Clevis and Boathead Takes on the Charles

Design of Electromechanical Robotic Systems (2.017)
Final Report

May 17th, 2018

Dayna Erdmann
Ty Ingram
Ryan Koeppen
Bailey Tregoning

Abstract

Over the course of the semester, team Clevis and Boathead was tasked with designing and developing an autonomous boat that could complete three different tasks in the Charles River. These three different tasks were: 1) travel in a straight line at a constant heading and a constant speed, 2) travel in a closed circular path at constant speed with a radius centered around GPS point, and 3) travel autonomously approach a buoy, make a U-turn around the buoy and return to the same location along the dock.

By carefully developing the mechanical, electronic, and control systems, we were able to successfully complete the first task, and complete the second task with some errors. We attempted to complete the third task, but due to time constraints, were unsuccessful. For the most part, the project was a success because of our sturdy mechanical design and secure electrical connections. Although not all of the tasks were completed, we also obtained valuable data from tasks 1-3, which will be analyzed at length in this document.

Acknowledgements

Clevis and Boathead would like to thank Dr. Consi and Dixia Fan for their help and guidance throughout the semester.

Table of Contents

Introduction

We were tasked with designing and developing an autonomous boat that could complete three different tasks in the Charles River. These three different tasks were: 1) travel in a straight line at a constant heading and a constant speed, 2) travel in a closed circular path at constant speed with a radius centered around GPS point, and 3) travel autonomously approach a buoy, make a U-turn around the buoy and return to the same location along the dock.

Mechanically, our boat adopted a simple design, which was facilitated by the components provided to us in a kit. The sensors we chose to aid us in navigation and maneuvering include a 9-axis IMU and a GPS. To estimate position and velocity of the boat, measurements from the GPS and IMU were combined in a Kalman filter algorithm. These state estimates were used both in the real time feedback control algorithms as well as for data logging to characterize the boat's performance of each task.

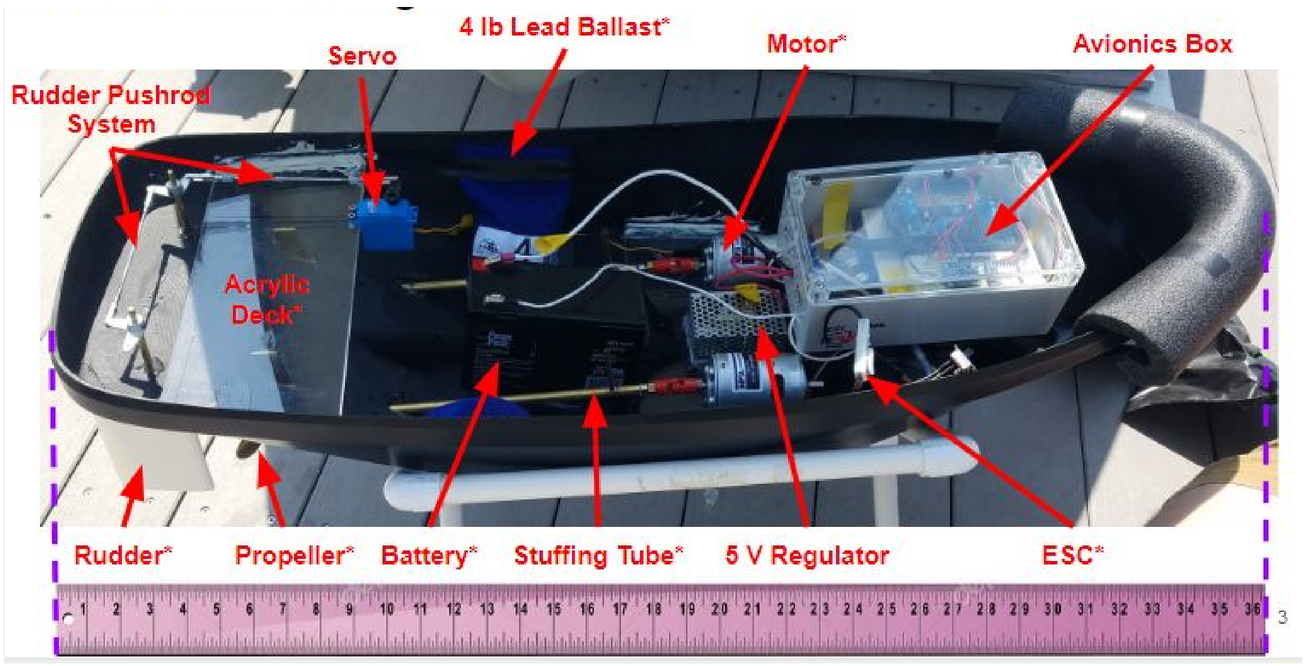


Figure 1. Photograph of our boat and all of its components. Later pictured: the boat with a trash bag on top, which helped prevent water from getting inside. It should be noted that the boat is three feet long. Asterisks in the labels denote that there are two of said component, but only one was called out for the sake of clarity in this diagram. The second copy of each component with an asterisk is positioned on the opposite side of the boat for proper balancing.

Pictured above (Figure 1) is an overall view of our boat and all of its components. We will go more in depth about each component in Section 3, Mechanics, but noteworthy overall components include our rudder actuation system, propulsion system, ballast methods, and electronics configuration.

There were several static and dynamic forces that we expected to act on our boat. We had to consider these in order to properly stabilize, calibrate, and maneuver our boat within our code.

Some of the static forces that we expected to act on our boat included gravity and the buoyancy force of our boat. Some of the dynamic forces that we expected to act on our boat included currents, wind, and waves in the Charles River. The boat also experienced a Munk moment due to the asymmetry of the stagnation points, which acted to turn the vehicle perpendicular to the flow. All of these forces worked against our boat, and had to be accounted for when implementing our feedback control.

Testing in a natural landscape such as the Charles River comes with its own set of challenges and uncertainties. The conditions and weather can vary minute-to-minute, so our mechanical components, sensors, and algorithms needed to be robust enough to account for these potential extremes and variations. More specifically, heavy rain and wind, strong currents, substantial waves produced by larger boats, and aggressive geese were all potential hazards that we had to account for. When we were testing, there were strong winds, which made the waves larger. As a results, we had waterproofing concerns (several times, waves crashed over the boat), but these were mitigated using a double-ply trash bag sealed with duct tape. The waves also made controlling the boat more difficult, as it caused strain on the rudders, causing the boat to sail off course.

Team Organization

In order to deliver on time and to specification, it was important that our team develop an organized plan of attack. We started by creating a Gantt chart for each subtask of each subsystem (mechanical, electronic, and control systems). By outlining a general timeline and milestone dates, our team was kept on track and working at an appropriate pace to deliver on time.

For the division of labor, Dayna and Bailey were the mainly in charge of the mechanical structuring and components of the boat, Ryan was mainly in charge of the electronics and sensors algorithms associated with the boat, and Ty was mainly in charge of the control systems and algorithms necessary for us to complete our tasks.

This division of labor stayed rather constant throughout the semester. If anything changed organizationally, it was when we accomplished certain benchmarks; some of these subtasks were accomplished a few days later than we had originally planned for. However, because we planned time for unforseen hitches, pushing back the completion date for some of these subtasks was acceptable and did not interfere with the completion of our end goals. By setting the benchmark and subtask dates for our Gantt chart during two separate "sprints," we were able to avoid a large pushback or propagation of error because we did not unrealistically set all of our dates seven weeks out, allowing less time for all dates to be pushed back.

Mechanical Design

Our team chose to keep the mechanical design of our vessel simple and compatible with the kit materials we were given in the interest of time. We used two each of: propellers, motors, 12V batteries, stuffing tubes, and rudders. We had two decks: one on which the motors sat, and one that the waterproof servo controlling the rudders was attached to. For ballast, we used small

4-lb bags of lead pellets, and for waterproofing, we had a small plastic avionics box that could be screwed shut, and covered the boat with a trash bag sealed with duct tape while testing.

Rudder Assembly and Struts

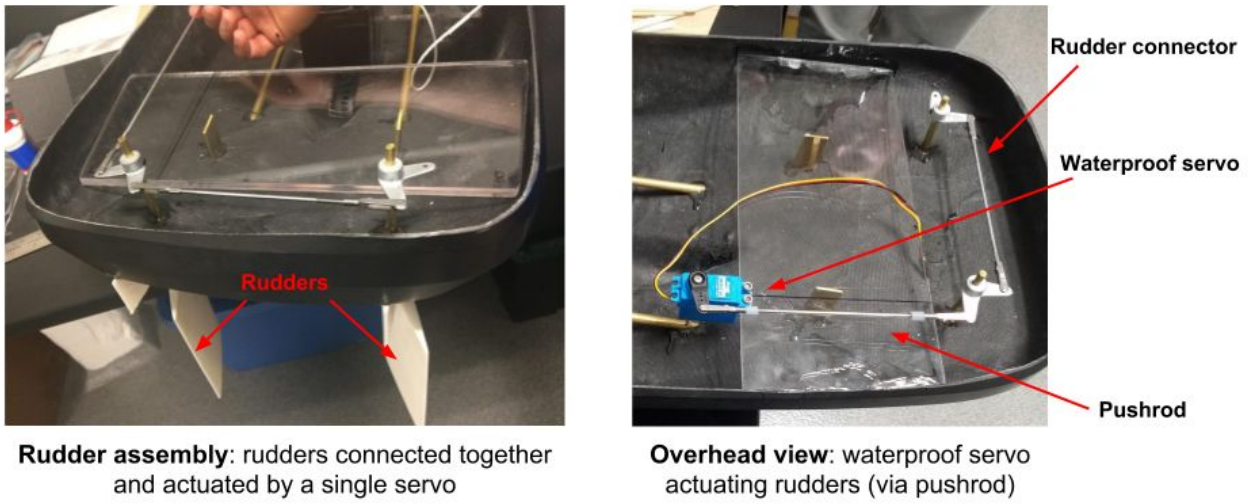

Rudders
Rudder assembly: rudders connected together and actuated by a single servo
Rudder connector
Waterproof servo
Pushrod
Overhead view: waterproof servo actuating rudders (via pushrod)

Figure 2. Snapshots of the fixation of the rudders to the hull and the actuation of the rudders by the servo.

