BadgerBoat Team 2
Autonomous Boat Final Paper

Peter Sauer, Thomas Kelly, Jake Rymsza, Yoni Gelb
Mechanical Engineering
University of Wisconsin - Madison
Apr 30, 2020

Executive Summary

BadgerBoat is a research project where the client, Lennon Rogers, posed a challenge without formal design specifications. Lennon Rogers wants to create an autonomous boat system that can navigate the local lakes and rivers to collect useful data. The specific way that the boat should help the community was up to the team. Our proposed project was to create an autonomous, solar powered boat that detects and records data on dangerous blue-green algae. The boat will reduce man hours required to collect information, and provide continuous streams of data, hopefully leading to cleaner, safer lakes and rivers. This year long challenge was broken down into five subsections of design specifications: boat construction, unmanned navigation, collect useful data, multi-day journey, and total autonomy. The project consisted of three teams that work in parallel. Team 2 was tasked with building the boat and making it autonomous, while team 1 focused on power supply and regeneration and team 3 focused on a payload that could record useful environmental data. In the fall semester, the goal of BadgerBoat team two was to have the boat built and achieve unmanned GPS navigation. In the Spring semester, team two was able to add obstacle avoidance to make the boat truly autonomous.

BadgerBoat team 2 chose the current boat design from careful research of other autonomous surface vessels and analytical prototypes. The initial idea was inspired by Michigan RobotX and the SeaRobotic Utility 3.6 boats. The team modeled the mechanics of a two hulled boat with differential steering in EES before making purchases. Using this analytical prototype, the team made informed decisions regarding the boats hull, motors, electronics and controls.

During the design process the team was selected to compete in the SICK TiM$10K lidar challenge to use LiDar technology in new and innovative ways. With a prototype hull and frame, electric trolling motors were connected with motor controllers to the BadgerBoat brain, a JetsonNano. A GPS sensor and MOOS libraries create waypoints for unmanned navigation, but the craft has a wifi modem for remote control from shore. LiDar technology and custom ROS nodes were used to provide an accurate obstacle avoidance mechanism for the boat, in conjunction with preexisting MOOS nodes.

By the end of the year, the BadgerBoat team completed multiple working prototypes achieving basic maneuverability and unmanned navigation on Lake Mendota. In the tests on December 7th, the BadgerBoats followed GPS waypoints around a course while the navigation program from MOOS continuously accounted for a strong crosswind. The software measured a maximum speed of 3 m/s, or 5.8 knots. In on land testing, the boat was capable of recognizing obstacles and planning new paths to avoid them. The boat will continue to see improvements in coming semesters, however we believe that this is a strong base upon which the payload team will be able to make autonomous testing a possibility.

Table of Contents

Introduction

Autonomy is becoming more and more prevalent in society today. Advances in modern computing and new programming techniques allow for the processing of large amounts of information in relatively quick time periods. This makes the integration of complex sensors into vehicles much easier -- thus allowing autonomy to be reached more easily and to be able to handle complex circumstances.

The client, Lennon Rodgers, felt that this new area of technology would be an interesting field for a capstone team to explore. He tasked the team to create an autonomous boat that could navigate local and national waterways and take useful data such as water samples along the way.

Our opinion of useful data is monitoring signs and photos of beaches at risk of poisonous blue green algae. The Yahara waterways are at high risk of toxins from decaying blue-green algae, a cyanobacteria that thrives in fertilizer-rich, slow moving water. Every summer, beaches around these lakes are closed for public safety because of harmful algal blooms, or HABs[1]. HAB's also decimate fish populations because the decaying algae raise water temperature and consume dissolved oxygen until fish suffocate underwater. The BadgerBoat will gather photos of blue-green algae and test with a probe for dissolved oxygen.

This report describes the building of the boat, making it autonomous, and testing on the lake. After clarifying the goals of the project in the problem statement, the report follows the steps the team took to choose a hull, frame, motors. Next, the team connected the nervous system of the boat. The battery power, GPS, motor controllers, and Lidar were connected to a Jetson nano. The boat was then outfitted with custom software and algorithms that allowed for GPS waypoint navigation and obstacle avoidance. Finally, the report discusses the results of the water tests with the boat on Lake Mendota.

Problem Statement and Design Specifications

The Badger BOAT project has continuously evolved throughout the year. It started with an open-ended challenge from our client, Lennon Rodgers. Lennon is the director of the UW-Madison Makerspace and has a special interest in autonomous technology. He sponsored our project with the hope of using autonomy in new and creative ways. His initial challenge was simple: create an autonomous boat capable of navigating the waterways around Madison and making it to the Mississippi while doing something useful. This last part was intentionally left open to allow the project to evolve how the team seemed fit.

As the project progressed, a purpose was found. Blue-Green Algae is a common problem that harms the lakes and Madison communities throughout the summer. An autonomous boat that could continuously track and study this algae or other contaminants could greatly improve the lakes and allow the Madison community to enjoy them for a larger percent of the summer.

Our final proposed project was to create an autonomous, solar powered boat that detects and records data on dangerous blue-green algae. The boat will reduce man hours required to collect information, and provide continuous streams of data, hopefully leading to cleaner, safer lakes and rivers. This year-long challenge was broken into three main subjects: power systems, autonomy and data collection. As the autonomy group, our three main focuses were the boat's structure, computing electronics and the autonomous software.

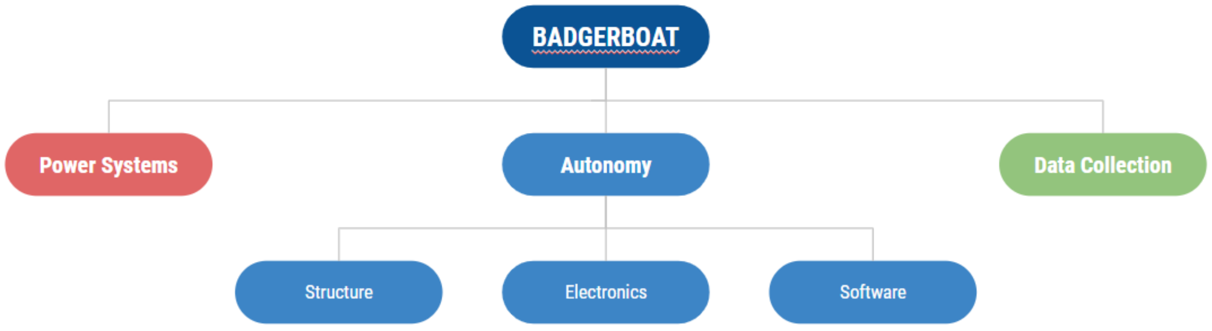


Figure 1. The overall breakdown of the BadgerBOAT project, including the three focuses of the autonomy group.

Structural Prototyping

The Badger Boat was designed by supporting qualitative decisions with calculations because there were few design specifications. Initial decisions were inspired by similar projects like the 2018 Autonomous Surface Vehicle from University of Michigan's RobotX team[2]. An EES model was the analytical prototype that provided quantitative design specifications. From the analytical model, the team made informed purchases of the hull.

The first design decision was hull shape; whether to use one or two hulls. We chose to use two hulls for stability and buoyancy. Not capsizing was paramount because an unmanned vessel would be unable to right itself. As supported in the paper "The Design of an Autonomous Surface Vehicle for the 2018 Maritime RobotX Challenge," a vessel with two hulls can be more stable than a larger, less modular, single hull design[2]. A stable vessel allows for a higher center of gravity, so the batteries and electronics can be located higher and drier. A wide, two hulled boat allowed for the use of twin screw steering. Twin screw steering is easier to make autonomous because it allows for boats to rotate without having to drive forward.

The team compared rigid versus inflatable hulls, and fiberglass versus rigid plastic. The SeaRobotics Utility 3.6 had rigid plastic hulls while Michigan RobotX had inflatable pontoons[3]. While inflatable hulls are easier to store, a quality inflatable hull was consistently more expensive. They also have a shorter lifespan according to "Inflatable Boat Lifespan" by Newport Vessels[4]. Similarly, fiberglass was more expensive and more difficult to repair than rigid plastic.

EES Prototype

Before any hull construction was considered, calculations were made in EES to determine important variables, such as drag force on the boat and ideal widths/lengths. With this analytical prototype, the cost-benefit of changing variables is quantifiable. As mass increases, the max speed decreases at a decreasing rate. The calculations helped the team to determine what hull would be ideal, but also ensured that the boat design was feasible before purchases were made.

Max speed. Top achievable speed was modeled using an approximation of total drag on the two hulls that was set equal to the thrust according to the motor manufacturers. Equation 1 and 2 break down the calculations for viscous and total drag. The mass of the boat included two batteries, and the hulls were modeled as triangular prisms. Two 35 lbf motors were initially chosen to reach a speed of ten knots.

$$\Sigma F_{drag} \sim 1.1 F_{viscous\ drag} \tag{1}$$

$$F_{viscous\ drag} = .5 * \rho_{water} * fr * A_{wet} * u^2 \tag{2}$$

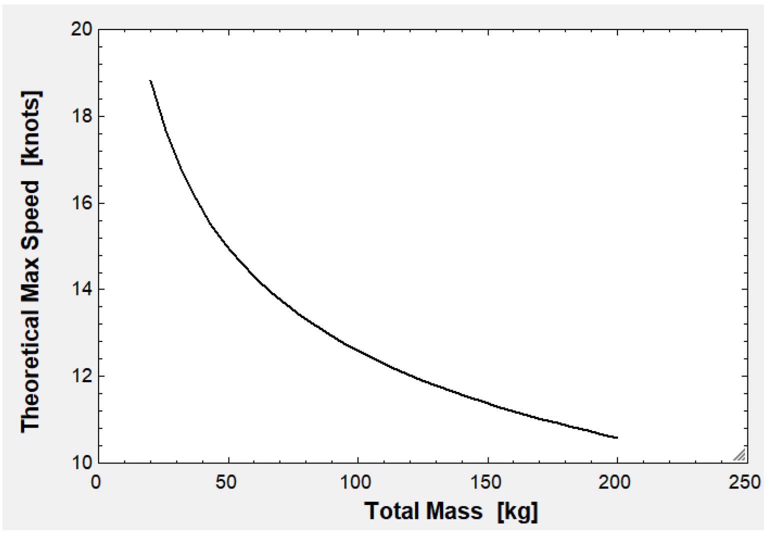


Figure 2: Theoretical max boat speed at boat sizes with two 35 lbf motors.

Final Hull Choice

Multiple different hull options and configurations were evaluated to find the optimal choice for the project. The table below compares four watercrafts sold as recreational paddle and trolling motor powered catamarans.

Product	Listed Price ($)	Max Weight	Weight	Length	Width	Pros	Cons
XCAT[5]	5,000	575 lbs	121 lbs	16’5”	6’10”	Width for middle twin screw.	Trampoline midsection.
Biyak[6]	2,000	325 lbs	130 lbs	12’7”	30-50”	Easy to modify midsection. Deals wt Rutabaga.	Mounting setup only for single motor.
BlueSky 360 Escape[7]	3,400	500 lbs	130 lbs	13’4”	48”	Steering options: rudder system and pedal prop.	More gear on it than we need.
Expandacraft Outrigger kit [8]	1,000	400 lbs	75 lbs	12’	5’-8’	Modular and easy to modify.	Plastic bolts and risers are weak points.

