.

Abstract

While there is considerable buzz about self-driving cars, self-driving boats are actually more fully developed. The Boat Hardware Control Platform Team was tasked with developing a fleet of small autonomous boats that travel to a destination while avoiding obstacles and staying in formation. The author's specific task was to develop software used by the boats to detect obstacles and plan a route to a destination. This was done using a method inspired by self-driving cars, which shows promise, but is still being tested at the time of writing. The entire project incorporated model-based systems engineering, which proved to be useful.

Introduction

While autonomous cars figure prominently in the press, the technology behind autonomous boats is actually more fully developed [1]. Small boats used to gather research data and tugboat-sized craft used to haul devices out to the water are already in use [2], and autonomous boats for other purposes are currently in development [1] [2]. One of these other purposes is the purpose of military use; the U.S. Navy is developing a fleet of small autonomous boats designed to protect larger ships, harbors, and other maritime property, in part by "swarming" enemy boats [3]. Boats like these can also be useful for training purposes, allowing larger ships to practice the tactics they might use against smaller boats and even swarms of boats that the enemy might use against them [3]. As such, the Boat Hardware Control Platform Team (consisting of Justin Jackson, Liam Carr, Sloan Becker, and the author) was formed to continue the development of a fleet of autonomous boats capable of navigating to a target destination while avoiding obstacles and staying in a formation. (The work of this team was to build on work completed on the fleet during the previous two years.) The main focus of this paper, however, is on the author's part of this project, which is the development of part of the software for this system; more specifically, the development of the portion of the software responsible for the boats' route planning and obstacle detection. This paper will also focus on the use of model-based systems engineering through the Cameo Systems Modeler software application [4] to design the overall system.

Scope of Work, Hardware Design, and Introduction to Model-Based Systems Engineering

This paper focuses on the development of software on a single microcontroller that is part of a larger boat control system to be fitted onto commercially available radio-controlled boats. This and all other development on the boat control system was done using model-based systems engineering, which is the design of a system through the development of a "model" consisting of interrelated project requirements, subsystems, components, behaviors, and other elements necessary to describe the system in detail [4]. Cameo Systems Modeler is a computer program that aids in this process by allowing the easy creation and relation of model elements through the act of drawing diagrams that describe the elements and their relationships [4]. The architecture and overall structure of the boat control system was developed by the Boat Hardware Control Platform Team as a whole using Cameo Systems Modeler and the work of previous teams before any software design took place. This was done by creating block definition diagrams (BDDs), which describe the subsystems that comprise the overall system (a.k.a. "blocks") and the components of those subsystems (which can also be blocks), and internal block diagrams (IBDs), which describe how the subsystems within a system communicate and interact with each other [4]. More specifically, the former diagrams consist of multiple blocks connected by lines that show which blocks compose which other blocks, and the latter diagrams consist of multiple "parts" (subsystems and components) connected by lines marked with directional arrows and labeled with the type of information, item, etc. that flows through each line [4]. Fig. 1 is a BDD that outlines the major subsystems that compose the boat control system and shows the components of some of those subsystems. The "power supply" delivers power to all the other components of the boat control system. The "sensor board" is responsible for directly

interfacing with the rudder and motor attached to the boat's body, along with the gyroscope module, accelerometer module, and GPS module that yield information about the boat's position. The "daughter board", which is the focus of this paper, handles route planning, as well as obstacle detection through the attached Pixy cameras. It receives commands wirelessly from an external computer program, which are requests for the boat to either provide its current GPS coordinates or to navigate to a target destination. Once the daughter board receives a command, it requests the data it needs to properly execute the command from the sensor board, then either transmits the boat's current coordinates back to the external computer or calculates a route through detected obstacles and sends this route to the sensor board, depending on what type of command was received. Each of the sensor board and the daughter board has a microcontroller at its heart that manages communication between components and performs calculations. The purpose of the software on the daughter board's microcontroller, then, is to receive and process commands from the external computer, retrieve necessary information from the sensors attached to the sensor board, detect obstacles using the attached cameras, and plan a route between those obstacles to a given destination.