Starting with the back of our boat, we installed the fixed rudder pipe tubes to be the only cuts made in the back of the boat (Figure 2, right). This way, the projected area cut out to the hull is smaller (a small pipe rather than the large, oblong contour of the rudder). Less area cut out of the hull means there is less of an opportunity for water to leak into the boat. The fixed rudders were attached with epoxy, and later reinforced with more epoxy.

Both rudders were controlled by a single servo, which allowed for extra consistency in both rudders. The rudder coupling system utilizes two pushrods to connect the two rudders to be geometrically coupled (Figure 2, left). The use of four clevises facilitates connecting the servo horn, to the rudder arm of the first rudder, to the rudder arm of the second rudder.

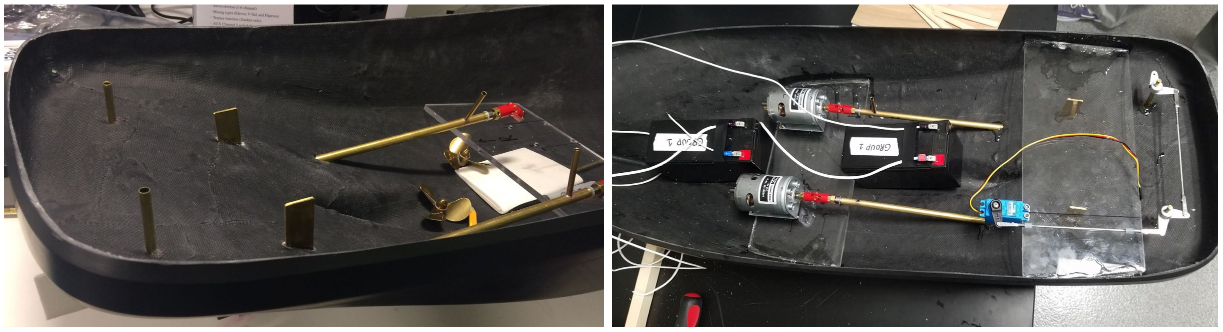


Figure 3. Snapshots of the strut spacing and overview of the lower and upper decks.

As pictured above in Figure 3, the struts were placed a reasonable distance away, as to properly space the propellers close enough to the rudders and close enough for the connectors for optimal motor turning.

DC Motors and Propellers

Also pictured in Figure 3 on the left, our team elected to use two motors and no bow thruster. We did not think the bow thruster would be necessary, and we believed that we could accomplish the kind of stability and maneuvering required for our three tasks via only two motors and two rudders. This picture on the left in Figure 3 also shows our lower and upper acrylic decks. The lower deck holds the two motors and 5V regulator (setup pictured previously in Figure 1), and the upper deck holds the servo actuation system (as shown in Figure 2, right panel). Both decks were laser cut and attached to the hull with epoxy. They had to be reinforced with flexible epoxy because as we moved the boat around, there was a creaking noise that suggested that there was strain between the platform and the fiberglass hull. This did not end up being a concern later on.

The motors were mounted to the acrylic platform via screws, but because of initial alignment issues that we had (the motor shafts were epoxied in but the connectors were jamming because the inner shaft was too far up), we had to drill new holes in order to place one of the motors further back. This was definitely one of the most tedious tasks of the project from the mechanical end. One of the motor oil tubes in the shaft fell off, so we used metal epoxy to reattach it. Additionally, once the motors were realigned, after testing the boat in the tank in the lab, we realized that one of the motors was spinning such that the propeller was being loosened, so we had to adjust that by switching the connections between the ESC and the motor and adding Loctite to ensure a strong bond.

Electronics

Schematic

Figure 4 shows a complete schematic of the electronic system used on our boat.

Control Computer

We used an Arduino AtMega 2560 to control the IMU, GPS and servo. We originally considered using an Arduino Uno as well, but found this unnecessary and stuck with a single microcontroller. The Arduino was housed in our avionics box, along with the rest of the sensors. Every time a task was attempted, either the new code was uploaded to the Arduino, or (in the case of simply attempting the task again without the need to modify the code) the reset button on the Arduino was pressed. Overall, the Arduino proved to be a simple and successful choice that interfaced well with the sensors.

Power System

Two 12V lead-acid batteries were connected in parallel to power the motors and ESCs.We used a fuse in between each ESC and the PWR side of the battery for protection. In addition, we had a power bus inside the avionics box that had all of the PWR connections on one side, and the GND

connections on the other so that they would not only be safely separated, but easily connected to the other components requiring power and everything that needed to be grounded.
A 5V regulator was used to convert the power from 12V to 5V for the servo controlling the rudders, the relay and the receiver. The batteries were connected to the power bus via crimp connections, which made them easy to remove. This proved useful for debugging (originally we had some smoke from the motor because of some less-than-ideal connections that were later cleaned up, and it was helpful to be able to easily remove the battery connections).

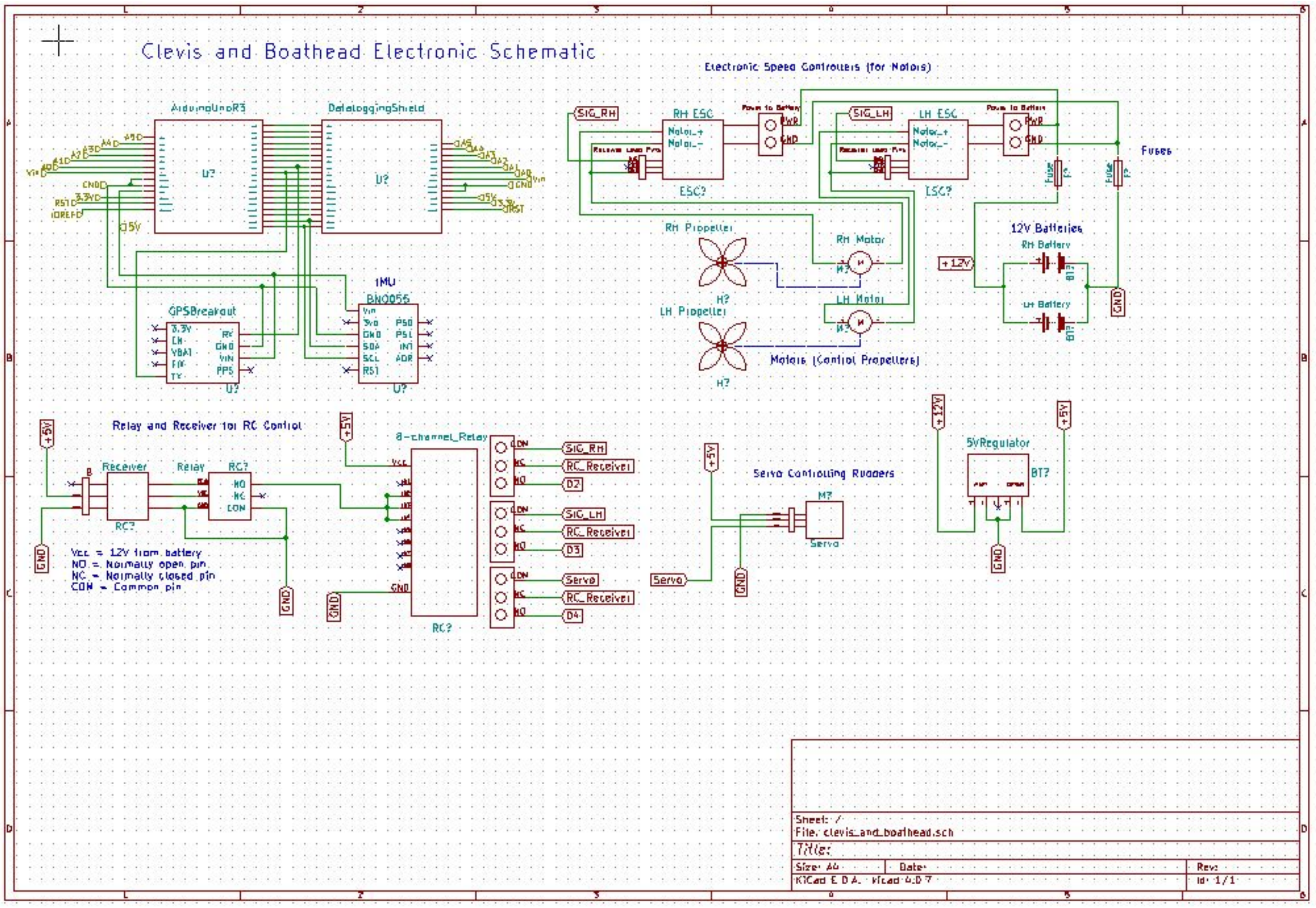


Figure 4. Schematic of electronic components, including the Arduino microcontroller, GPS, IMU, receiver and relay, 5V regulator, servo, ESCs, motors and batteries.

Actuators and Driver Circuits

We used a single waterproof servo to control both rudders, whose SIG line was connected to the COM line of the 8-channel relay. (see above schematic) The ESCs were connected to the motors and to the power bus, and their SIG lines were also connected to the relay. This allowed for easy simultaneous switching between autonomous control of the ESCs and servo and manual RC control.

Radio Control System

Radio-Controlled Relay for Backup Control of Boat

While the boat was designed to be controlled autonomously while performing tasks, it was necessary to include a safety mechanism to allow the autonomous control to be overridden by radio control. Because a docking procedure was not ultimately included in the control scheme for the boat, radio control is necessary to allow the boat to return to the dock without manually retrieving it from the middle of the Charles river. In addition, because the boat is operating in an environment with other aquatic vehicles and collisions are a possibility, a manual override feature is crucial for ensuring the safety of other vehicles and people on the water.