Table 1: Different hull options and evaluation metrics.

The Expandacraft Outrigger Kit was the final choice because it is lightweight and can be easily separated into small parts and modify, as well as being the least expensive. The lightweight hull reduces the needed thrust to reach ten knots. With the hull chosen, a frame design could begin.

Frame Design

The current frame used in boat testing is not the planned final frame -- scrap metal from the Expandacraft pre-built frame was used. to make something that basic testing could be done on while the material and skills (welding) are obtained.

Intermediate Frame for Testing. The frame built for testing used scrap metal from the kit sent by Expandacraft. Steel Screws were substituted for the plastic ones given by the kit to make the frame strong enough to stand up to lake conditions. A semi-waterproof box was strapped to the frame using a ratchet strap and holds the batteries and electronics. The motors are clamped onto the frame using their given mounting system.

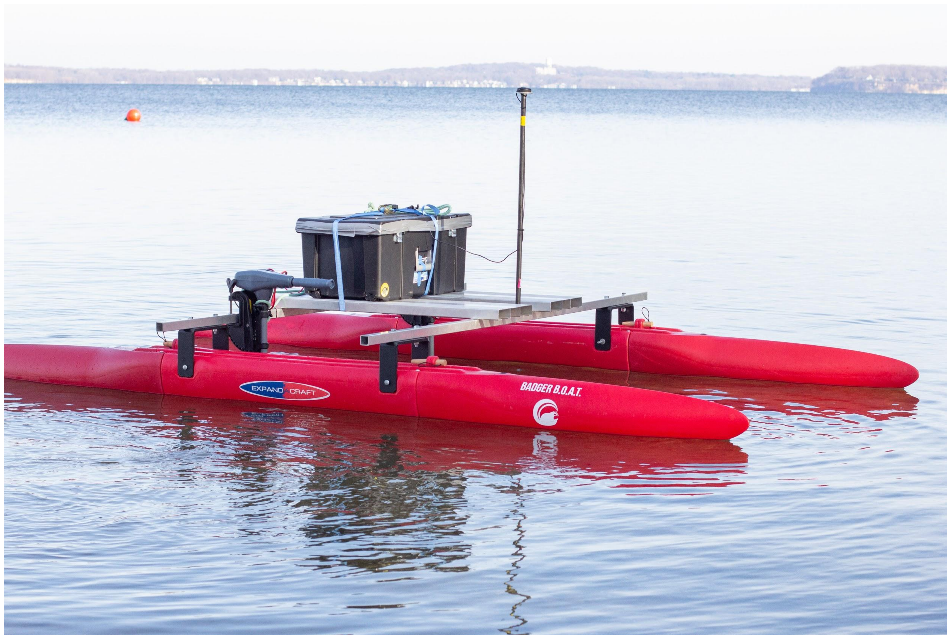


Figure 3: Intermediate frame on the boat during testing.

Second Iteration Frame CAD and Finite Element Analysis. The second iteration of our design planned to use welded aluminum tubing which was much more structurally sound than the basic testing design. This frame was designed in SolidWorks and we were able to perform FEA to determine if the frame would be capable of carrying the planned weight of the boat.

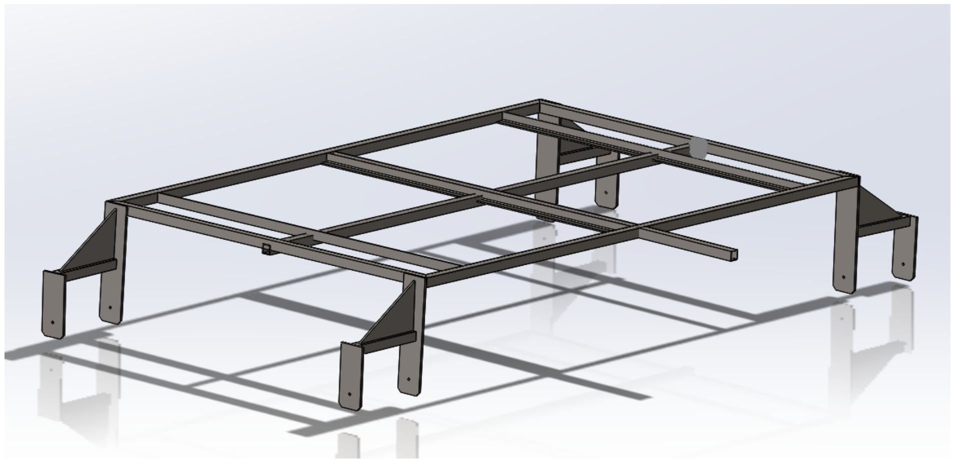


Figure 4: Future frame designed in SolidWorks

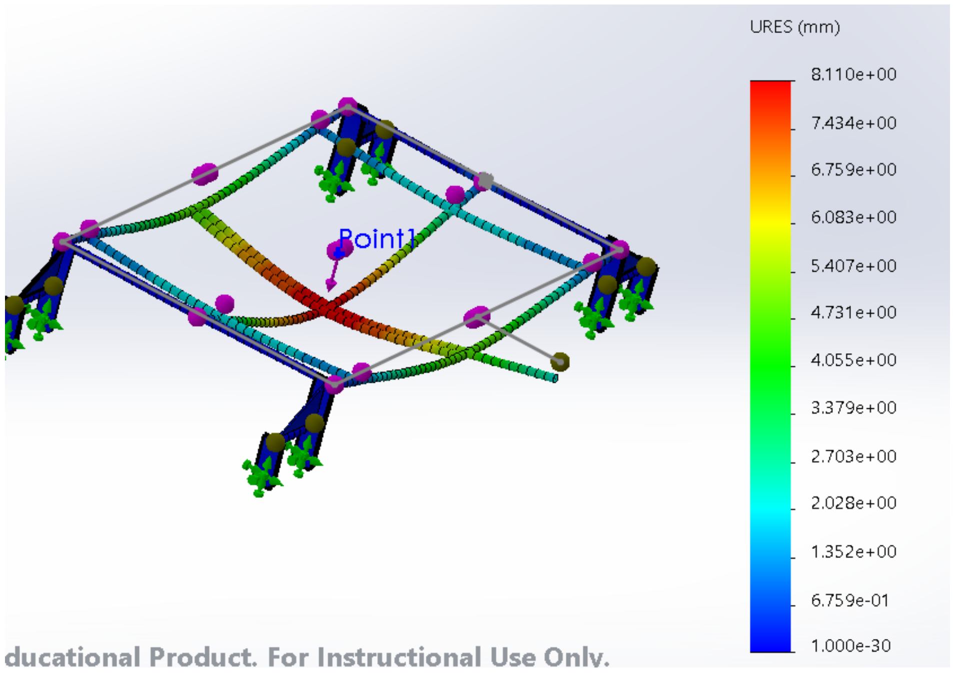


Figure 5: Deflection in mm with a 300 lb point load.

From FEA it was found that with a 300lb point load applied to the center of the frame only less than a centimeter of deflection occurred. This seemed to be reasonable to the team as the point load is likely to create a much greater deflection than if the weight were distributed more evenly across the frame. 300 lb is also a great deal more weight than will be put on the frame.

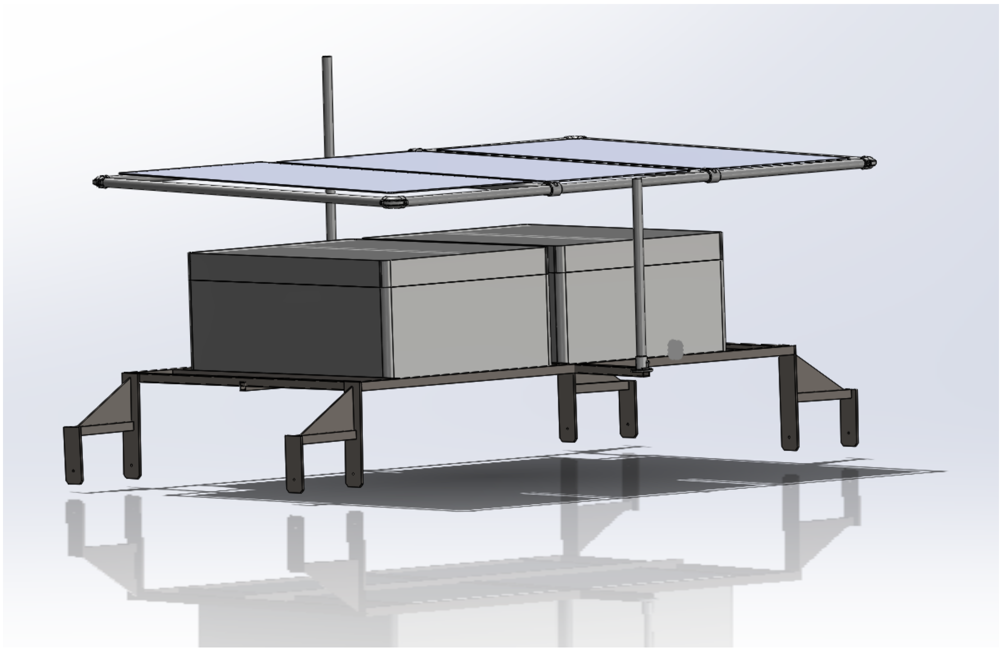


Figure 6: Full frame with external elements in SolidWorks

The external elements of the frame (such as the planned addition of pelican cases and solar panels) were also designed in CAD to ensure that all would fit nicely on the boat. The plan for the solar array was to have it at a fixed angle, but the design does allow for that angle to be adjusted -- and also possibly allows for adjustment during a run in the future.

Final Frame - Aluminum Extrusion. Eventually, due to time constraints, it was decided that the frame of the boat would be made of lightweight, slotted, aluminum extrusion that allowed for quick and easy customization to fit nearly any mission or payload. The frame included a solar rack that can hold three 100 watt solar panels. The rack be lifted up for easy access to the electronics and can be completely removed for smaller journeys that do not require solar power, but angle adjustment capabilities were saved for future designs.