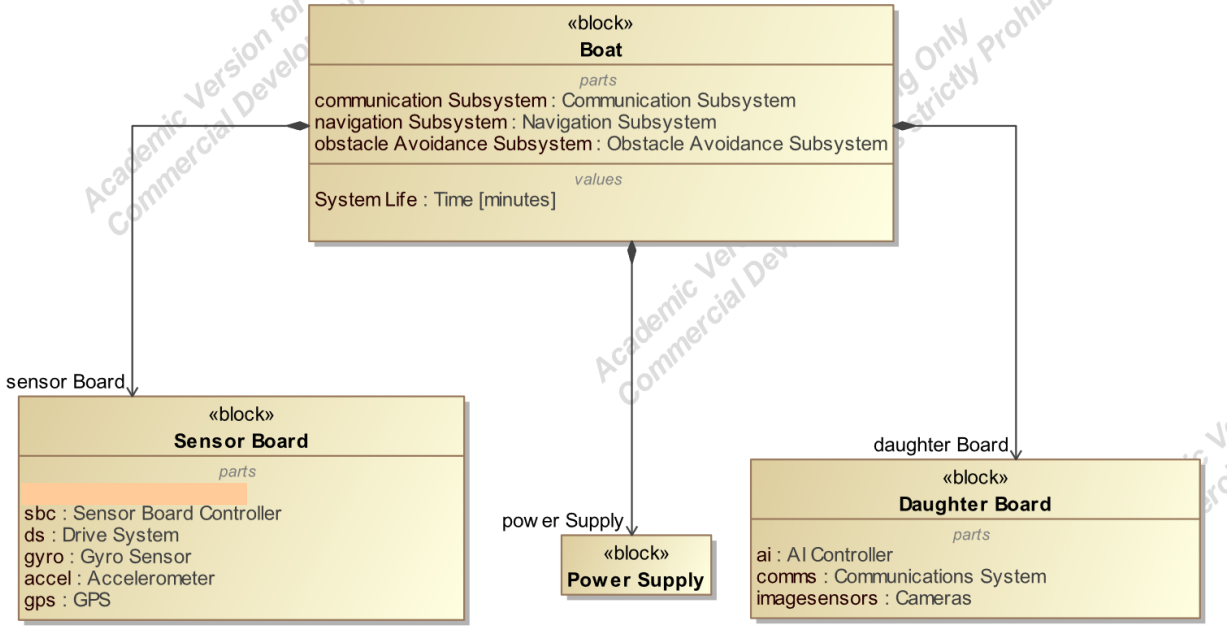

Figure 1:Block definition diagram of boat control system. (Developed by Boat Hardware Control Platform Team with assistance from Aleczander Jackson.)

Software Design Methodology

Once the physical structure of the boat control system was laid out, work could begin on developing the overall logic of the control software. This was done using the process of functional decomposition, which consists of breaking each of a system's main functions down into smaller and smaller functions [4] until the necessary actions the system must perform are clear. The process began with the development of a use-case diagram, which describes the different functions a system must perform in terms of how it interacts with its users and the environment (i.e. use cases) [4]. These use cases are then decomposed using activity diagrams, which show the order of functions used to complete the use cases and which part of the system is responsible for each function [4]. Some of these functions are then decomposed further using additional activity diagrams, as are some of the functions within those activity diagrams, and so on [4].

Each activity diagram begins with a large black dot that represents the start of the activity [4]. At least one dashed arrow leads out of this dot and to a chain of rounded rectangles linked by more dashed arrows, with each rounded rectangle representing an action that comprises the overall activity [4]. The chain continues until it reaches a large black dot surrounded by a white circle, which marks the end of the activity [4]. However, the overall chain may be interrupted by some additional features, such as diamonds that represent decisions and loops and thick black bars that split and join the overall control flow to denote actions completed in parallel [4]. There are also solid arrows that indicate data flows between ports attached to actions, as well as vertical columns or "swimlanes" into which actions are placed based on which part of the system needs to perform the activity [4]. Fig. 2 shows a sample activity diagram used as part of the overall model.