The relay circuit implemented in the boat used a Pololu RC Switch with Relay and SainSmart 8-Channel 5V Relay Module. By combining these components as shown above in Figure 4, relay switching of all actuators in the boat was controlled by toggling a switch on the radio controller. When the relay switch is turned off, all actuators receive command (PWM) signals from the radio receiver. When the relay is switched on, the actuators receive signals from the Arduino microcontroller to perform the desired autonomous task. The choice was made to make the boat radio controlled when the relay switch is turned off. This decision served as another safety precaution such that the boat could still receive radio signals in the event of a connection failure in the relay mechanism.

Algorithm Software

Algorithms for accomplishing tasks 1-3

Task 1 simply implemented the PID code setting the boat to stay at a constant speed and heading. The desired heading was chosen arbitrarily, the desired speed was 1 m/s as the controller design was linearized around this speed.

The algorithm for accomplishing task 2 worked by first having the boat drive 30 m away from the dock. Upon reaching this distance it set a center point a distance R directly perpendicular to its current heading. It then circled around this point. Originally this point was suppose to be set to be on the starboard side of the boat, but as we will discuss later is was moved to the port side. After setting this point the boat used an on-off controller to circle the center point it set, at a constant radius R. When the boats absolute distance from the center point was great than R the boat turned inward toward the center point. When the absolute distance to the center point was less than or equal to R the boat went straight. In the second iteration it turned slightly outward in this regime as we thought it might better maintain a constant radius this way. Figure 5 shows a simple state diagram for this algorithm. Figure 6 shows a visual representation of the algorithm.

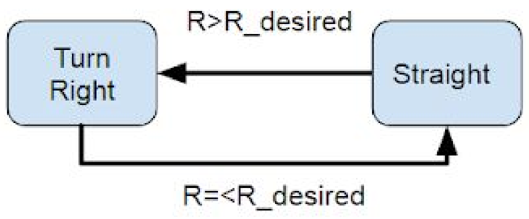


Figure 5: State space diagram for accomplishing task 2. Code worked like an off-on controller, sending the boat strait if it was in or on the circle and turned the boat inward if it was out of the circle. R is the distance of the boat to the circling point. R_Desired is the rotating radius.

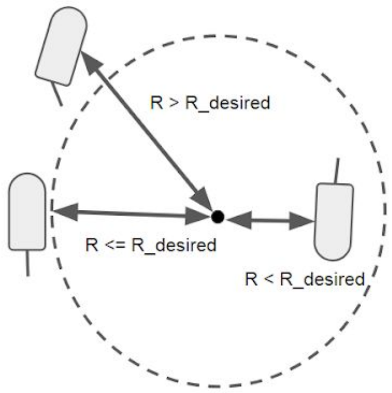


Figure 6: Visual representation of how the algorithm for task 2 functioned.

The algorithm for accomplishing task 3 involved two separate algorithms. The first part plotted a course around the buoy. Given the buoy's general location the algorithm set 4 points around the buoy in a 20 by 20 m square as well as a point at it's starting location. This would be the path for the boat to travel. The boat would go to each point of the square and then back to the starting position. Figure 7 depicts a visual representation of the algorithm.

The second algorithm for task 3 was used to get the boat to the desired points set by the the first algorithm for task 3. The desired heading could be calculated by taking the inv tangent of the slope between the desired point and the current location. The PID controller could then be used to achieve this desired heading. Once the boat had moved slightly the desired heading would be recalculated and plugged back into the PID controller. This process would continue such that boat should always want to point at the desired point. Eventually if the boat enter a zone close enough to the point, in this case a box 6m by 6m around the point it would check the point off and then proceed to the next point set by the first algorithm for task 3. This whole process is visually depicted in Figure 8.

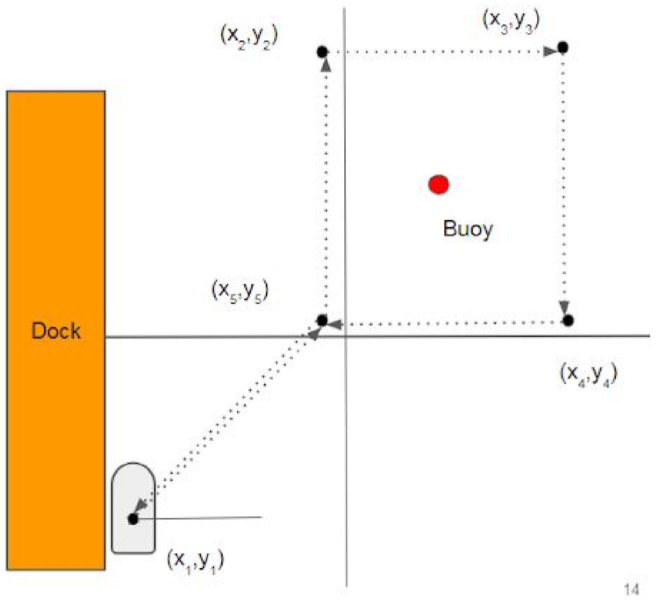


Figure 7: Above is a visual representation of the navigation algorithm for rounding the buoy in task 3. Given the general location of buoy the algorithm plotted a path with four way points around it.

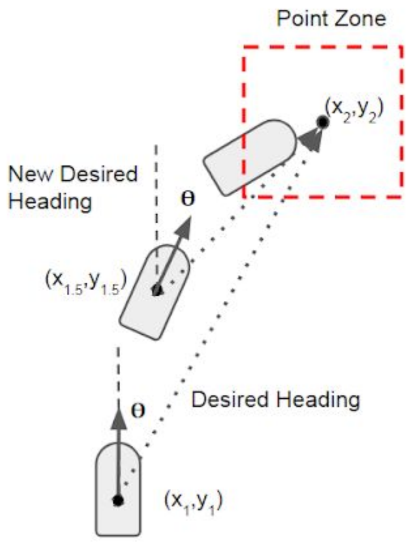


Figure 8: Above is a visual representation of the navigation algorithm for getting to a desired way point. Based on the position of the point and the boat position the desired heading was set. Once the boat enter the “goodzone”, was within a certain distance of the point, it checked the point off and proceeded to the next desired point.

One of the most significant issue that the algorithms ran into was that the GPS coordinate system and the compass heading did not line up in a standard manner. The GPS coordinate system had positive Y going North and positive X going East. Based on this and the right hand rule, East should have corresponded with 0 degrees and north with 90 degrees. In actuality North corresponded to 0 degrees and the degrees rotated clockwise such that East was 90 degrees. This meant that if the algorithm chose a desired heading of 20 degrees based on the algorithm described above for task 3, the actual desired heading needed to be 70 degrees. In the second iteration of the the algorithms for tasks 2 and 3 accounted for this angle shift. This is also why task 2 was changed to rotate to the left rather than right because this angle shift flipped the location of 90 degrees, since the heading rotated clockwise.

Control Systems Software

PID control

The speed and heading of the boat where controlled using feedback control systems. PID was chosen as it was the standard controller for physical system of this nature. Both systems required integral gain to deal with outside disturbances and derivative gain to smooth the oscillatory behavior introduced by integral gain. Proportional gain is needed for any control system. The diagram below in Figure 9 shows how the PID control was implemented in the code for speed control. As is standard of PID code there are 3 gains, Kp, Ki, and Kd corresponding to proportional, derivative and integral control. These gain are multiplied by the error, the sum of the error over time, and the instantaneous derivative of the error. The last two are found discreetly using a time step dt. These 3 terms are then summed to get the controller output which for speed was throttle angle and for heading was rudder angle. Green boxes show added features to the PID code. A smoothing filter was added on the derivative control as it was very susceptible to noise and had a high likelihood of spiking which would saturate the controller. Wind-up prevention was added to the integral control. This prevented it from becoming too large or building up in certain situations, such as if the boat was turned on but still sitting on the dock. Once all the control was summed it was limited to a range to prevent over saturating the controller. In the case of speed the throttle only went from 0 to 90, and for heading the rudder angle could only move a max of 30 degrees in either direction.

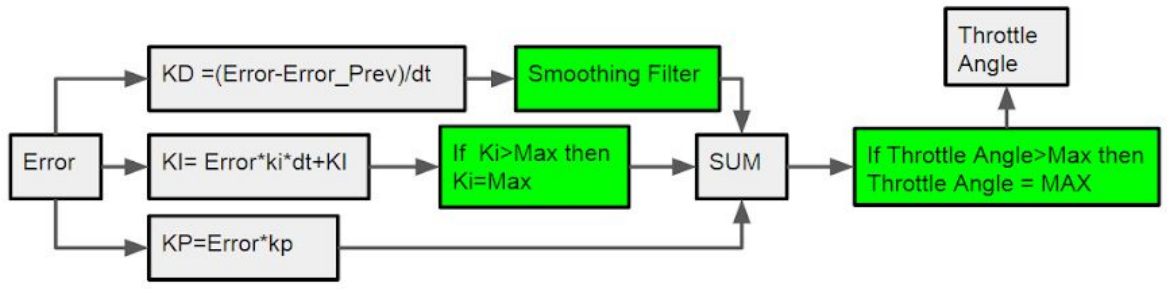


Figure 9: PID code structure for controlling speed. Green blocks indicate added features. Heading PID code worked in roughly the same manner

The PID code for heading worked almost exactly like the code for speed, except there was an additional feature added to correct the error. At first if the boat was headed at 270 deg and the desired heading was 0 this would output and error of -270.. This was fixed such that the boat never could have a error with an absolute value greater than 180 and would instead turn the opposite direction if this occurred. In this case the error would be 90 instead of -270.