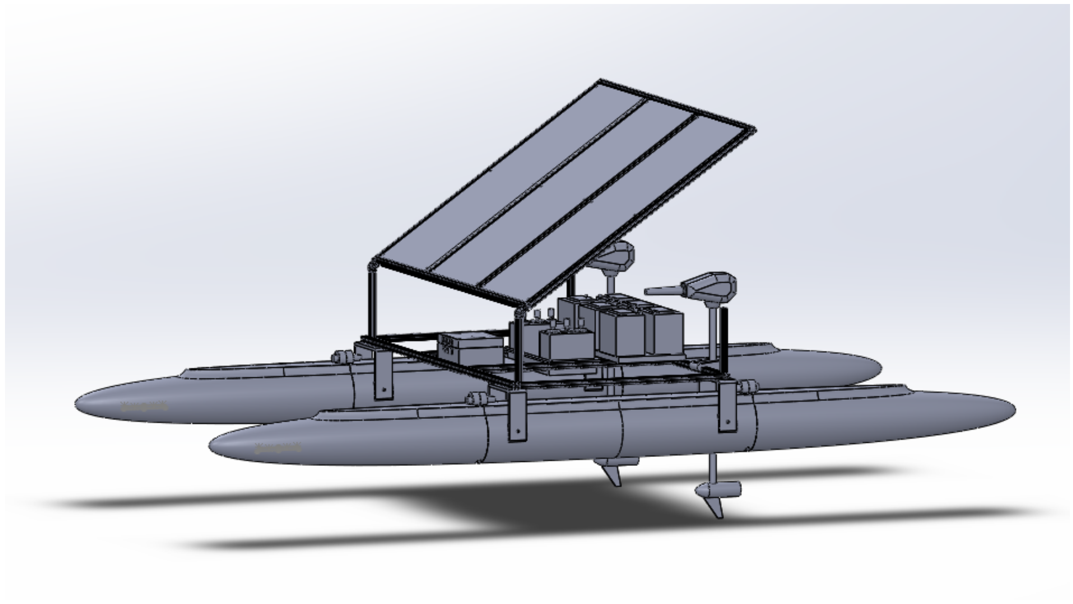


Figure 7: Final frame design using aluminum extrusion in SolidWorks.

Electronics and Control

Motor Selection

One of the most important parts of the boat is the motors. We researched a wide variety of different options, but we were limited by our need for electric motors that can run underwater. Because of this, we settled on looking at trolling motors. Several different brands and configurations of trolling motors were researched and several factors were considered in motor selection.

Motor	Thrust Provided	Cost	Other Factors
Newport Vessels[9]	36 lbs	$139	Small, low power, easily mounted to frame, 12 V
Newport Vessels[9]	55 lbs	$159	Small, higher power draw, easily mounted to frame, 12 V
MinnKota Ulterra[10]	80 lbs	$2,849	Medium sized, higher power draw, 24 V, harder to control, has powered steering
MinnKota Vantage[11]	80 lbs	$1,549	Large, higher power draw, 24 V, has

			powered lift

Table 2: Motor selection design matrix

Ultimately, the Newport Vessels 36 lbs motor was chosen because of its low price point and ease of configuration. The ideal motor was the MinnKota Vantage, but because of its high price point it was not selected.

Control Systems

In order to be able to control the boat both remotely and autonomously, a control scheme was devised. Two VEX Pro motor controllers, an Arduino Nano and an NVIDIA Jetson Nano are used to control the two Newport Vessels 36lbs thrust trolling motors that are used for propulsion.

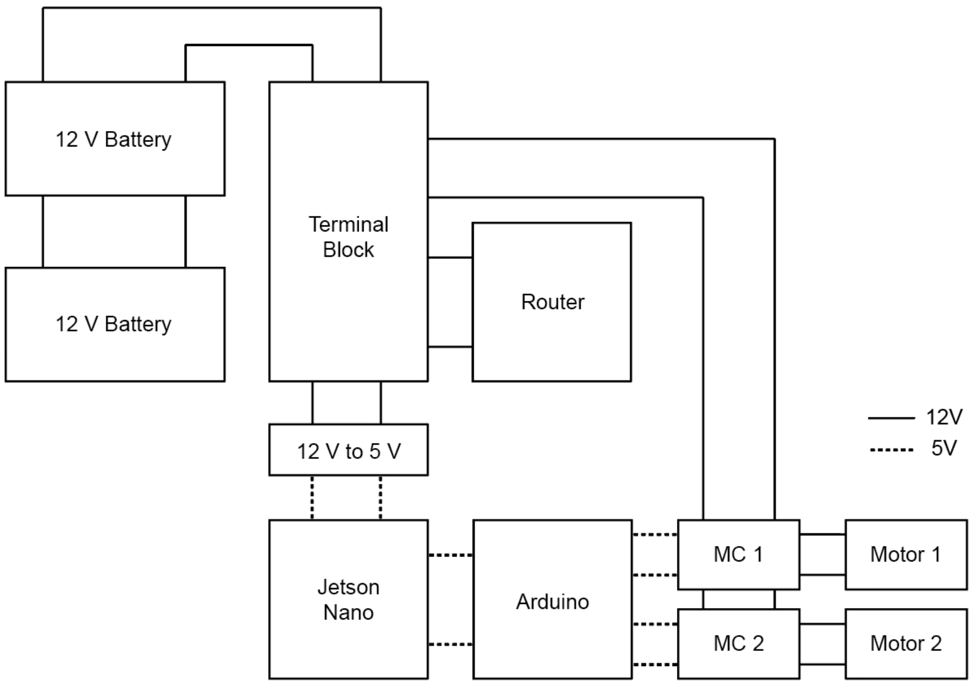


Figure 8: Wiring diagram for the boat

A router was used to provide wireless communication between the boat and a control device (laptop) either on shore or on the water with the boat.

Solar Array and Batteries

The boat was powered by a bank of 6 Valence 12V batteries. These batteries held enough charge for missions lasting up to 11 hours, or more if using the solar panels. The batteries could be remotely monitored using a built in CAN-BUS connection that provided reliable battery information. The details of the battery system were beyond the scope of our group, but they are relevant as they provided power for the rest of the electrical system the boat runs on. This includes the motors, sensors and the computers that made the boat autonomous.

Jetson Nano

For the brains of our system we chose to use a Jetson Nano. The Nano is essentially a small computer powerful enough to handle all the calculations and processing the boat needs to do. We chose the Nano due to its easy to use operating system and the well documented libraries that exist for it. The Nano also has all of the ports and data pins that we need for connecting it to the rest of our system. The Jetson takes data from the GPS system via USB connection. It also takes in data from an onboard router that we used to directly communicate to the Jetson when needed. Motor controls are then output through a USB port to an Arduino Nano. The Arduino then reads the given directions and converts it into a proper PWM signal for our motor controllers. The reason for using these two boards is simple, the Jetson makes all the decisions on how to control the board as it is optimized for processes of automation. The data is then sent to the Arduino as it runs on a constant cycle and can be trusted to send out PWM commands via analog output without any distortion. In this way the Jetson is serving as the systems brain and the Arduino serves as a muscle.

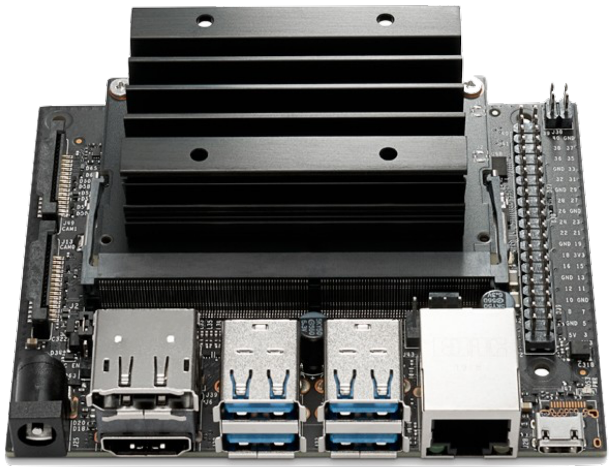


Figure 9: Jetson Nano

Controls Box

To encapsulate all of our electronic equipment into a waterproof environment, a box was designed using a pelican case, which was capable of mounting all of the necessary equipment. Ports were made which allowed certain wires to leave the box so that they could make important connections to batteries/motors and other external elements. This box also went through two iterations, one of which was much larger, less waterproof and incorporated the batteries on the inside. The other was completely watertight, about 1/8th of the size of the original, and left the batteries on the outside for thermal management reasons.

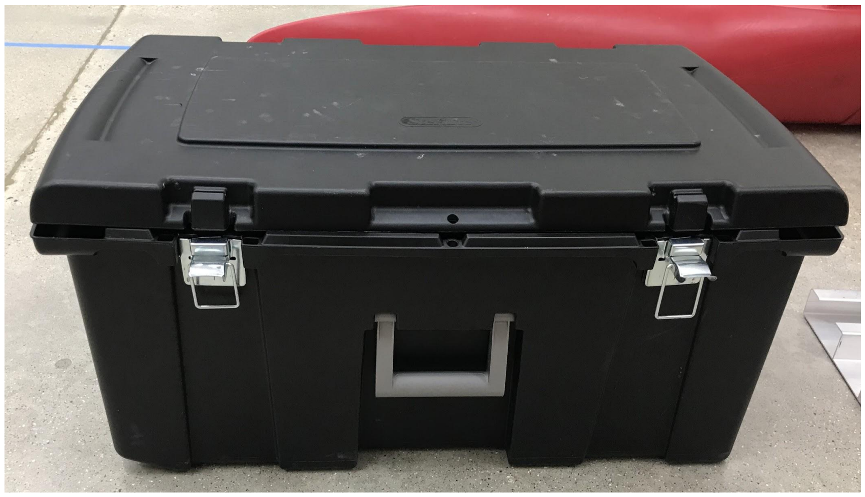
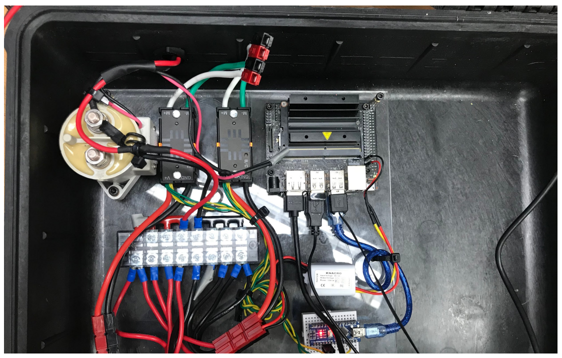
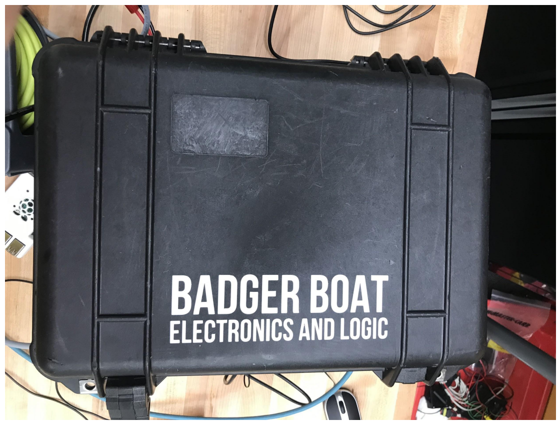


Figure 10: Original Controls Box.

Figures 11-12: The inside and outside of our final control box design.

Software

The boat uses two different softwares to achieve its goal. On the autonomy side is MOOS IvP. MOOS handles the autonomy of the boat and is tasked with taking in input and making decisions based on the boat's mission and surroundings. ROS is the other software and plays a role in the sensing side of things. ROS is primarily in charge of running the SICK LiDAR at the front of the boat and processing the data into something that MOOS can use to make important decisions. Using a hybrid system allows the boat to take advantage of the best parts of ROS and MOOS, creating a smarter and more robust system.