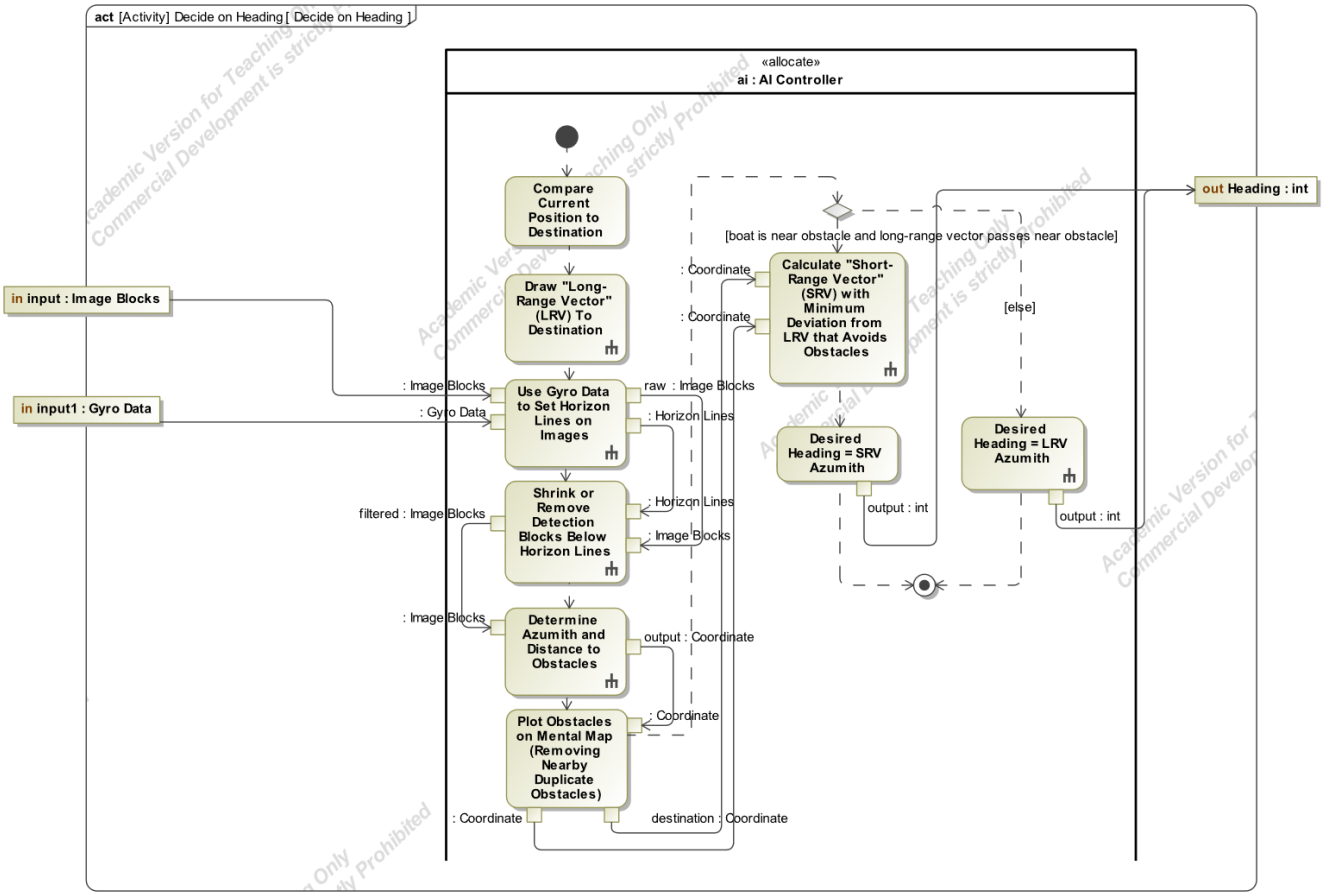

Figure 2: Example activity diagram.

The Route-Planning Algorithm

The general philosophy of the boat's route-planning algorithm is similar to the manner in which a self-driving car navigates; the boat keeps a "mental map" of its immediate surroundings and makes both a long-term plan of how to navigate to its destination and a short-term plan of how to avoid obstacles [5]. (Incidentally, navigation to a target destination and obstacle avoidance are the extent of the current design of the boat's route-planning algorithm, with the ability of the boat to stay in a formation to be developed later, once the other two functions are tested and verified to function properly.) In order to describe the algorithm in more detail, it is helpful to compare the boat's memory of the route and obstacles to a sort of "radar display" as depicted in Fig. 3. First, the daughter board retrieves the GPS coordinates from the sensor board and compares them to the destination coordinates transmitted to the daughter board via the external computer, using the information to calculate a "long-range heading" in the direction of the destination coordinates relative to the boat's heading (represented by the "O" on Fig. 3). Next, the Pixy cameras are used to detect obstacles on the water and measure their location relative to the boat. While the Pixy cameras' built-in image-detection capabilities, which are best at detecting brightly-colored objects [6], should be fairly accurate when detecting the bright yellow balls that will serve as obstacles, the reflection of the balls in the water may cause inaccuracy in the Pixy cameras' obstacle position measurements, which are based on the sizes of the obstacles the Pixy detects relative to the sizes of the overall images captured. To compensate for this, the daughter board retrieves gyroscope data consisting of the boat's current attitude from the sensor board and uses it to set "horizon lines" on the images. When the Pixy cameras detect objects, they draw "blocks" around the detected objects [7]; the