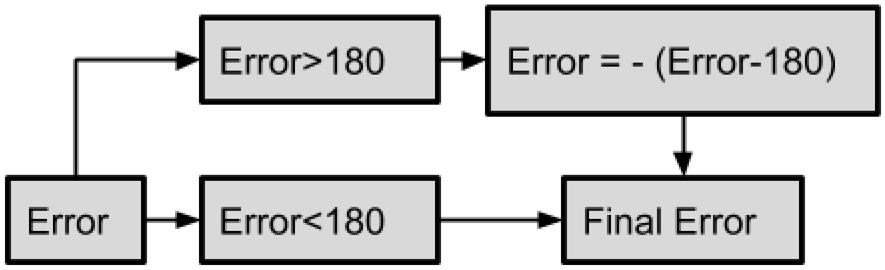


Figure 10: Structure of error correction code used in the heading PID controller. Code made sure the boat always turned in the direction of minimum rotation toward the desired heading

Simulink Modeling

In order to choose appropriate gains for the PID controller simulink models where created in matlab for the speed and heading of the boat. The models can be seen in the appendix. To create these simulations the transfer functions for the speed and heading of the boat where estimated. This involved making approximations for the drag force added mass in the forward or x direction and rotational drag, moment of inertia and added moment of inertia in the z direction. For this the boat was approximated as box with the dimensions of the boat. Obviously this lead to very rough estimations that overestimated drag force and added mass. To account for this both the factors were usually decreased by a factor of 30-50%. The drag force was proportional to speed squared or rotational speed squared which is nonlinear and can't be easily simulated. To account for this the drag terms were linearized around an equilibrium speed or equilibrium rotational rate. Using the mass and drag terms in a force balance and then taking the laplace transform the transfer functions for speed and heading were found.

Once the systems for the speed and heading had been estimated the relation between the controller and the physical control output also had to be estimated. For speed this involved relating throttle angle to thrust. To do this the boat was run at varying constant throttle ranging from 0 - 80 degs and the force of holding the boat in place was measured to get thrust. The data for this experiment can be been below. Based on this there was approximately 0.18N/deg of throttle.

For heading the rudder angle needed to be related to torque applied to the boat. To do this the lift force on the rudders and the distance of the rudders to the center of mass where approximated. The lift was approximated using the equation (i).

$$T = R(A\rho B v^2/2)\phi \qquad \text{(i)}$$

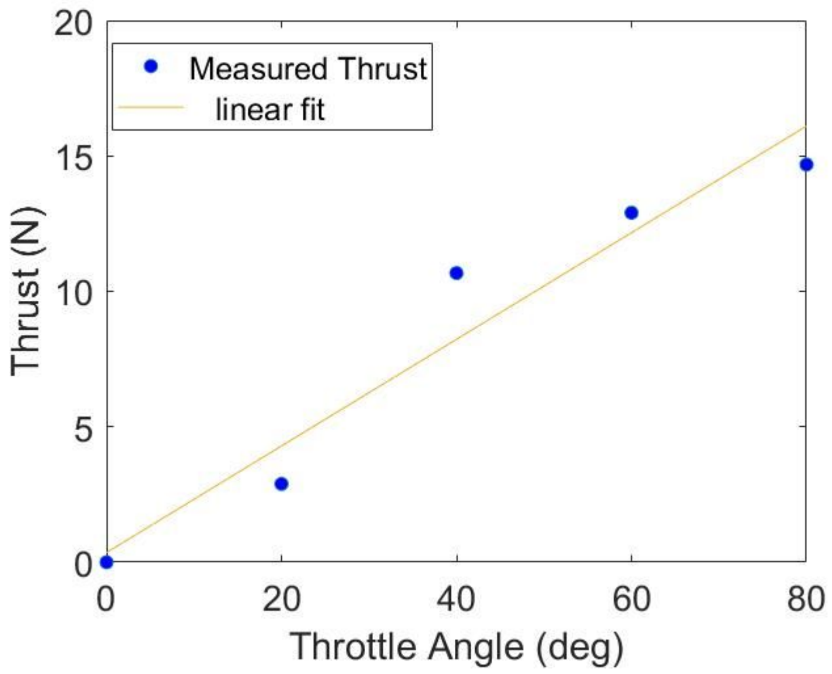


Figure 11: Test results measuring thrust at different throttle angles. This was used to relate control output to force on the boat for the simulink model and get gains of the correct magnitude.

T is the Torque from the rudders, R is the distance to the center of mass. A is the area of the rudder, ρ is the density of water, B is a constant, in this case, related to lift, and is the angle of the rudder. This formula is a valid approximation for lift force with small angles, B and Φ together account for the changing coefficient of lift.

The boat also experienced a munk moment from the immediate diagonal flow caused by the rudder. This munk moment was approximated using equation (ii) below.

$$T = \Phi(A_{22}-A_{11}) \qquad \text{(ii)}$$

Again T is the torque applied on the boat, in the same direction as the lift from the rudders, and Φ is the rudder angle. A_{22} is the estimated added mass in the Y direction and A_{11} is the added mass in the X direction of the boat.

Using the sum of the two terms above relation between rudder angle and torque was estimated. Both these equation where directly used in the simulink model to convert the controller output angle into applied torque on the boat. It should be noted that like in the code the output angle was bound in the simulation between -30 and 20 degrees. It was also converted to radians before being used in the above equations.

Once the transfer functions and controller output where estimated the simulink model where used to tune the PID gains such that the boat had acceptable responses to a step function and a step disturbance. Figure 12 show the best simulated response for speed and heading. Both systems where tuned to try get a settling time under 10s and an overshoot below 20%, while maintaining a high level of robustness. The gains used to achieve these responses were used in the PID code.

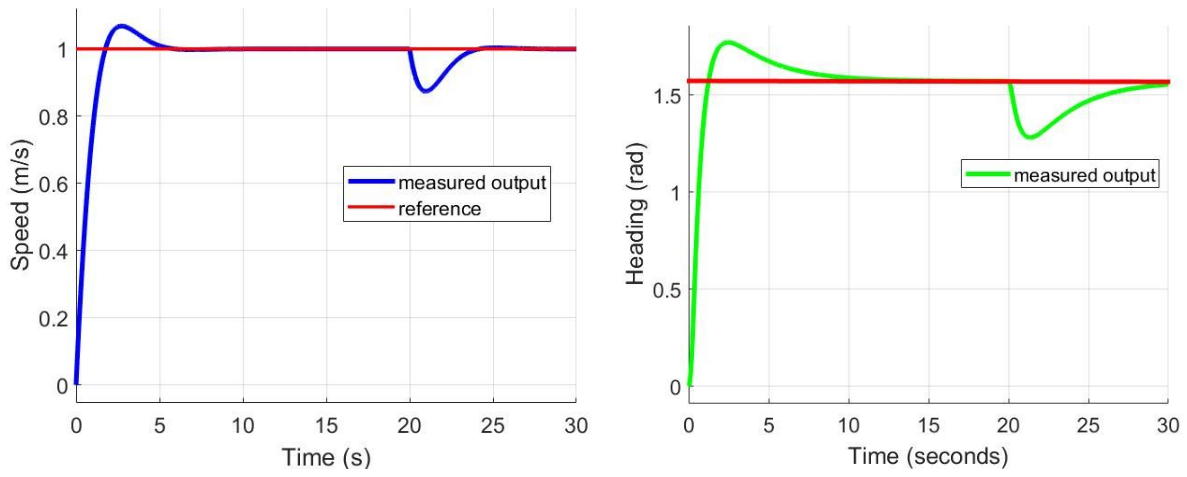


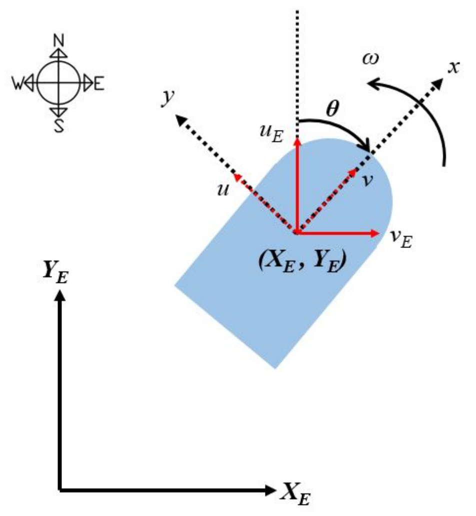
Figure 12: Tuned step response and disturbance rejection for the simulink model for speed, to the left, and heading, to the right. The gains were optimized to achieve low settling time, and low overshoot as well as overcome disturbances.

Sensor Data Acquisition and Processing Algorithms

State Estimation with GPS and Inertial Measurements

Feedback controllers that determined the motor and rudder angle commands relied on having a reasonably accurate estimate of the boat's degrees of freedom. Specifically, these controllers depended on some combination of the boat's speed, heading, and position (depending on the task). In addition, estimates of these degrees of freedom were necessary to quantify how well the boat performed the tasks it was executing. Therefore, the high-level strategy to controlling the boat involved making as few assumptions as possible about the boat's motion, relying on measurements from multiple inertial and position sensors for state estimation, and using these measurements for feedback control. With this strategy, potential disturbances to the boat's motion such as waves, wind, and currents can be corrected for with real time measurements.

To implement this state estimation, the boat was modeled as a two-dimensional rigid body with three defining degrees of freedom: the two translational degrees of freedom of the boat's center of mass relative to its position at the start of the task and the boat's heading defined as an angle relative to the Earth's magnetic north (see Figure 13). The feedback controllers required these states to be measured at a reasonably high sampling rate (to avoid degradation of controller performance) and with reasonable smoothness.



$$\vec{\boldsymbol{X}} = \begin{pmatrix} x_E \\ y_E \\ u_E \\ v_E \end{pmatrix}$$

$$\dot{\vec{\boldsymbol{X}}} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2\omega \\ 0 & 0 & -2\omega & 0 \end{pmatrix} \vec{\boldsymbol{X}} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \sin\theta & -\cos\theta \\ \cos\theta & \sin\theta \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix}$$

Figure 13: A schematic of the two-dimensional model of the boat. The diagram shows the kinematic variables considered in the model. The boat's velocity components u and v are relative to a coordinate system which is fixed to the moving boat (moving frame). The velocity components u_E and v_E are relative to a coordinate frame relative to the Earth with origin at the boat's initial position (E-frame). The boat's heading, θ, is defined relative to the Earth's magnetic north. The IMU collects linear (gravity-compensated) accelerations a_x and a_y are relative to the moving frame.