MOOS IvP

The Jetson Nano is running our code through a program called MOOS IvP. MOOS was written by Paul Newman in 2001 to support operation with autonomous marine vehicles in the MIT Ocean Engineering department [12]. It has then been constantly updated by various institutions including MIT, Oxford and the NOAA. MOOS stands for "Mission Oriented Operating Suite" and its primary purpose is to build highly capable autonomous systems. It is based on three main principles; publish and subscribe autonomy middleware, which means that there are various applications that will publish and subscribe to various data streams. Backseat driver paradigm, which means that vehicle autonomy is seperate from vehicle control. It's last principle is behavior based autonomy.

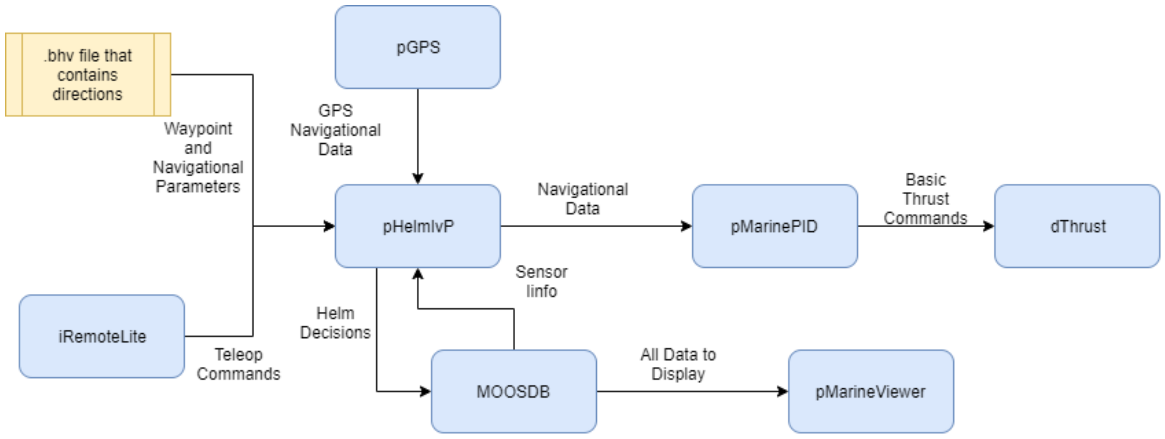
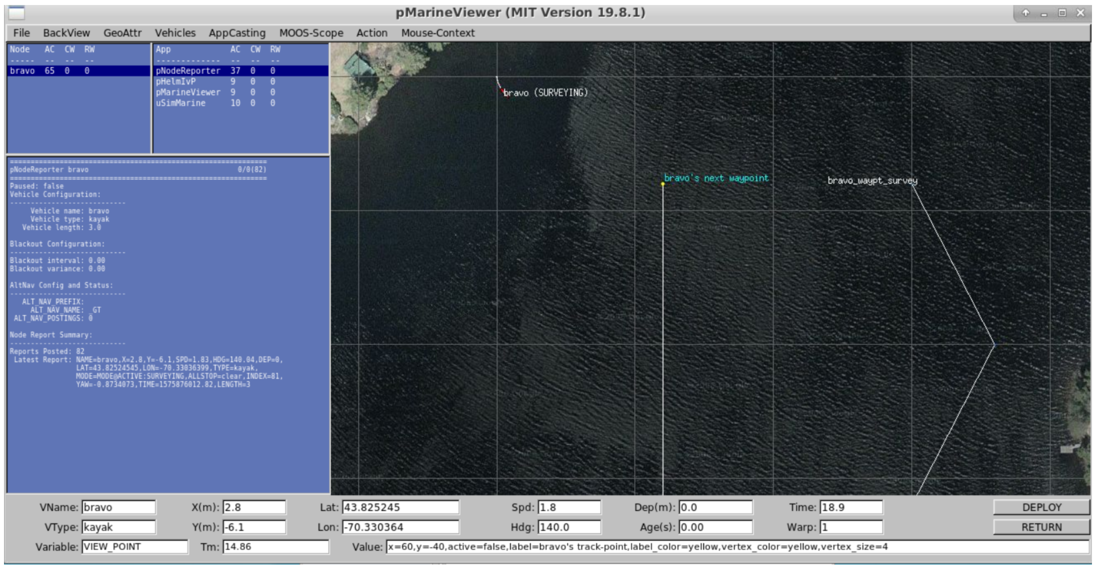


Figure 13: A visualization graph of how the boats nodes interact within MOOS

Differential Control and Serial Connections. In order for the boat to complete a journey, it needs to know where it's going and how to get there. For this MOOS took in GPS data that included position, heading and velocity, to understand where it was and where it was going. After this we wrote a driver that would take the generic MOOS control commands and convert them into differential motor control commands. This would then output two thrust commands -100 to 100, that was then sent over serial command to the Arduino Nano. The serial command is a connection between the Jetson and the Arduino via the USB connection. The arduino then mapped the thrust values to appropriate PWM commands to our motors. MOOS uses a PID controller to accurately control the motors to the proper position by reducing noise and rapid changes.

GPS. In order for the boat to complete a journey, it needs to know where it's going and how to get there. The boat will mainly be taking long outdoor journeys and needs a navigation system that will be reliable. We decided to answer these questions using GPS. GPS modules are affordable and can get reliable data in most outdoor locations. We achieved readings that are accurate within a few feet and were able to get other important information such as heading and speed by measuring the time and distance between consecutive GPS points. We are using GPS coordinates to tell the boat where to go. By comparing our desired coordinates to the boat's current ones, we can calculate which direction the boat should be facing and how far it needs to travel. Figure 14 shows how the GPS is read out through a graphic user interface that easily allows users to check the boat's status and set waypoints.


pMarineViewer (MIT Version 19.8.1)
File BackView GeoAttr Vehicles AppCasting MOOS-Scope Action Mouse-Context
bravo (SURVEYING)
bravo's next waypoint
bravo_waypt_survey
DEPLOY
RETURN

```
Node   AC  CW  RW
-----  --  --  --
bravo  65  0   0

App            AC  CW  RW
-------------  --  --  --
pNodeReporter  37  0   0
pHelmIvP       9   0   0
pMarineViewer  9   0   0
uSimMarine     10  0   0

==============================================================
pNodeReporter bravo                                    0/0(82)
==============================================================
Paused: false
Vehicle Configuration:
----------------------------
     Vehicle name: bravo
     Vehicle type: kayak
   Vehicle length: 3.0

Blackout Configuration:
----------------------------
Blackout interval: 0.00
Blackout variance: 0.00

AltNav Config and Status:
----------------------------
   ALT_NAV_PREFIX:
     ALT_NAV_NAME: _GT
 ALT_NAV_POSTINGS: 0

Node Report Summary:
----------------------------
Reports Posted: 82
 Latest Report: NAME=bravo,X=2.8,Y=-6.1,SPD=1.83,HDG=140.04,DEP=0,
                LAT=43.82524545,LON=-70.33036399,TYPE=kayak,
                MODE=MODE@ACTIVE:SURVEYING,ALLSTOP=clear,INDEX=81,
                YAW=-0.8734073,TIME=1575876012.82,LENGTH=3
```

VName:	bravo	X(m):	2.8	Lat:	43.825245	Spd:	1.8	Dep(m):	0.0	Time:	18.9
VType:	kayak	Y(m):	-6.1	Lon:	-70.330364	Hdg:	140.0	Age(s):	0.00	Warp:	1

Variable: VIEW_POINT Tm: 14.86 Value: x=60,y=-40,active=false,label=bravo's track-point,label_color=yellow,vertex_color=yellow,vertex_size=4

Figure 14: An example of the GPS readout with waypoints for an on water test.

pObstacleMgr. One of the most important MOOS nodes is called pObstacleMgr. It allows us to import Lidar data directly into the MOOSDB and have that data be interpreted by the software as an obstacle. The boat will then automatically calculate a new path around said obstacle and continue on its original path.

Obstacle Avoidance

A big part of this project involves competing in the SICK TiM$10K challenge to use LiDar technology in new and innovative ways. These sensors work by sending out thousands of light pulses and measuring the time it takes for each pulse to reflect back [13]. Each time is recorded and can easily be converted into distance measurements using equation 2 where D is the distance from the sensor to the object, C is the speed of light, and t is the total time it took the light pulse to return.

$$D = (C * t)/2 \tag{2}$$

The sensor reads the data as a point cloud that can be visualized a very pixelated version of the surroundings. The first step was using a software provided by SICK called SOPAS to get our sensors reading and producing a graphical visualization. Figure 15 below shows an example of the readout, each of the colored lines represents a surface that the scan bounced off.

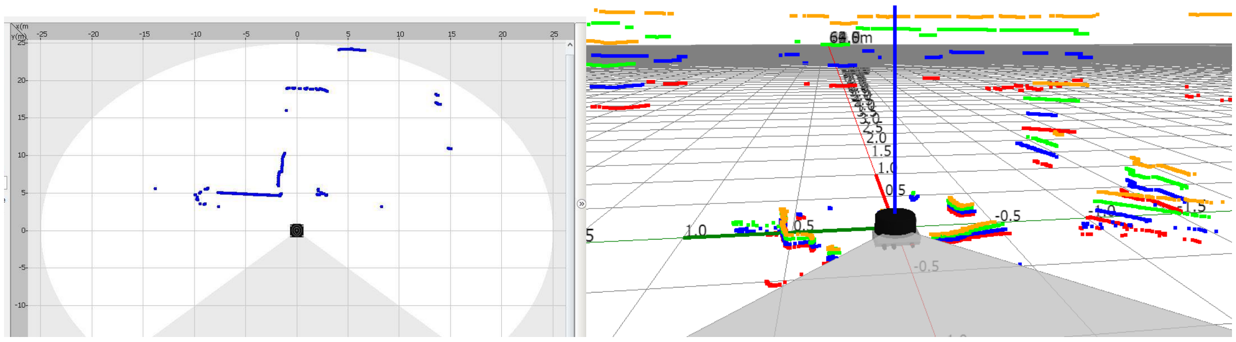


Figure 15. A graphical visualization from a scan of a room using our 2D(left) and 3D (right) LiDAR sensor, using the SICK SOPAS software.

Once we had the sensor reading data, we needed to integrate it into our system. The SICK TiM 781 series sensor is mounted to the front of the boat and is used to gather data about its surroundings. The data is then processed by a custom ROS node that cleans up any noise in the scans. The process the data goes through is explained in figure 16 below. Raw data is put through a 2D median filter that reduces noise and produces sharper edges. Similar points are then grouped together and assigned a unique obstacle ID.