algorithm vertically shrinks and shifts blocks that cross the horizon line in an effort to "cut off" the portions of the blocks that lie below the horizon line, and it also deletes blocks that lie entirely below the horizon line. Once the water reflections in the images are thus dealt with, the horizontal positions of the blocks in each image are used to determine the angle of the obstacles relative to the boat (by scaling the blocks' horizontal position in the frame to the camera's field of view), and the width of the blocks are used in the following formula to determine the distance between the obstacles and the boat:

$$D = (W * F)/P \quad (1)$$

, where W is the actual width of the obstacle (known beforehand), F is the focal length of the camera lens (fixed [8] and determined empirically by applying Equation 1 to an arbitrary object with a known size and distance from the Pixy camera [9]), and P is the width of the block [9]. Ultimately, the boat plots the relative location of each obstacle it detects (represented by the "X"'s on Fig. 3). Then, if the boat would pass sufficiently close to an obstacle if it maintained its long-range heading, the boat calculates a "short-range heading" that would result in the boat avoiding the obstacles while deviating as little as possible from its long-range heading. Finally, the daughter board instructs the sensor board to turn the boat to the short-range heading if it was calculated and to turn the boat to the long-range heading otherwise. This overall cycle of taking inventory of the boat's current surroundings and planning a route based on the information gathered is repeated at a rapid pace until the boat reaches its current destination, at which point the daughter board instructs the sensor board to stop the boat's motor. In other words, as the boat moves and turns, the marks on Fig. 3 will also move, since their position relative to the boat is always changing. More specifically, the boat's motion will cause the marks to move so that the "O" mark representing the boat's destination tends to line up with the point of the boat's bow as shown on Fig. 3 (i.e. the boat is traveling in a straight line toward the destination), provided that no obstacles are in the way. If an "X" mark representing an obstacle does appear between the "O" mark and the point of the boat's bow, the boat will turn as little as possible to avoid the obstacle, eventually turning back to face the "O" mark once it passes the obstacle.

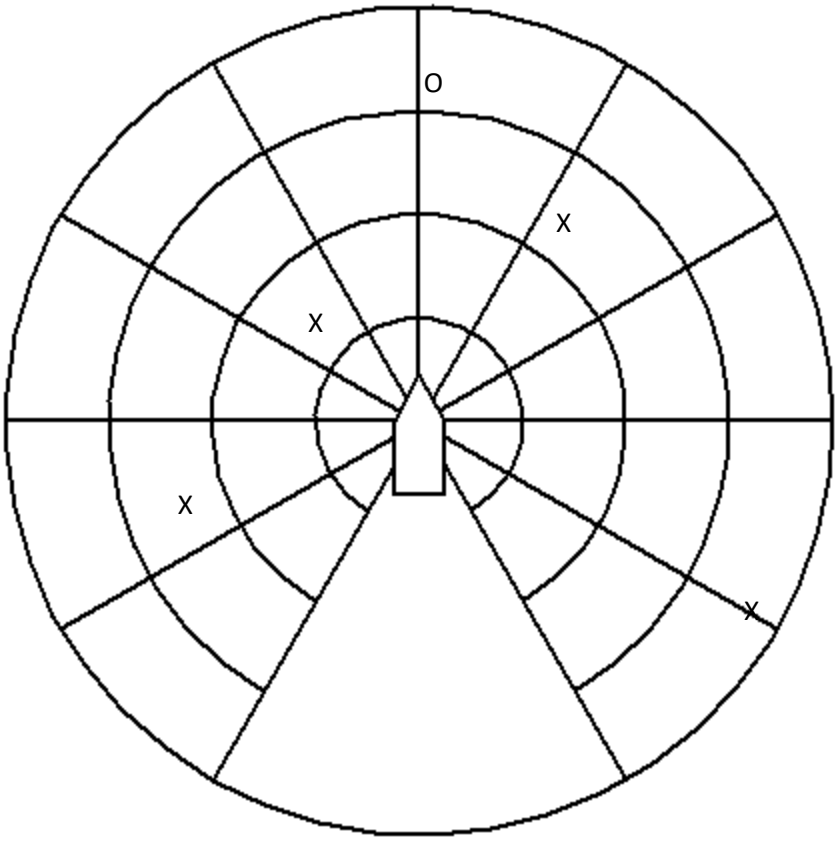

Figure 3:Visual "radar display" representation of boat's internal memory.