Two sensors were used to measure necessary quantities for defining the boat's degrees of freedom. The Adafruit BNO055 Absolute Orientation Sensor was used to collect inertial measurements such as acceleration, angular velocity, and angular orientation. This inertial measurement unit (IMU) also had on-board microprocessors which combined the raw sensor measurements and output highly accurate, fused orientation measurements as well as other compensated measurements. The Adafruit Ultimate GPS Breakout V3 was used to measure absolute position of the boat relative to the Earth. The sensor outputted standard latitude and longitude coordinates in degrees.

The first attempt to measure boat kinematics involved using the GPS sensor to define the two translational position states and using the IMU's fused sensor data to define the boat's heading. The GPS coordinates were converted to distance measurements by modeling the Earth as a sphere with a constant radius of 6371 kilometers (see Appendix A for details). Tests were performed early on to determine whether this combination of measurements would be acceptable to achieve the desired boat functionality. From this initial test, it was determined that while the GPS provided reasonably accurate position measurements, it also operated with a low sampling rate and was characterized by low resolution of approximately 3 to 5 meters. Additionally, meaningful velocity data could not be extracted from the measurements due to these limitations (see the right side of Figure 15).

These findings suggested that a data processing algorithm was needed that provided greater accuracy and incorporated measurements from multiple sensors. The next implementation of state estimation made use of inertial measurements provided by the IMU. A Kalman filter algorithm was developed which estimated the following four-dimensional state vector:

$$\vec{\boldsymbol{X}} = \begin{pmatrix} x_E \\ y_E \\ u_E \\ v_E \end{pmatrix} \qquad (1)$$

where x_E and y_E are the boat's position relative to its position at the start of the task, and u_E and v_E are the boat's velocity in the x- and y-directions as defined in Figure 13 relative to the fixed Earth frame. Based on our two-dimensional model of the boat, we developed the following state space representation of the system:

$$\dot{\vec{\boldsymbol{X}}} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2\omega \\ 0 & 0 & -2\omega & 0 \end{pmatrix} \vec{\boldsymbol{X}} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \sin\theta & -\cos\theta \\ \cos\theta & \sin\theta \end{pmatrix} \begin{pmatrix} a_x \\ a_y \end{pmatrix} \equiv \boldsymbol{A}\vec{\boldsymbol{X}} + \boldsymbol{B}\vec{\boldsymbol{u}} \qquad (2)$$

where ω is the angular velocity of the boat, θ is the heading of the boat relative to the Earth's magnetic north, and a_x and a_y are the measured linear (gravity-compensated) accelerations relative to its moving coordinates. Note that a_x and a_y are accelerations compensated for out-of-plane motion. See Appendix A for details on this state space model.

Because the Kalman filter functions in discrete time and is an iterative filter, the state space representation was converted to discrete time form using a backwards difference approximation:

$$\frac{\vec{\boldsymbol{X}}_{\boldsymbol{k}} - \vec{\boldsymbol{X}}_{\boldsymbol{k-1}}}{\Delta \boldsymbol{t}} = \boldsymbol{A}_{\boldsymbol{k}}\vec{\boldsymbol{X}}_{\boldsymbol{k}} + \boldsymbol{B}_{\boldsymbol{k}}\vec{\boldsymbol{u}}_{\boldsymbol{k}} \qquad (3)$$

$$\vec{\boldsymbol{X}}_{\boldsymbol{k}} = (\boldsymbol{I} - \boldsymbol{A}_{\boldsymbol{k}}\Delta t)^{-1}\left[\vec{\boldsymbol{X}}_{\boldsymbol{k-1}} + \Delta t \boldsymbol{B}_{\boldsymbol{k}}\vec{\boldsymbol{u}}_{\boldsymbol{k}}\right] = \boldsymbol{F}_{\boldsymbol{k}}\vec{\boldsymbol{X}}_{\boldsymbol{k-1}} + \boldsymbol{B'}_{\boldsymbol{k}}\vec{\boldsymbol{u}}_{\boldsymbol{k}} \qquad (4)$$

where Δt is the time (in seconds) between samples. This model was then passed into the following Kalman filter algorithm [1][2]:

$$\vec{\boldsymbol{z}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \vec{\boldsymbol{X}} \equiv \boldsymbol{H}\vec{\boldsymbol{X}} \tag{5}$$

$$\widehat{\boldsymbol{X}}_{\boldsymbol{k}} = \boldsymbol{F}_{k}\vec{\boldsymbol{X}}_{\boldsymbol{k-1}} + \boldsymbol{B'}_{k}\vec{\boldsymbol{u}}_{\boldsymbol{k}} \tag{6}$$

$$\boldsymbol{P}_{\boldsymbol{k|k-1}} = \boldsymbol{F}_{k}\boldsymbol{P}_{\boldsymbol{k-1|k-1}}\boldsymbol{F}_{k}{}^{T} + \boldsymbol{Q} \tag{7}$$

$$\boldsymbol{y}_{k} = \boldsymbol{z}_{k} - \boldsymbol{H}\widehat{\boldsymbol{X}}_{\boldsymbol{k}} \tag{8}$$

$$\boldsymbol{S}_{k} = \boldsymbol{R} + \boldsymbol{H}\boldsymbol{P}_{\boldsymbol{k|k-1}}\boldsymbol{H}^{T} \tag{9}$$

$$\boldsymbol{K}_{k} = \boldsymbol{P}_{\boldsymbol{k|k-1}}\, \boldsymbol{H}^{T}\boldsymbol{S}_{k} \tag{10}$$

$$\boldsymbol{X}_{k} = \widehat{\boldsymbol{X}}_{\boldsymbol{k}} + \boldsymbol{K}_{k}\boldsymbol{y}_{k} \tag{11}$$

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_{k}\boldsymbol{H})\boldsymbol{P}_{\boldsymbol{k|k-1}}(\boldsymbol{I} - \boldsymbol{K}_{k}\boldsymbol{H})^{T} + \boldsymbol{K}_{\boldsymbol{k}}\boldsymbol{R}\boldsymbol{K}_{k}{}^{T} \tag{12}$$

The matrices $\boldsymbol{R}$ and $\boldsymbol{Q}$ represent measurement and model covariance matrices, respectively. These particular matrices are diagonal because the position and velocity estimates are assumed to be independent in both the IMU model and the GPS measurements. Additionally, the values on the diagonal in each matrix were assumed to be equal for simplicity of implementation. The values on the diagonals needed to be designed. The design was implemented in practice by letting $\boldsymbol{Q}$ be an identity matrix and $\boldsymbol{R}$ be defined by:

$$\boldsymbol{R} = \begin{pmatrix} \mu & 0 \\ 0 & \mu \end{pmatrix} \tag{13}$$

where μ was a constant to be chosen by design.

Optimization and Quantitative Performance of State Estimation Algorithm

To estimate μ, sensor data was collected while walking a closed path of approximately one kilometer with the sensors held as close to horizontal as possible throughout the trajectory. The raw sensor data were then imported into MATLAB and run through a Kalman filter simulation while manually varying the value of μ. The constant was varied until the filtered position data matched reasonably well with the trajectory predicted by the GPS data while the speed profile reached maximum smoothness as determined by visual inspection (see Figures 14 and 15). Analysis of the test data revealed that values of μ between approximately 500 and 3000 provided acceptable estimates of boat position when compared to GPS data. Varying μ within this range tends to affect noise levels in the speed profile more than accuracy of position estimates.

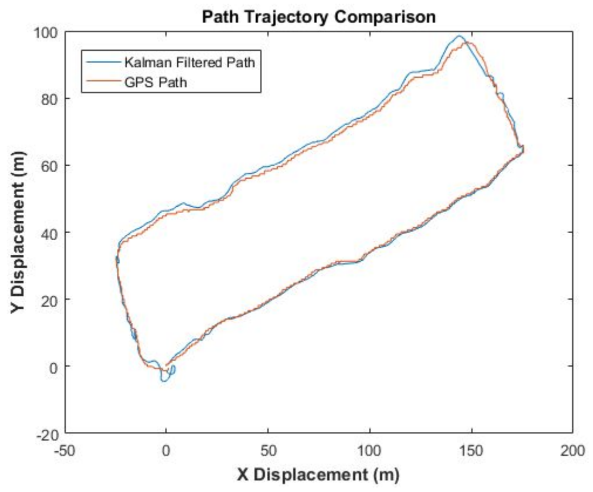


Figure 14: Position estimates from GPS measurements and from Kalman filter outputs. For covariance matrix estimates, a scaling value of $\mu = 1000$ was used in the above output. This test data demonstrates that the Kalman filter outputs provide reasonably accurate position measurements over large displacements while providing smoother measurements over smaller displacements.

By choosing to use the Kalman filter, the iterative algorithm is inherently limited by the sampling rate of the slowest sensor (i.e. 10 Hz from the GPS module). However, because the IMU can update at a rate as high as 100 Hz, a strategy that was attempted for improving this sampling rate was to perform dead reckoning estimation using Equation 4 in between GPS samples. Initial test results from land-based tests for this modified algorithm are shown in Figure 16. The results revealed an important trade-off between sampling rate and accuracy. The introduction of any amount of dead reckoning did not significantly affect stability of the algorithm but did have a noticeable effect on the noise in the algorithm's output measurements. The noise could be slightly reduced by increasing the value of μ, but this resulted in decreased accuracy in the estimates in the long term.