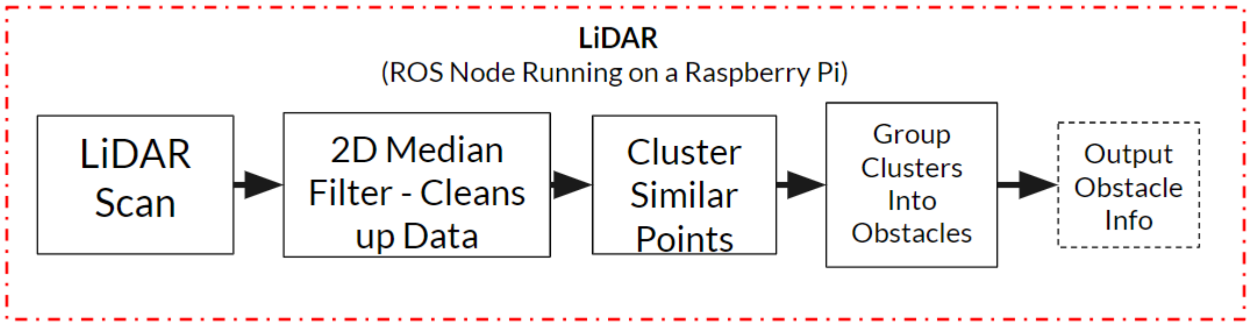

LiDAR
(ROS Node Running on a Raspberry Pi)
LiDAR Scan
2D Median Filter - Cleans up Data
Cluster Similar Points
Group Clusters Into Obstacles
Output Obstacle Info

Figure 16. The processing of LiDAR data by ROS Nodes.

We chose to use ROS for the LiDAR processing because it had preexisting drivers that allowed us to quickly and easily get the sensor running. Once the scan data is cleaned and clustered into groups, the result is published as a ROS topic. The ROS-MOOS bridge takes this topic and uses the scan data to our MOOSDB. Since MOOS only takes this data in terms of actual GPS coordinates, and we are given the data in a heading and distance from the boat, we must make necessary calculations to convert the data into usable data for MOOS. This is done by taking the boat's current heading and GPS position, and doing matrix math to produce an absolute obstacle position from the relative data. pObstacleMgr uses this data to calculate new routes for the boat, avoiding any obstacles that are in the way.

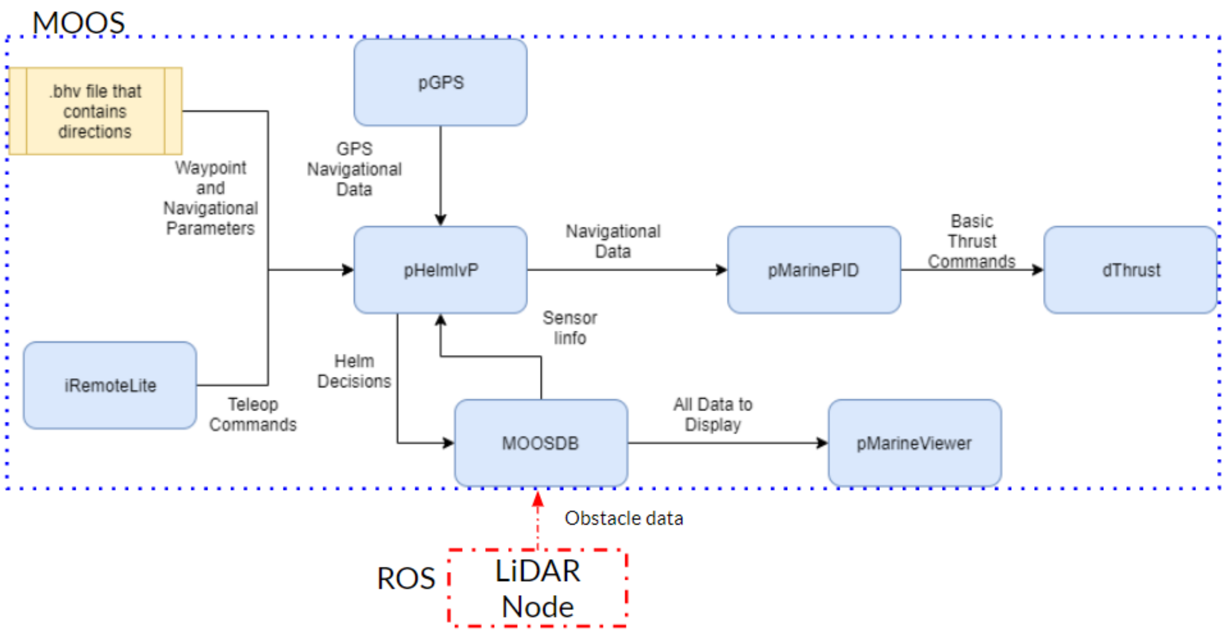


Figure 17. Where the LiDAR data fits into our MOOS Map.

Experimental Results

Throughout the year, the team performed multiple experimental tests on different parts of the boat as pieces were finished, new parts were ordered and models were verified.

First Water Test

Once the hull and a motor arrived, the team conducted a proof of concept test on the water. A teammate sat on the boat and steered by turning the motor and twisting the throttle manually. A baseline boat speed with one motor was calculated at approximately 3 knots. This confirmed that the boat would be capable of moving at reasonable speeds with two motors. Turning tests were also conducted to prove that the boat would be able to turn efficiently using differential thrust. These also proved successful.

Final Water Tests

Bluetooth and Wifi Control. The final prototype boat was tested on December 5th and 7th. The boat could be assembled and launched by a single person because it weighed 100 lbs and motors were easy to swing upward. On the 5th, the boat was driven around a set buoy course using a bluetooth keyboard as a control from a nearby canoe. Here the differential steering and speed of the boat were assessed. The motors were mounted close enough to the centroid of the hull that it could quickly rotate in place. The boat moved fast enough to outpace the canoe, but speed was not quantified. Later that day, a router was integrated into the setup that allowed for control of the boat from shore.

GPS Waypoint. On the 7th, the boat was launched with GPS waypoints set just past each of the bouy, such that it could follow the waypoints without running into the buoys. The navigation code successfully adjusted its course continuously to account for a strong crosswind that gusted 10 mph. The GPS software measured a maximum speed of 3 m/s, or 5.8 knots. While the speed was short of 10 knot goal, the current motors could easily be substituted for stronger ones if a research purpose necessitates it. The end result was a successful autonomous test.

Takeaways. The experimental tests successfully demonstrated the boats capability for unmanned navigation around buoys, and provided the team with important problems to address in the spring. There was a lag between remote controls and motor response. The prospect of LiDar obstacle avoidance was promising because the differential steering made the boat very maneuverable. Even with LiDar, the boat is a long way from being trusted unmonitored.

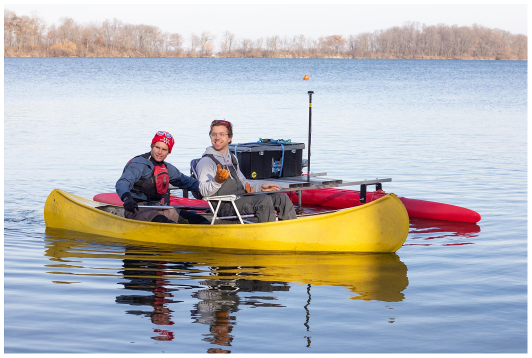


Figure 18: Bluetooth keyboard control of the boat

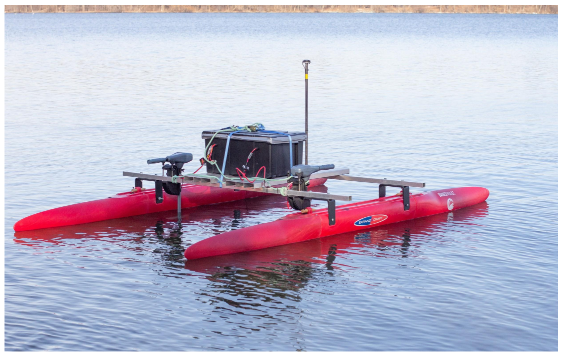
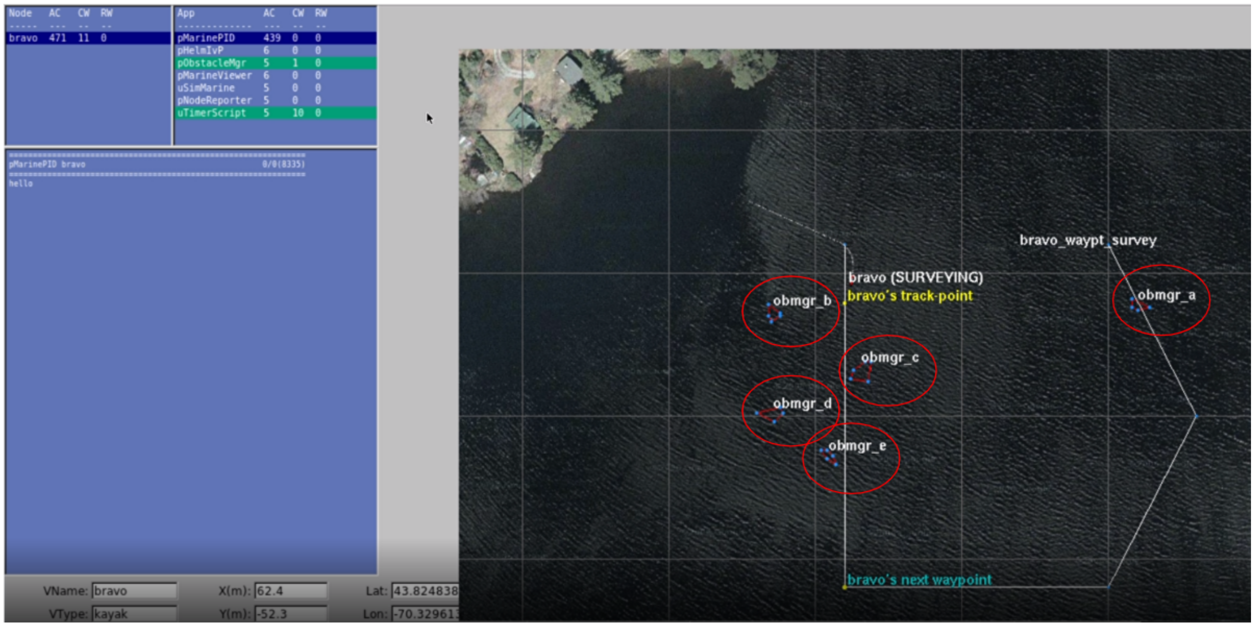


Figure 19: Boat controlled over WiFi from shore

Obstacle Avoidance Land Testing

Unfortunately, due to both the long winter and the COVID-19 crisis, on-water testing of our Lidar obstacle avoidance system was not possible. Instead, on land testing was successful in demonstrating that the Lidar could accurately cluster obstacles so that MOOS would be able to use them in its pObstacleMgr node. We believe that had on the water testing been possible, the boat would have been capable of obstacle avoidance.



```
Node   AC    CW   RW
-----  ---   --   --
bravo  471   11   0
```

```
App              AC    CW   RW
-------------    ---   --   --
pMarinePID       439   0    0
pHelmIvP         6     0    0
pObstacleMgr     5     1    0
pMarineViewer    6     0    0
uSimMarine       5     0    0
pNodeReporter    5     0    0
uTimerScript     5     10   0
```

```
==============================================================
pMarinePID bravo                                     0/0(8335)
==============================================================
hello
```

VName: bravo X(m): 62.4 Lat: 43.824838
VType: kayak Y(m): -52.3 Lon: -70.329613

Figure 20. The user interface showing pObstacleManager from our obstacle detection simulation with detected objects circled in red. The simulation shows that the boat is able to detect and avoid obstacles.