Conclusions and Future Work

An approach to route planning similar to that used by self-driving cars has potential for use in planning the navigation routes of each member of the fleet of self-driving boats being developed by the Boat Hardware Control Platform Team. However, testing this program is nearly impossible without hardware prototypes to load the program onto. These prototype circuit boards have been developed and are shown in Fig. 4 below as the green circuit boards near the center of the displayed boat control system assembly. The smaller daughter board sits on top of the larger sensor board, which supplies power to both boards and all other attached components via the nine-volt batteries. Each boat's motor and rudder are powered by a separate battery pack, but they will both connect to the sensor board for control. With this prototype complete and installed in a boat as shown in Figs. 5 and 6, development and testing of the boat control software can proceed. This process is currently near the end of its first stage, in which the software developed last year that enables the boat to navigate to a destination is modified to work with the new hardware and is tested for functionality. As such, work has begun on the second stage, in which obstacle avoidance functionality is added to the software in gradual steps, starting with rudimentary obstacle avoidance code and gradually developing it until it becomes the more sophisticated code outlined above. Only when both the first and second stages of development and testing are complete will the third stage, which is the development of fleet dynamics and coordination, begin. Ultimately, the entire implementation and testing of the boat hardware control system has been and will be informed by the various diagrams created earlier using model-based systems engineering. This system offers a clear methodology for developing the complicated system that is a fleet of autonomous boats by focusing on detailed description of the parts of the system and how they work together [4], and it will continue to be useful as development of the project continues through the implementation and testing stages.