Both variants on the Kalman filter algorithm were tested with the boat in real time, and it was determined that a slower update rate was better tolerated by the feedback controllers than greater levels of noise in the speed measurements. Therefore, the Kalman filter was implemented with the 10 Hz sampling rate and did not include any amount of dead reckoning estimates in between GPS updates. Ultimately, a value of $\mu = 500$ was chosen for use in the real-time algorithm to produce reasonably accurate position measurements with a maximally smooth speed.

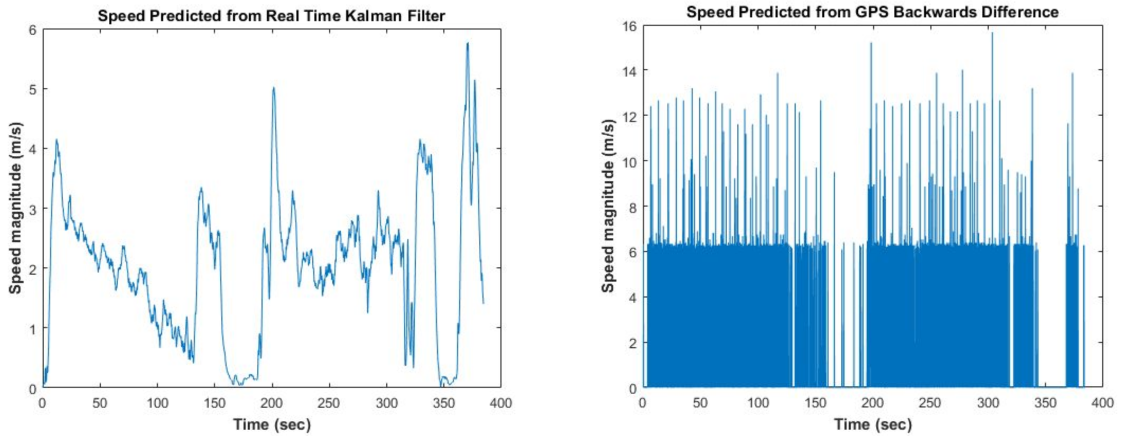


Figure 15: Boat speed estimated from the real time Kalman filter outputs (left) and backwards difference calculations from GPS measurements (right). Because the GPS measurements are not smooth over short time intervals, the speed profile from the backwards difference calculations results in noisy and inaccurate speed measurements which are unusable in real-time feedback controllers.

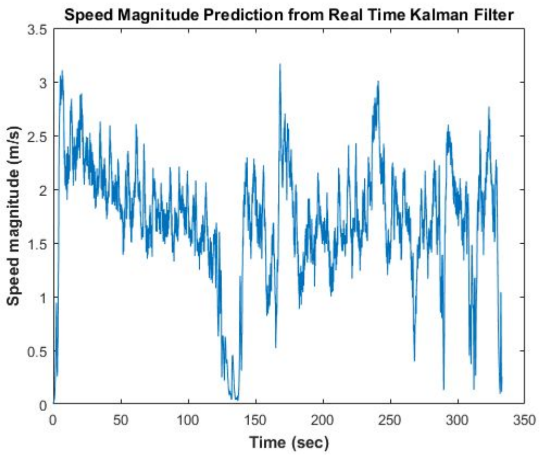


Figure 16: Boat speed estimated from the real time Kalman filter algorithm with dead reckoning estimates included in between GPS updates. Comparison with Figure 15 shows a significant increase in noise levels in the signal with a negligible effect on state estimate accuracy.

Results

Each task followed a very similar procedure to begin. Before beginning any test, the Arduino is reset and follows the calibration procedure for the sensors. The magnetometer on the IMU required the sensor to be rotated in a figure-eight-like motion in order to locate the direction of true north. The gyroscope was calibrated by allowing the sensor to remain at rest for a few seconds. An indicator light on a breadboard was used to indicate when sensor calibration was successful. Once calibration was completed and a GPS fix was received, the Arduino began collecting sensor data and performing state estimation. The boat remained in a remote-controlled state as the boat was prepared and placed in the water. For each test, the boat is initially oriented parallel to the dock. The task was initiated by flipping a switch on the RC controller which opens the relay switches and shifts control of the boat from RC to autonomous navigation. Each task continues indefinitely until RC control is reinitiated on the RC controller and the boat is manually driven back to the dock.

Constant Heading and Speed Task: Data and Analysis

To test task 1, the boat was given a setpoint heading of 140 degrees relative to true north and a setpoint speed of 1 meter per second. The control algorithms launch the boat from the dock and control the heading and speed to match the setpoints.

Post-task data processing was performed on the steady state task motion which ignored transient behaviors to reach the setpoint as well as data collected during manual RC navigation. The speed and heading estimates from one test of this task are shown in Figure 17 along with the average values for the displayed window of data. The data show that the boat's speed was maintained at 1.45 ± 0.63 m/s and its heading was maintained at 136.10 ± 11.79 degrees. It should be noted that the speed profile was highly oscillatory, indicating that the system has a significant amount of overshoot. This result was qualitatively observed during testing when the motors appeared to switch between being fully on and fully off.

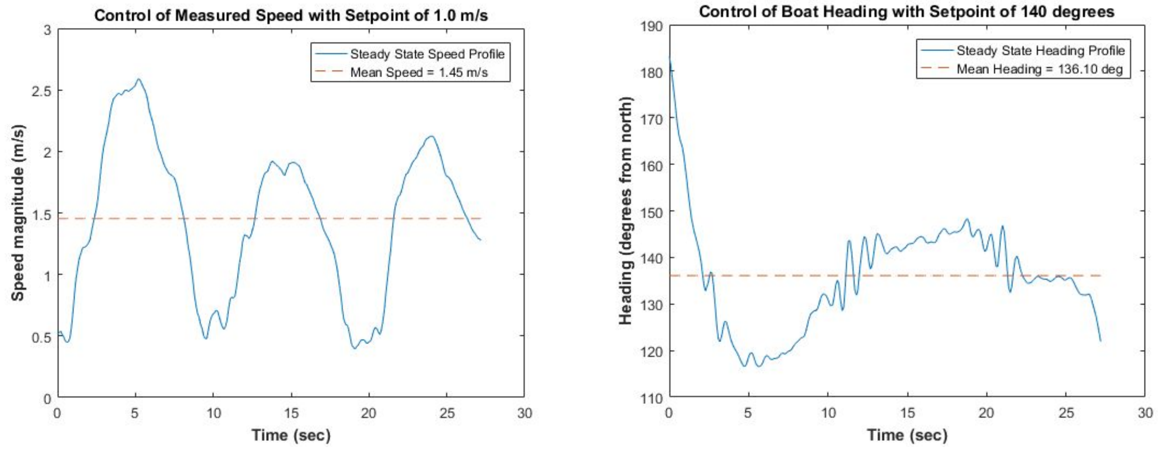


Figure 17: Boat speed (left) and heading (right) during steady state operation of the constant speed and constant heading control task. The dashed line indicates average value of the displayed window. The control algorithm had setpoints of 140 degrees for the heading

and 1 meter per second (m/s) for the speed. Average speed of the boat was 1.45 ± 0.63 m/s and average heading was 136.10 ± 11.79 degrees.

Circular Motion Task: Data and Analysis

To test task 2, the boat was commanded to travel at a constant speed (1 meter per second) at a heading of 180 degrees relative to north until it reached a distance of approximately 30 meters from the dock. The microcontroller then calculated a centerpoint for the boat's circular trajectory that was approximately 10 meters perpendicular to its current heading. The control scheme as detailed in previous sections was then initiated the circular motion control.

Figures 18 and 19 show the x-displacement, y-displacement, and speed of the boat during steady state operation of this task. As in the previous task, an average value of boat speed was calculated as 2.11 ± 0.71 m/s. The x- and y-displacements of the boat over time are important for determining whether the boat performs the circular motion about the centerpoint with constant radius. If the motion was performed with constant radius, the peak-to-peak measurements of the sinusoidal profiles in the x- and y-displacements plots should be equal. As shown in Figure 19, the estimated radius from each profile (defined as half the peak-to-peak measurement) was 11.22 and 12.44 meters, respectively.

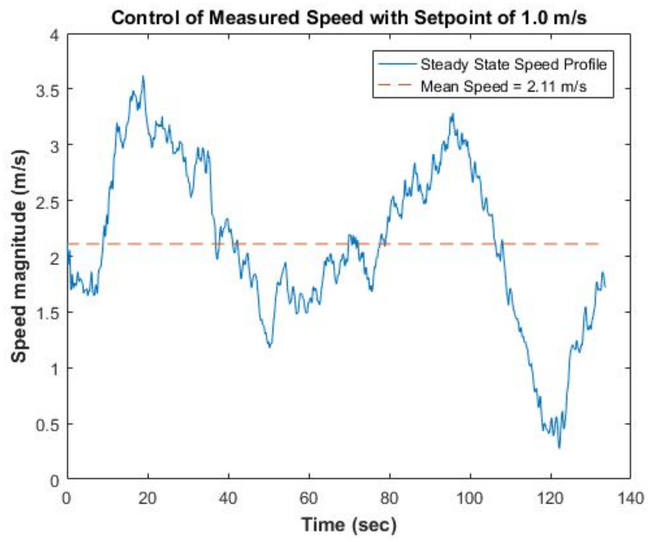


Figure 18: Boat speed during steady state operation of the circular motion task. The dashed line indicates average speed of 2.11 ± 0.73 m/s over the displayed window.