Accountability and Liability

The boat should be registered to the state of Wisconsin as a motorized craft less than 16ft in length. After mailing the Wisconsin registration application, the DRN should provide registration numbers, a validation sticker, and new hull identification numbers. It is very important to write "None-Homemade" under the hull identification so that the DNR provides us with original numbers. The registration numbers are 3" tall and must be placed on both sides of the bow, with the validation sticker in line. The hull identification numbers must be permanently attached to the transom, but do not need to be visible from a distance like the registration.

While the boat may not technically require registration if treated as an RC boat instead of a recreational vessel, the registration will be helpful if the boat gets lost and when it is sold or transferred. The boat can be considered remote controlled while testing within line of sight during the day, regardless of autonomy because the missions can be overridden in MOOS. Anyone controlling the boat is considered the reasonable skipper and must follow all boating rules in accordance with the Wisconsin DNR and Coast Guard [14]. If operating at night, the boat needs red and green lights on the port and starboard bow, and a white light in the middle or stern.

Long unaccompanied missions are a legal grey area because there are few maritime regulations on autonomy. If the boat's autonomy and safety protocol are unfit, the owners could face negligent or reckless operation litigation. Capt. DegInnocenti writes about his standard in "Marine autonomous vehicles and the law: Assessing risks and liability" published by Professional Mariner. Rule 23(d)(ii) implies that vessels slower than 7 knots and shorter than 7 meters in length are less dangerous than ones exceeding those parameters [15]. With that legal background, the author makes up a guideline that boats over 7 meter and faster than 7 knots need to have an on-board means of preventing collisions at sea[15]. The BadgerBoat is smaller and falls outside of this recommendation, but still has an on-board means of preventing collisions.

In Milwaukee's NPR article "On Lake Michigan, Almost Time for Autonomous Boats?" the author noted that "Some universities around the Great Lakes are also talking about sharing an autonomous vessel. But the U.S. Coast Guard is still working oen regulations for use [16]." Without regulations to follow, it is up the engineers to design and operate the boat safely. The standard for a design defect is that "the foreseeable risks of harm posed by the product could have been reduced or avoided by the adoption of a reasonable alternative design ... and the omission of the alternative design renders the product not reasonably safe." This is called the "risk-utility test[17]." A good way to defend from design defect litigation is by documenting all the tests. Since Mendota is active with swimmers and idiot boaters, the safest time to test long journeys is early in the morning.

Conclusion

Motivated by new contaminants found in the waterways and the desire to expand the uses of autonomous vehicles, the goal was to develop the BadgerBOAT into a tool that can improve our understanding of pollutants and keep our beaches clean and accessible. By limiting human interaction with harmful chemicals while decreasing man hours required to collect valuable data, hopefully the boat can be successful in this goal.

This year, the focus of the team was to build a solid base that could be improved on by other teams in the coming years. This was defined at the beginning of the year as a boat that could autonomously navigate between GPS waypoints and avoid obstacles that come into its path.

The BadgerBOAT was successfully created to achieve the goals which were defined at the start of the semester. It is fully capable of navigating GPS waypoints without input from any user. The LiDar sensors are capable of providing constantly updated obstacle data to the MOOS nodes that can process that into course corrections. Unfortunately, we were unable to test the obstacle avoidance algorithm on the water, but on land it did seem effective. Hopefully this testing can be done by another group in the future.

Appendix A: Design Requirements

The BadgerBoat project is a research project where the client posed a challenge without formal design specifications. Lennon Rogers wants to create an autonomous boat system that can navigate the local lakes and rivers to collect useful data. The specific way that the boat should help the community was up to the team. The proposed project was to create an autonomous, solar powered boat that detects and records dangerous blue-green algae. The boat will reduce man hours required to collect information, and provide continuous streams of data, hopefully leading to cleaner, safer lakes and rivers. This year-long challenge was broken down into five subsections of design specifications: boat construction, unmanned navigation, autonomy, collecting useful data, and multi-day journey. In the fall semester, the goal of BadgerBoat team 2 was to have the boat built and achieve unmanned navigation.

The client's vision for the BadgerBoat can be broken down into five sub-sections, each with their own engineering requirements. While parts of the subsections can be worked on in parallel, the list below shows the order in which the design requirements were and will be completed.

Boat Construction

The boat must float. The frame must support the weight of the blackbox containing batteries and electronics. The electronics must be shielded from water. The client must be able to easily store the boat.

Unmanned Navigation

The boat must steer without assistance to follow GPS waypoints.

Autonomous

The boat must create a new path to the GPS waypoint once an obstacle is sensed.

Collect Useful Data

The boat must be able to record useful water quality data. The proposed measurement tool are listed below in table B-1.

Multi-Day Journey

The boat must regenerate power for multi-day journey.

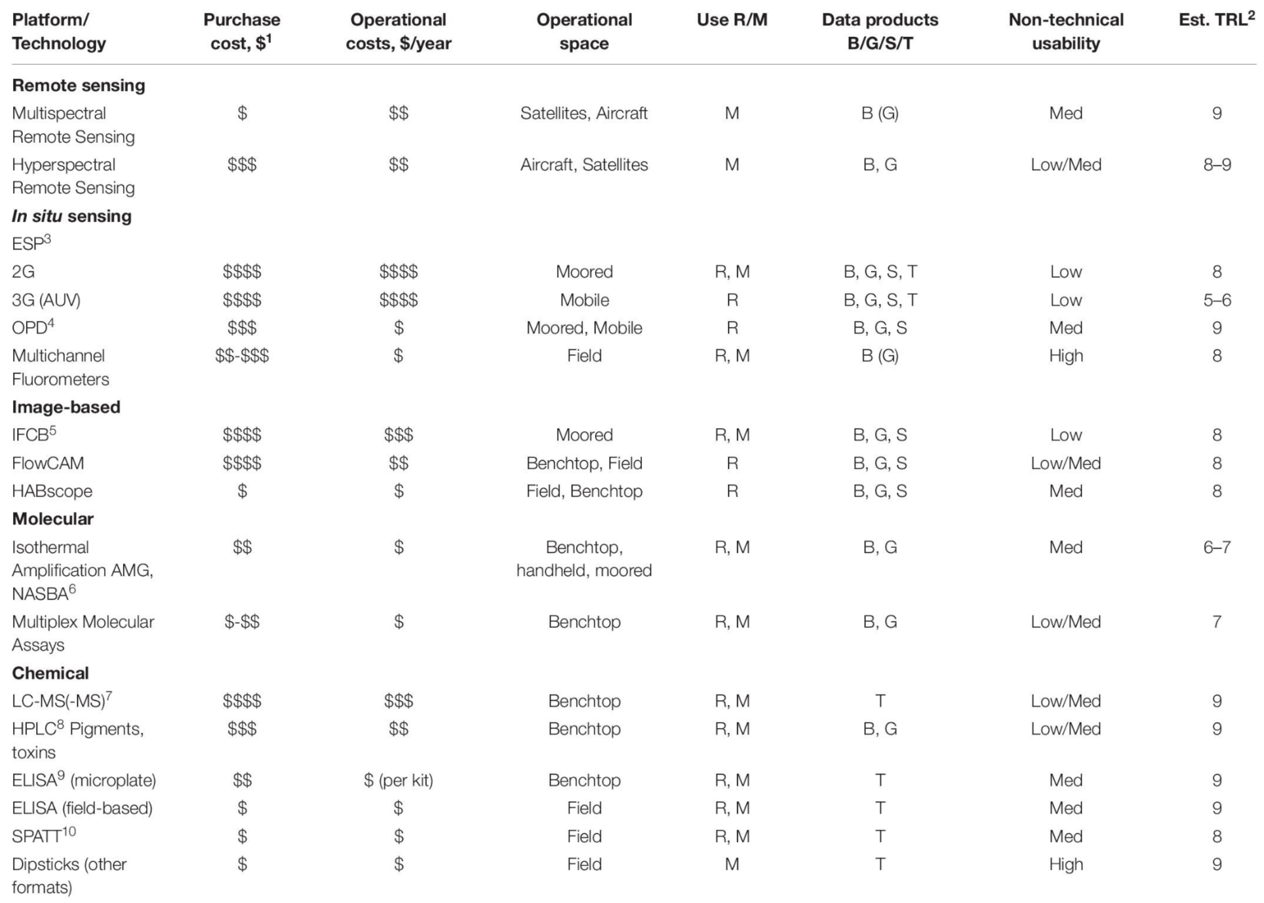
Appendix B: Harmful Algal Blooms

Over the summer, the Yahara waterways of Wisconsin turn bright green, foamy, and foul smelling from algal blooms. Run-off from farms in the watershed wash fertilizers and animal waste into the lake, which feed algae and cyanobacteria. This influx of a limiting nutrient is called eutrophication. Harmful algal blooms, or HABs, have widespread effects on the ecosystem and people's interaction with the water. They starve the water of oxygen and can produce deadly cyanotoxins when they decay. In slow-moving water, oxygen dissolves from the atmosphere easily and the algae bloom faster undisturbed. The algae effectively create a dead zone because they block out light from the aquatic plants and suffocate fish from the lack of dissolved oxygen. The University of Wisconsin, Madison Limnology Department advises people and pets to stay out of the water when it looks green as a matter of public safety because cyanotoxins which attack the nervous system.

The United States Geological Survey is a scientific agency that monitors waterways in Wisconsin and updates a website with temperatures and toxicity levels. The website run by the USGS is called INFOSyahara and has webcams at the major dams and outlets in the Yahara waterways. The Limnology Department also measures water quality at the Hessler Lab and at a buoy in the center of Lake Mendota, where data is used to study the lake's ecology.

The BadgerBoat will be an invaluable tool for studying the waterways. The lightweight boat can navigate shallow waters inaccessible to manned boats and where there is not easy access by land. The vessel also saves unpleasant man-hours spent in foul waters with dead animals or sewage treatment plant ponds. The data collected can be used directly by scientists in the Hessler Lab, or be an asset to the INFOSyahara website as an extra webcam sent to at risk areas. As of right now, the Limnology Department issues warnings based on the color or the water, not evidence of the toxins.

In the future, the boat will be outfitted measurement tools to gather data and possibly test for dangerous levels of algae or toxins. While temperature and dissolved oxygen probes are and indirect measurement of algae levels, the probes are cheaper and easier to implement on an autonomous boat. These will be added to the BadgerBoat in the spring. The stretch goal of the project is to use a water collection system or on-board test that directly measures algae in a field test. Table B-1 shows the current state of technology that can test for HABs according to "Considerations in Harmful Algal Bloom Research and Monitoring," published in Frontiers in Marine Science. This survey of technology will be used to determine which measurement system can be applied to the autonomous boat. The goal in the spring is to have basic measurements like temperature and dissolved oxygen operational, and a plan in place to add additional technologies.