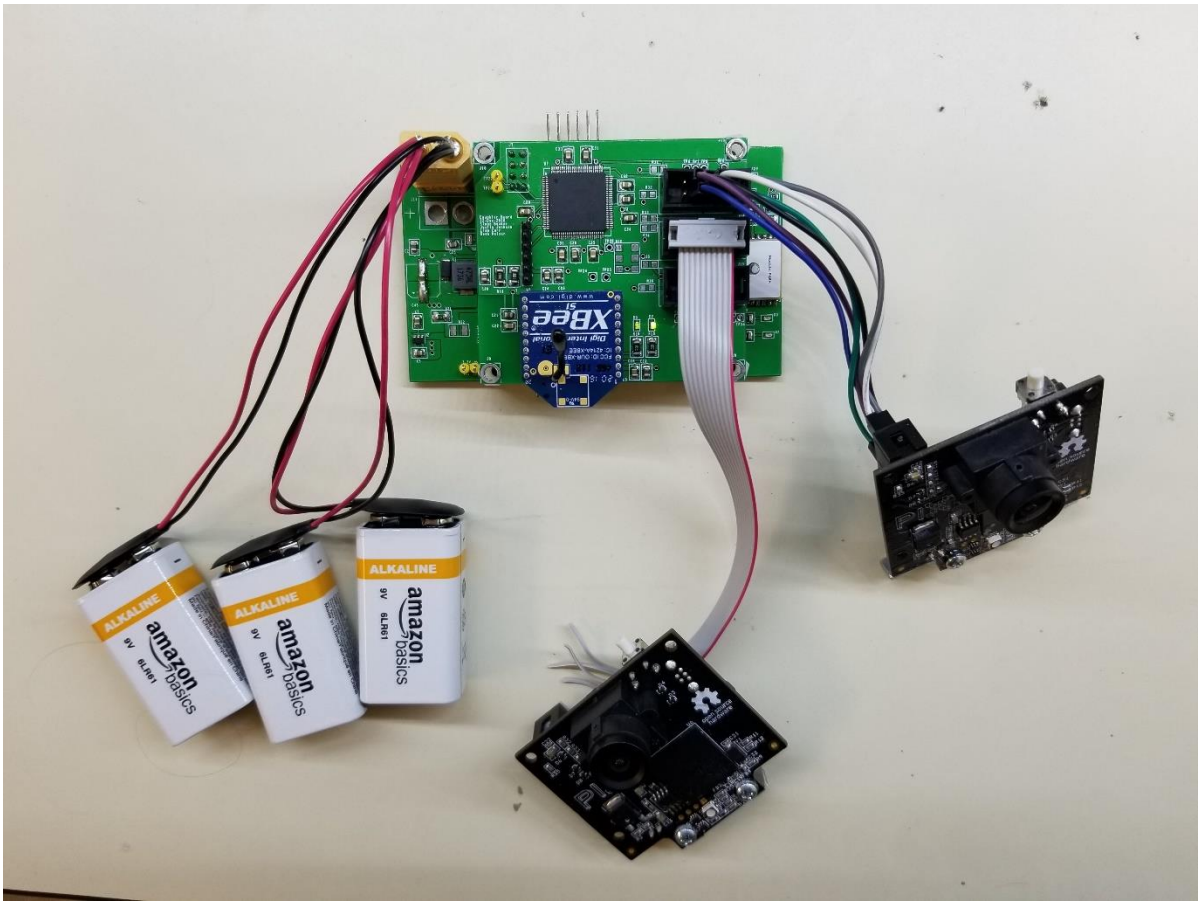

Figure 4: Assembled boat control system.

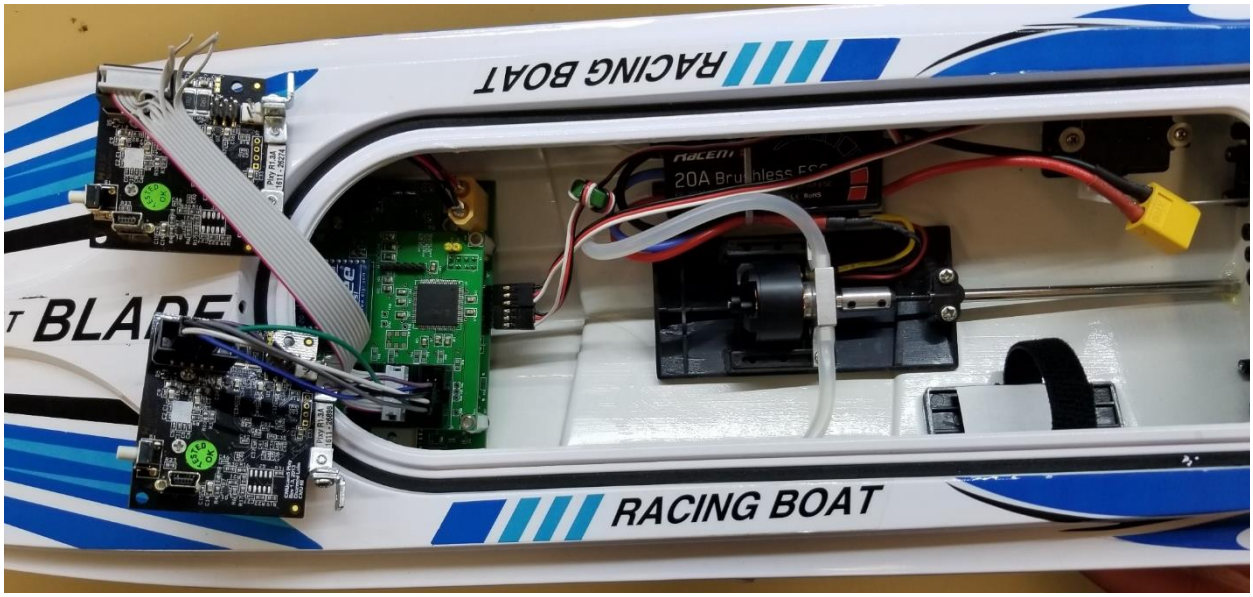

Figure 5: Boat control system installed inside of boat.