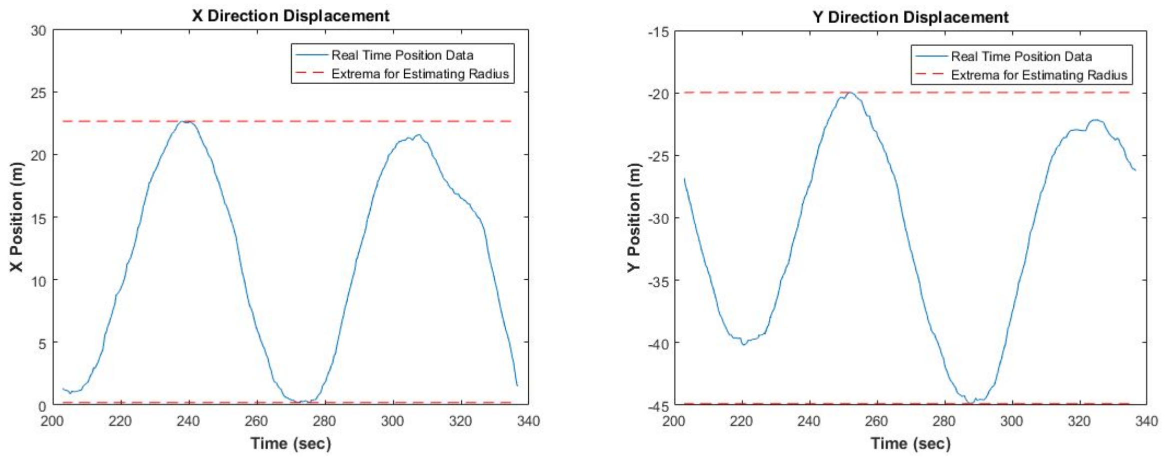


Figure 19: Boat x-displacement (left) and y-displacement (right) during steady state operation of the circular motion task. The dashed lines indicate extrema in each dimension. The radius of the motion predicted from the x-displacement (defined as half of the peak-to-peak value) was 11.22 meters, while the same prediction from the y-displacement was 12.44 meters.

Discussion and Conclusion

Mechanical Design

Stability and Maneuverability

Our boat was quite stable because of a number of design choices that were made. First, because a GPS antenna was not necessary to achieve greater GPS accuracy, the GPS module could be contained completely in the boat which helped keep the center of mass low. Additionally, the avionics box was rather compact, which allowed it to fit inside the boat and lowered the center of mass. Finally, the boat was properly ballasted so that it sat at the water line and the weight was evenly distributed for added stability.

While motor alignment was not well controlled during assembly, the flexible shaft couplings allowed small misalignments in of the motor shaft to be tolerated and helped boat maneuverability. Improvements to initial shaft alignment could improve maneuverability because if the motors were more symmetric, differences in motor speeds would reduced and allow the boat to more easily travel in a straight line. Despite the fast winds and rather choppy currents, the boat performed rather well with RC control. From a mechanical standpoint, the boat was quite maneuverable and stable, which we attribute to ballasting at the water line and sufficiently waterproofing the vehicle.

Waterproofing and Logistics

If we had more time, it would have been useful to have a more permanent cover for the boat. We originally planned to laser-cut an acrylic top and use weatherproofing so that it would be watertight and easily removed, but in the end settled for a trash bag cut and folded, and taped with duct tape. While this approach was by no means professional, it was rather effective, but after a couple of test runs a sponge had to be used to soak up the water that accumulated at the bottom of the boat. It was also tedious to have to untape the cover in order to re-calibrate the sensors each time, which is why a more permanent cover would have been useful. This problem could be further improved by finding a way to avoid sensor recalibration each time the Arduino program is reset.

Task Performance

Section 7 detailed the results from our boat's performance of tasks 1 and 2. In general, the results showed that the boat's heading control was robust and reliable despite the presence of significant external disturbances such as high winds and waves. In the performance of task 1, the control system successfully maintained the boat's setpoint heading and speed to within one standard deviation of the average heading and speed. In both controllers, overshoot of the setpoint became significant in some cases. This observation is most clear in the speed profile where the speed tended to oscillate about the mean value. In task 2, the boat successfully navigated a trajectory that qualitatively appeared circular. Similar to task 1, the speed profile displayed some oscillatory behavior about a mean value. However, in task 2 this speed was significantly different from the setpoint of 1 meter per second.

From the task 2 results, it could be shown that the boat was navigating in a circular path with a radius that was reasonably close to the commanded setpoint in the control algorithm. Additionally, because the x- and y-displacement profiles estimated similar values for the radius, the radius was considered reasonably constant. However, while the boat was performing motion that appeared circular and could be considered circular, it is possible for the performance to improved to maneuver in a more circular path. One explanation for the fact that motion was not perfectly circular may be that the navigation algorithm employs an "on-off" control scheme. While acceptable for an initial attempt at control, "on-off" control tends to result in large errors over time. In future work, other control schemes may be devised that make use of error measurements to control the motion which can decrease error in the trajectory.

These results showed clearly that further adjustment and optimization of the controllers may be necessary to achieve more continuous and accurate control of heading and speed. In the case of the heading control system, this would most likely involve tuning of gains because the heading measurements from the IMU have negligible noise and are highly accurate. In the case of the speed controller, it may be possible to improve the Kalman filter algorithms to provide more accurate speed estimates with lower levels of noise in addition to optimizing gain values of the controllers. It is also possible that other bugs occurred in the speed controller that were not found prior to or during testing and should be investigated in future work.

In testing our boat's control algorithms for each task, we were unable to obtain functionality or data for task 3. This was due to lack of time to correctly implement and debug the conceptualized algorithm. In future work, the algorithm for this task will be reexamined to identify bugs in the program. In addition, the devised algorithm relied on knowing the general location of the buoy for proper operation. Future iterations of the algorithm may involve including an obstacle detection sensor such as an ultrasonic sensor to detect the buoy regardless of location.

Practical Applications

Below are some of the possible applications for our boat:

- Getting sensor data from the Charles (temperature, humidity, etc.)
- Programming the boat to autonomously drag the Charles for trash
- Stealthy assault boat
- Attacking geese
- Transporting backpack or supplies across the Charles for you when you're walking back to Boston after a late night 2.017 pset group
- Intercepting messages transmitted from a buoy and gathering intel on other boats

References

[1] Terejanu, G. A., "Discrete Kalman Filter Tutorial."

[2] Lacey, T., "Tutorial: The Kalman Filter."

Appendices

Appendix A: Derivation of Kinematic Model for Kalman Filter State Estimation

The following derivation details how raw sensor data was converted to the state space representation given in Figure 13 on page 15 with the state vector defined in equation 1.

The GPS module provided latitude and longitude measurements in degrees. However, because the IMU provided displacements in units of length, the GPS measurements needed to be converted to displacements. Therefore, a coordinate system was established each time that the Arduino was reset in which the origin is located at the start position of the boat with the positive y-axis defined in the direction of true north. GPS coordinates were then converted to displacements using small angle approximations:

$$x_E = \left(\frac{\pi}{180}\right)(\lambda - \lambda_{start})\, R_E \cos\left(\frac{\pi}{180}\lambda_{start}\right) \qquad (14)$$

$$y_E = \left(\frac{\pi}{180}\right)(\phi - \phi_{start})\, R_E \qquad (15)$$

where x_E is the boat's x-coordinate (in meters) relative to its starting position, y_E is the boat's y-coordinate (in meters), λ and ϕ are the boat's GPS longitude and latitude (respectively) in degrees, R_E is the radius of Earth (in meters), and λ_{start} and ϕ_{start} are the boat's GPS longitude and latitude at the start of operation. Note that it was assumed that $\lambda \approx \lambda_{start}$ in the correction of the Earth's radius for x_E . The simplifying assumptions made in these conversion equations are justified by the fact that the boat operates over a relatively small distance along the Earth's surface.

Along with the position estimates from the GPS module, estimates of position and velocity were obtained from kinematic measurements from the IMU. A state space representation for the boat's motion based exclusively on kinematic variables was developed. This model assumed that the boat was a two-dimensional rigid body with three degrees of freedom. The model was developed from the following kinematic equations for two-dimensional planar motion:

$$\frac{du}{dt} = a_x + v\omega \qquad (16)$$

$$\frac{dv}{dt} = a_y - u\omega \qquad (17)$$

$$u_E = u\sin\theta - v\cos\theta \qquad (18)$$

$$v_E = u\cos\theta + v\sin\theta \qquad (19)$$

$$\frac{dx_E}{dt} = u_E \qquad (20)$$

$$\frac{dy_E}{dt} = v_E \qquad (21)$$

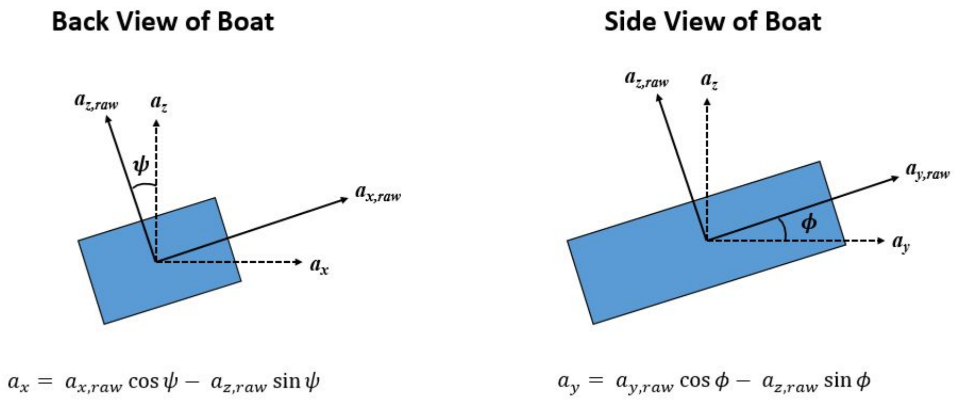
where u and v are the boat's velocity components in a reference frame fixed to the boat, x_E and y_E are the boat's position relative to its position at the start of the task, u_E and v_E are the boat's velocity in the x- and y-directions relative to the fixed Earth frame (with the y-axis aligned with the direction of true north), ω is the angular velocity of the boat, θ is the heading of the boat relative to the Earth's magnetic north, and a_x and a_y are the measured linear (gravity-compensated) accelerations of the boat relative to its moving coordinates. The model is shown schematically in Figure 13. Equations 16 through 21 can be simplified to express the system in state space representation as in equation 2.