Platform/ Technology	Purchase cost, $[1]	Operational costs, $/year	Operational space	Use R/M	Data products B/G/S/T	Non-technical usability	Est. TRL[2]
Remote sensing							
Multispectral Remote Sensing	$	$$	Satellites, Aircraft	M	B (G)	Med	9
Hyperspectral Remote Sensing	$$$	$$	Aircraft, Satellites	M	B, G	Low/Med	8–9
***In situ* sensing**							
ESP[3]							
2G	$$$$	$$$$	Moored	R, M	B, G, S, T	Low	8
3G (AUV)	$$$$	$$$$	Mobile	R	B, G, S, T	Low	5–6
OPD[4]	$$$	$	Moored, Mobile	R	B, G, S	Med	9
Multichannel Fluorometers	$$-$$$	$	Field	R, M	B (G)	High	8
Image-based							
IFCB[5]	$$$$	$$$	Moored	R, M	B, G, S	Low	8
FlowCAM	$$$$	$$	Benchtop, Field	R	B, G, S	Low/Med	8
HABscope	$	$	Field, Benchtop	R	B, G, S	Med	8
Molecular							
Isothermal Amplification AMG, NASBA[6]	$$	$	Benchtop, handheld, moored	R, M	B, G	Med	6–7
Multiplex Molecular Assays	$-$$	$	Benchtop	R, M	B, G	Low/Med	7
Chemical							
LC-MS(-MS)[7]	$$$$	$$$	Benchtop	R, M	T	Low/Med	9
HPLC[8] Pigments, toxins	$$$	$$	Benchtop	R, M	B, G	Low/Med	9
ELISA[9] (microplate)	$$	$ (per kit)	Benchtop	R, M	T	Med	9
ELISA (field-based)	$	$	Field	R, M	T	Med	9
SPATT[10]	$	$	Field	R, M	T	Med	8
Dipsticks (other formats)	$	$	Field	M	T	High	9

Table B-1: Summary of existing technologies currently used to measure HAB biomass, toxins, indicating relative purchase and operational costs, operational space (i.e., benchtop versus moored technologies), effectiveness of use for research (R) versus monitoring (M), data products measured (B: biomass, G: genus, S: species, T: toxin).

Appendix C: EES Code

Before making final design decisions and purchases, and analytical model of the boat was created. The goal of this model was to simulate the fluid dynamics of the boat and predict its behavior with different configurations and options.

```
"Badger BOAT EES Model"

"Boat Dimensions"
{M_boat = 150 [kg]    "Mass of boat guess"}
{m_boat = M_motors + M_batteries + M_solar + M_aluminum_tube + M_Aluminum_decking + M_risers + M_Payload +
M_computer + m_hull_total}
L_boat = 12*convert(ft,m)    "Length of boat"
W_boat = 8*convert(ft,m)    "Width of boat"
H_boat = 1.5*convert(ft,m)    "Total height of boat"
width_hull = 10*convert(in,m)                          "width of hull"
height_hull = 10*convert(in,m)                         "height of hull"

"Boat Components"

"Solar Panels"
n_solarp = 4    "Number of solar Panels"
Panel_mass = 1.9 [kg]

"Motors"
n_motors = 2    "Number of motors"
motor_mass = 17.2 [kg]    "Mass of one motor"
T_motor  = 35*convert(lbf,N)       "Thrust of each motor"
F_thrust = n_motors*T_motor    "Total thrust from motors"

"Batteries"
{n_batteries = 6}
n_batteries = 2                                                                                          "using two
batteries for the Fall Test"
Battery_mass = 6.4 [kg]
Bat_cap = 40*60 [Coulombs]    "Capacity of each battery"

"Frame Bars"
N_alum_tubes = 2    "Number of aluminum frame tubes"
Length_tubes = 8*convert(ft,m)    "Length of each aluminum frame tube"
mass_tube=1 [kg]    "Mass of each tube based on length"

"Frame decking"
Area_decking=8.167 [ft^2]    "Surface area of aluminum deck"
Th_decking = .01 [ft]
rho_aluminum = density(Aluminum, T=25 [C])*convert(kg/m^3, lbm/ft^3)
mass_decking = Area_decking*rho_aluminum*th_decking*convert(lbm,kg)    "Mass of deck based on surface area"

"Risers"
N_risers = 4    "Number of risers"
Mass_riser= 1.8 [kg]

"Power Systems/Power Draw"
P_solar = 49 [W]    "Power from each solar panel"
P_solar_total=P_solar*n_solarp    "Total power input from solar panels"
P_motors = 50 [A]    "Power draw of each motor"
P_senors=1    "Sensor Power Draw"

"Boat weights"
M_motors = n_motors*motor_mass
M_batteries = n_batteries * Battery_mass
M_solar = n_solarp * Panel_mass
M_aluminum_tube = mass_tube*N_alum_tubes
M_Aluminum_decking = mass_decking
M_risers = Mass_riser*N_risers
M_Payload = 6.9[kg]
M_computer = 7[kg]
m_hull_middle = 11*convert(lbm,kg)
m_hull_bow = 7.5*convert(lbm,kg)
```

```
m_hull_total = 4*m_hull_bow + 2*m_hull_middle


A_tboat = 96*convert(ft^2,m^2)    "Total surface area of the boat"


X_wspeed = 10.8 [mph]    "Average wisconsin wind speed in fall"
Y_wspeed = 10.8 [mph]    "Average wisconsin wind speed in fall"
c_motors =1    "Motor configuaration"


 u= 3 [m/s]                                   "from experimental results"
Boat_speed = u*convert(m/s,mph)

"Drag Forces"
m_boat = 2*rho_water*Volume_submerged         "volume displaced per hull"
volume_submerged = .5*L_boat*height_submerged*width_submerged                 "volume of
submerged triangular prism"
width_hull/height_hull = width_submerged/height_submerged                     "law of signs
for height/width relationship"
hyp_submerged = (height_submerged^2 + (.5*width_submerged)^2)^.5              "hypotenuse of
submerged triangular cross section"
A_wboat = 4*L_boat*hyp_submerged                                              "total wetted
surface of submerged triangular prism model"

fr = .005                                     "fricton factor assumption from dress ~1/(Re^.25)"
rho_water = density(Water, T=25[C], P=101.3 [kPa])
Tau_max = .5*rho_water*fr*u_max^2             "Shear from water"
F_d_viscous_max = tau_max*a_wboat
F_d_total_max= 1.1*f_d_viscous_max            "Viscous drag makes up ~90% of total drag in most boats -Dress"

Tau = .5*rho_water*fr*u^2                     "Shear from water"
F_d_viscous = tau*a_wboat
F_d_total= 1.1*f_d_viscous                    "Viscous drag makes up ~90% of total drag in most boats -Dress"

F_d_total_max = F_thrust

"fall test"
m_boat = m_boat_fall
m_boat_fall = m_motors + battery_mass*2 + M_aluminum_tube
```

SOLUTION

Unit Settings: SI C kPa kJ mass deg

```
Area_decking = 8.167 [ft^2]                   A_tboat = 8.919 [m^2]
A_wboat = 1.9 [m^2]                           Battery_mass = 6.4 [kg]
Bat_cap = 2400 [coulombs]                     Boat_speed = 6.711 [mph] {3 [m/s]}
c_motors = 1 [dim]                            fr = 0.005 [dim]
F_d,total = 46.88 [N]                         F_d,total,max = 311.4 [N]
F_d,viscous = 42.62 [N]                       F_d,viscous,max = 283.1 [N]
F_thrust = 311.4 [N]                          height_hull = 0.254 [m]
height_submerged = 0.1162 [m] {4.573 [in]}    hyp_submerged = 0.1299 [m]
H_boat = 0.4572 [m]                           Length_tubes = 2.438 [m]
L_boat = 3.658 [m]                            mass_decking = 6.243 [kg]
Mass_riser = 1.8 [kg]                         mass_tube = 1 [kg]
motor_mass = 17.2 [kg]                        M_Aluminum,decking = 6.243 [kg]
M_aluminum,tube = 2 [kg]                      M_batteries = 12.8 [kg]
m_boat = 49.2 [kg] {108.5 [lbm]}              m_boat,fall = 49.2 [kg]
M_computer = 7 [kg]                           m_hull,bow = 3.402 [kg]
```

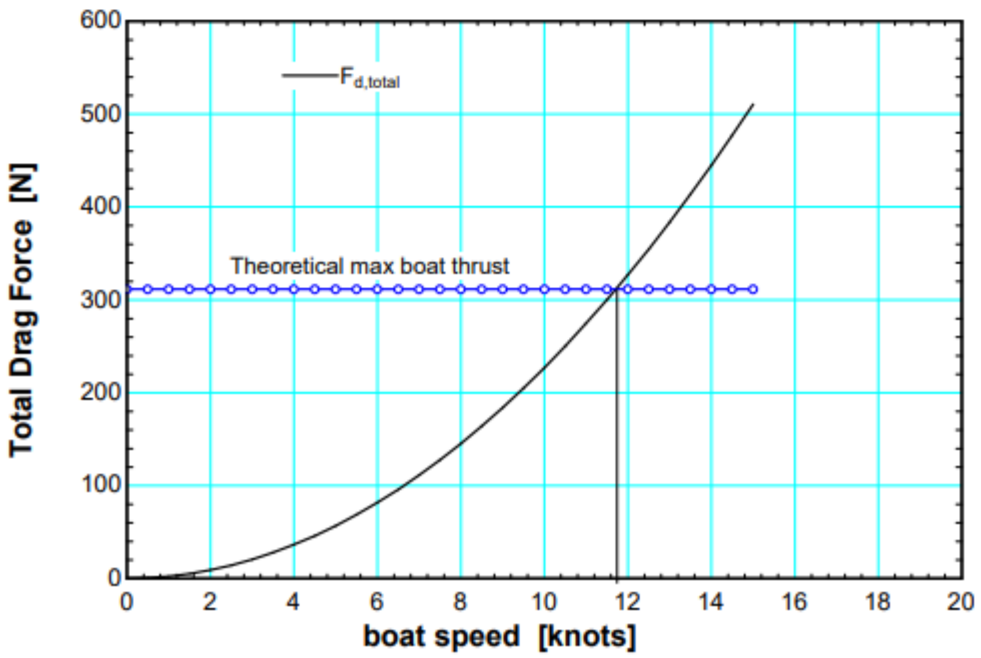
$m_{hull,middle}$ = 4.99 [kg]	$m_{hull,total}$ = 23.59 [kg]
M_{motors} = 34.4 [kg]	$M_{Payload}$ = 6.9 [kg]
M_{risers} = 7.2 [kg]	M_{solar} = 7.6 [kg]
$N_{alum,tubes}$ = 2	$n_{batteries}$ = 2
n_{motors} = 2	N_{risers} = 4 [dim]
n_{solarp} = 4 [dim]	$Panel_{mass}$ = 1.9 [kg]
P_{motors} = 50 [A]	P_{senors} = 1 [W]
P_{solar} = 49 [W]	$P_{solar,total}$ = 196 [W]
$\rho_{aluminum}$ = 168.5 [lbm/ft^3]	ρ_{water} = 997 [kg/m^3]
τ = 22.43 [Pa]	τ_{max} = 149 [pa]
$Th_{decking}$ = 0.01 [ft]	T_{motor} = 155.7 [N]
u = 3 [m/s] {6.711 [mph]}	u_{max} = 7.731 [m/s] {17.29 [mph]}
$Volume_{submerged}$ = 0.02467 [m^3] {24.67 [L]}	$width_{hull}$ = 0.254 [m]
$width_{submerged}$ = 0.1162 [m]	W_{boat} = 2.438 [m]
X_{wspeed} = 10.8 [mph]	Y_{wspeed} = 10.8 [mph]

No unit problems were detected.