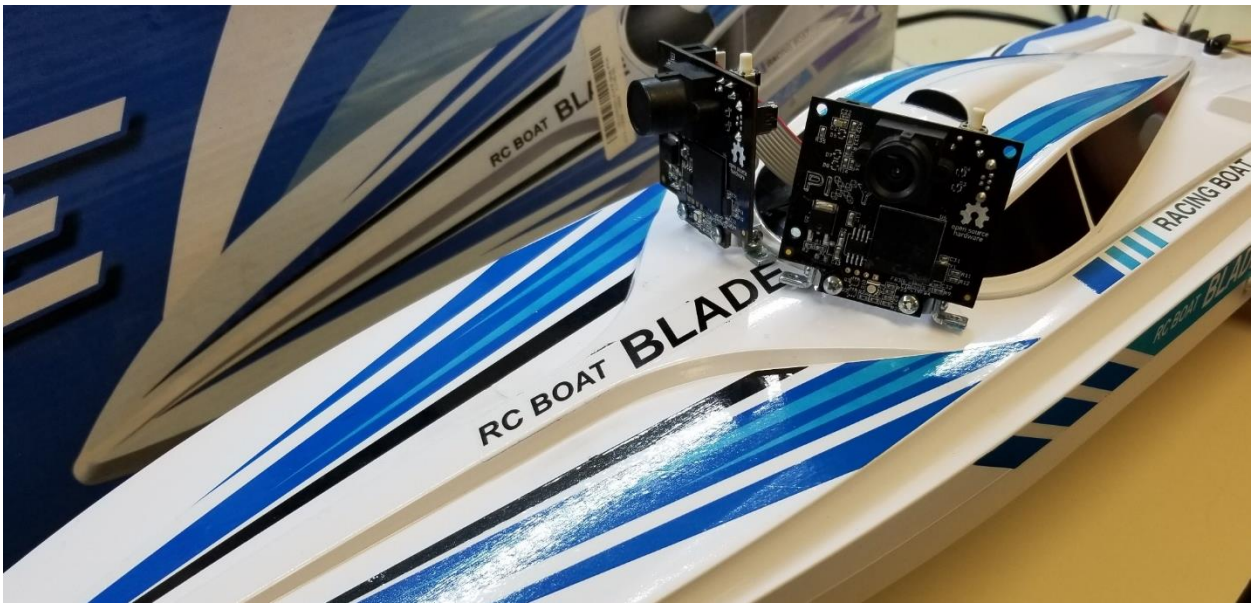

Figure 6: Boat with control system installed. (Camera cables are fed through a hole cut in the faux pilot cabin / access hatch.)

References

[1] T. Lindeman, "Autonomous Boats Will Be On the Market Sooner Than Self-Driving Cars," Vice Media LLC, 18 April 2018. [Online]. Available: https://motherboard.vice.com/en_us/article/ne95qm/autonomous-boats-will-be-here-before-self-driving-cars. [Accessed 4 November 2018].

[2] L. Mearian, "Autonomous boats -- yes, boats -- to hit the water in 2017," IDG Communications, Inc., 19 September 2016. [Online]. Available: https://www.computerworld.com/article/3121748/emerging-technology/autonomous-boats-yes-boats-to-hit-the-water-in-2017.html. [Accessed 4 November 2018].

[3] E. Adame, "The Navy's New Robot Boats Swarm the Enemy on Their Own," Condé Nast, 1 August 2017. [Online]. Available: https://www.wired.com/2017/01/navys-new-robot-boats-swarm-enemy/. [Accessed 4 November 2018].

[4] A. Aleksandraviciene and A. Morkevicius, MagicGrid(R) Book of Knowledge, Kaunas: Vitae Litera, 2018.

[5] S. Rayej, "How do self-driving cars work?," ROBOTS Association, 3 June 2014. [Online]. Available: https://robohub.org/how-do-self-driving-cars-work/. [Accessed 24 November 2018].

[6] Pixy Documentation Wiki Contributors, "Teach Pixy an Object," PixyCam, 24 January 2018. [Online]. Available: https://docs.pixycam.com/wiki/doku.php?id=wiki:v1:teach_pixy_an_object_2. [Accessed 24 November 2018].

[7] Pixy Documentation Wiki Contributors, "How to talk to Pixy," PixyCam, 12 January 2018. [Online]. Available: https://docs.pixycam.com/wiki/doku.php?id=wiki:v1:porting_guide. [Accessed 24 November 2018].

[8] D. Berkenfeld, D. Black, M. Corrado and L. Silverman, "Understanding Focal Length," Nikon Inc., 2012. [Online]. Available: https://www.nikonusa.com/en/learn-and-explore/a/tips-and-techniques/understanding-focal-length.html. [Accessed 25 November 2018].

[9] A. Rosebrock, "Find distance from camera to object/marker using Python and OpenCV," PyImageSearch, 19 January 2015. [Online]. Available: https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/. [Accessed 24 November 2018].

[10] J. P. Mueller and J. Cogswell, C++ All-In-One for Dummies, 2nd ed., Hoboken: John Wiley & Sons, Inc., 2009, pp. 293-298.