While the boat was modeled as planar, the effects of out of plane pitch and roll orientations can be quite significant. Therefore, the values of a_x and a_y account for out of plane motions with:

$$a_x = a_{x,raw} \cos\psi - a_{z,raw} \sin\psi \qquad (22)$$

$$a_y = a_{y,raw} \cos\phi - a_{z,raw} \sin\phi \qquad (23)$$

where $a_{x,\,raw}$ and $a_{y,\,raw}$ are the accelerations collected from the IMU, ψ is the pitch angle of the boat and φ is the roll angle of the boat. Similar to heading, pitch and roll angles can be measured very accurately and with negligible levels of noise from the fused orientation output from the IMU. This correction is accounted for schematically in Figure 20.



Figur 20: Out of plane orientation and its effects on linear acceleration measurements from the IMU. ψ denotes the pitch angle of the boat and φ denotes the roll angle of the boat.

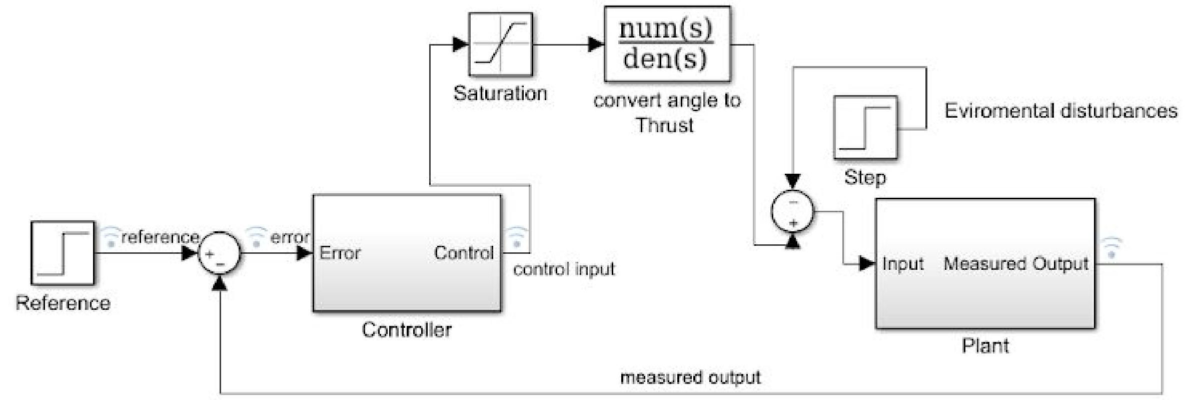
Appendix B: MATLAB Code Used to Generate Kalman Filter Source Code

The Kalman filter algorithm was initially created in MATLAB because it handles matrix operations more easily than the same operations in the Arduino language. The below code was passed into the MATLAB Coder application and source code for the function was generated in C language to be used in the Arduino compiler.

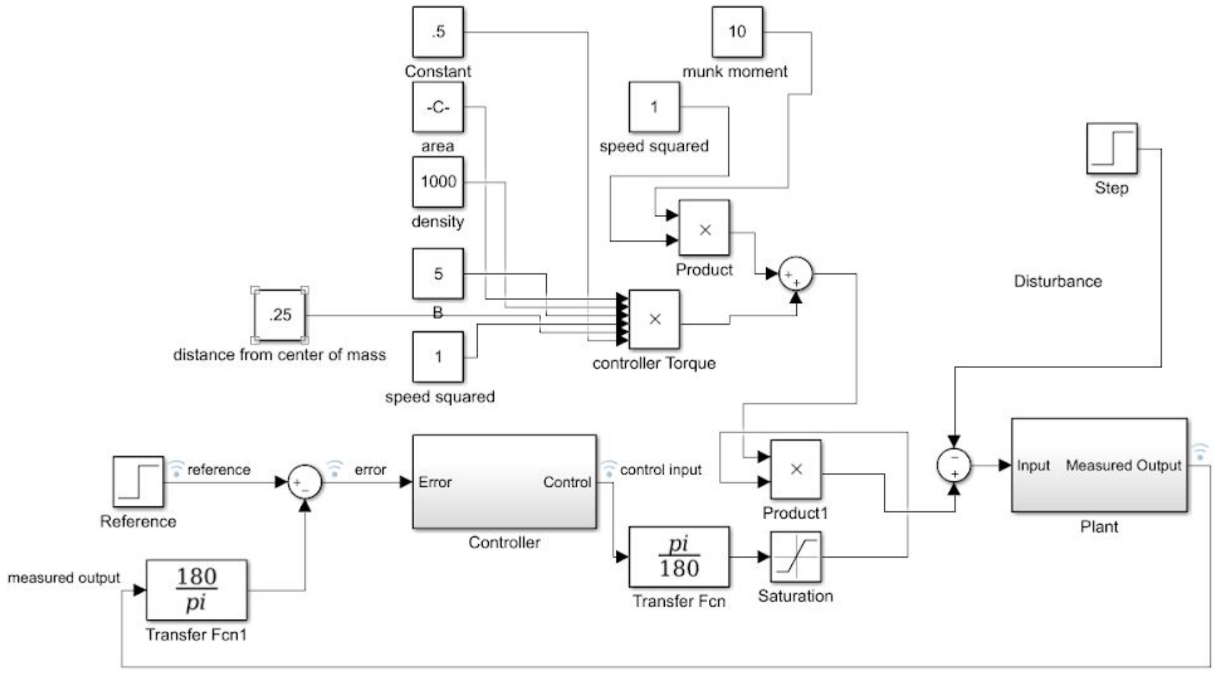
```
function [x, y, velX_earth, velY_earth, Pk_11_next, xDeadReckoning] = kalmanFilterMatrices(om
ega_movingAvg, accelX_movingAvg, accelY_movingAvg, deltaT, x_i_1, y_i_1, velX_earth_i_1, velY
_earth_i_1, heading, varQ, varR, GPSx, GPSy, Pk_11_input, xDeadReckoning_i_1)
    % Set up the system
    F = 1/(4*(deltaT/1000.0*omega_movingAvg)^2 + 1) .* ...
        [4*(deltaT/1000.0*omega_movingAvg)^2 + 1, 0, deltaT/1000.0, 2*omega_movingAvg*(deltaT
/1000.0)^2; ...
        0, 4*(omega_movingAvg*deltaT/1000.0)^2 + 1, -2*omega_movingAvg*(deltaT/1000.0)^2, del
taT/1000.0; ...
        0, 0, 1, 2*omega_movingAvg*deltaT/1000.0; ...
        0, 0, -2*omega_movingAvg*deltaT/1000.0, 1];

    B_prime = (deltaT/1000.0)/(4*(deltaT/1000.0*omega_movingAvg)^2 + 1) * ...
        [2*omega_movingAvg*(deltaT/1000.0)^2*cos(heading * pi/180) + sin(heading * pi/180)*de
ltaT/1000.0, 2*omega_movingAvg*(deltaT/1000.0)^2*sin(heading * pi/180) - sin(heading * pi/180
)*deltaT/1000.0; ...
        -2*omega_movingAvg*(deltaT/1000.0)^2*sin(heading * pi/180) + cos(heading * pi/180)*de
ltaT/1000.0, 2*omega_movingAvg*(deltaT/1000.0)^2*cos(heading * pi/180) + sin(heading * pi/180
)*deltaT/1000.0; ...
        2*omega_movingAvg*(deltaT/1000.0)*cos(heading * pi/180) + sin(heading * pi/180), 2*om
ega_movingAvg*(deltaT/1000.0)*sin(heading * pi/180) - cos(heading * pi/180); ...
        -2*omega_movingAvg*(deltaT/1000.0)*sin(heading * pi/180) + cos(heading * pi/180), 2*o
mega_movingAvg*(deltaT/1000.0)*cos(heading * pi/180) + sin(heading * pi/180)];

    u = [accelX_movingAvg; accelY_movingAvg];
    H = [1, 0, 0, 0; 0, 1, 0, 0];
    x_1 = [x_i_1; y_i_1; velX_earth_i_1; velY_earth_i_1];

    Q = [varQ, 0, 0, 0;...
        0, varQ, 0, 0;...
        0, 0, varQ, 0;...
        0, 0, 0, varQ];

    R = [varR, 0; 0, varR];
    z = [GPSx; GPSy];

    % Position estimate
    xhat = F*x_1 + B_prime*u;

    % Kalman filter algorithm
    Pk_1 = F*Pk_11_input*F' + Q;

    yhat = z - H*xhat;

    S = R + H*Pk_1*H';
    K = Pk_1*H'*inv(S);

    xMat = xhat + K*yhat;

    x = xMat(1);
    y = xMat(2);
    velX_earth = xMat(3);
    velY_earth = xMat(4);

    Pk_11_next = (eye(4) - K*H)*Pk_1*(eye(4) - K*H)' + K*R*K';
    % Dead reckoning estimation for comparison
    xDeadReckoning = F*xDeadReckoning_i_1 + B_prime*u;

end
```

Appendix C: Simulink Models

Simulink Model for Speed



Simulink Model for Heading


.5
Constant
10
munk moment
-C-
area
1
speed squared
1000
density
Product
Step
Disturbance
5
B
.25
distance from center of mass
controller Torque
1
speed squared
reference
error
control input
Error
Control
Product1
Input
Measured Output
Reference
Controller
Plant
measured output
180
pi
Transfer Fcn1
pi
180
Transfer Fcn
Saturation