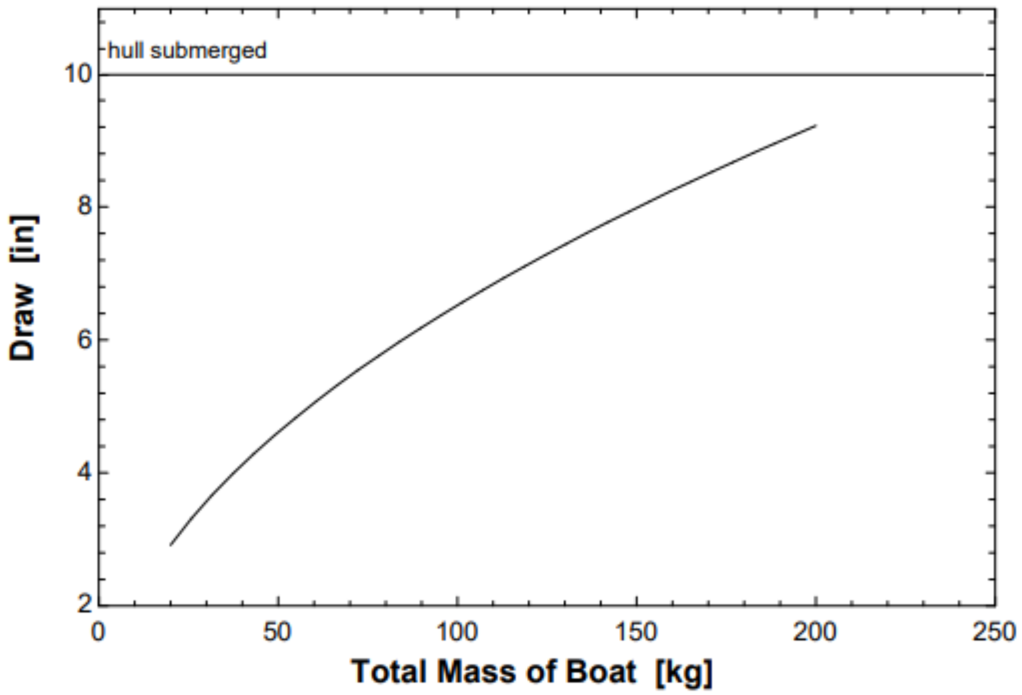
KEY VARIABLES

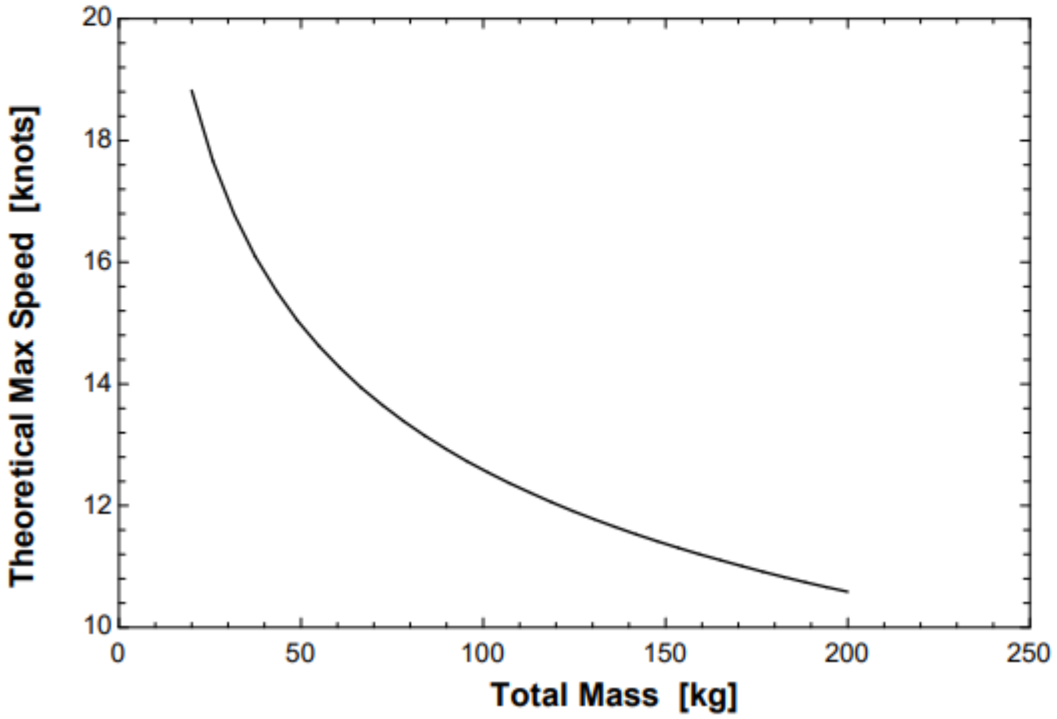
$Boat_{speed}$ = 6.711 [mph] {3 [m/s]}	*Experimental max speed*
$F_{d,total,max}$ = 311.4 [N]	*Max drag found with force balance from thrust*
u_{max} = 7.731 [m/s] {17.29 [mph]}	*Max speed found with force balance from thrust and drag*
$F_{d,viscous}$ = 42.62 [N]	*Calculated drag from experimental speed*

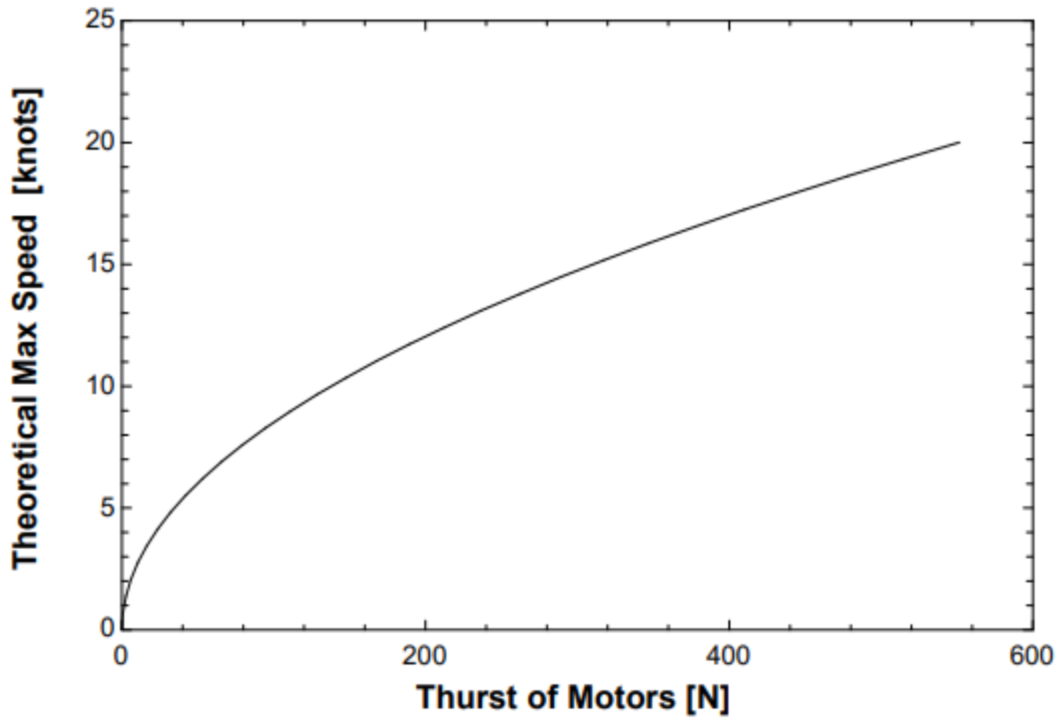
–

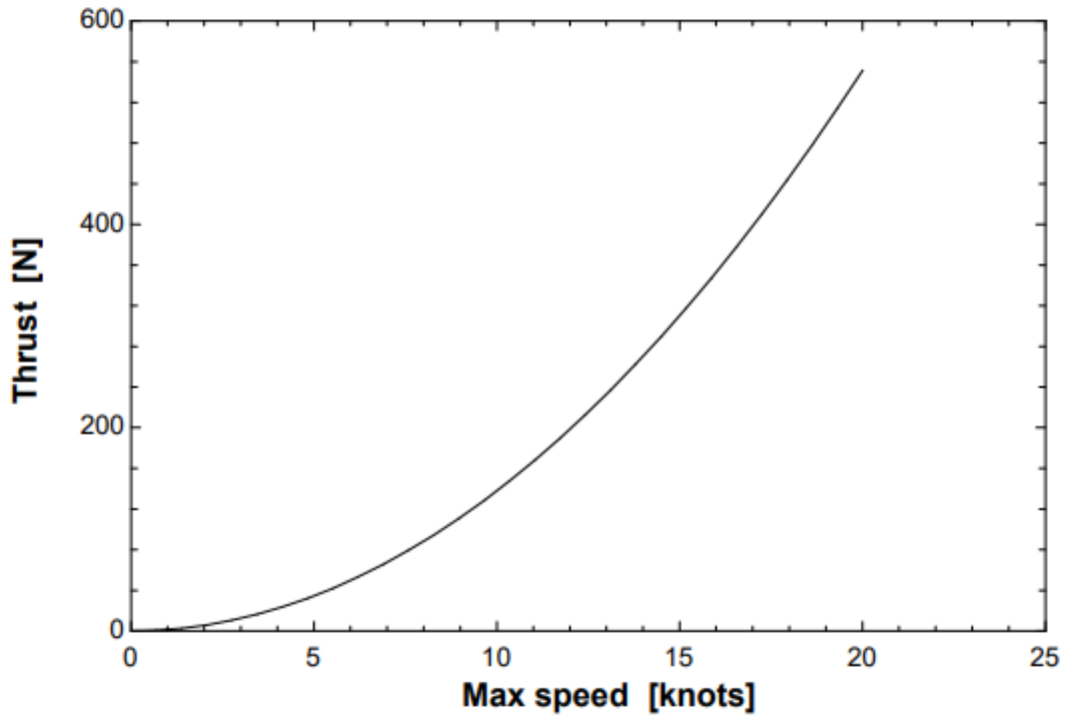


–


hull submerged
Draw [in]
Total Mass of Boat [kg]
0
50
100
150
200
250
2
4
6
8
10


Theoretical Max Speed [knots]
Total Mass [kg]
0
50
100
150
200
250
10
12
14
16
18
20

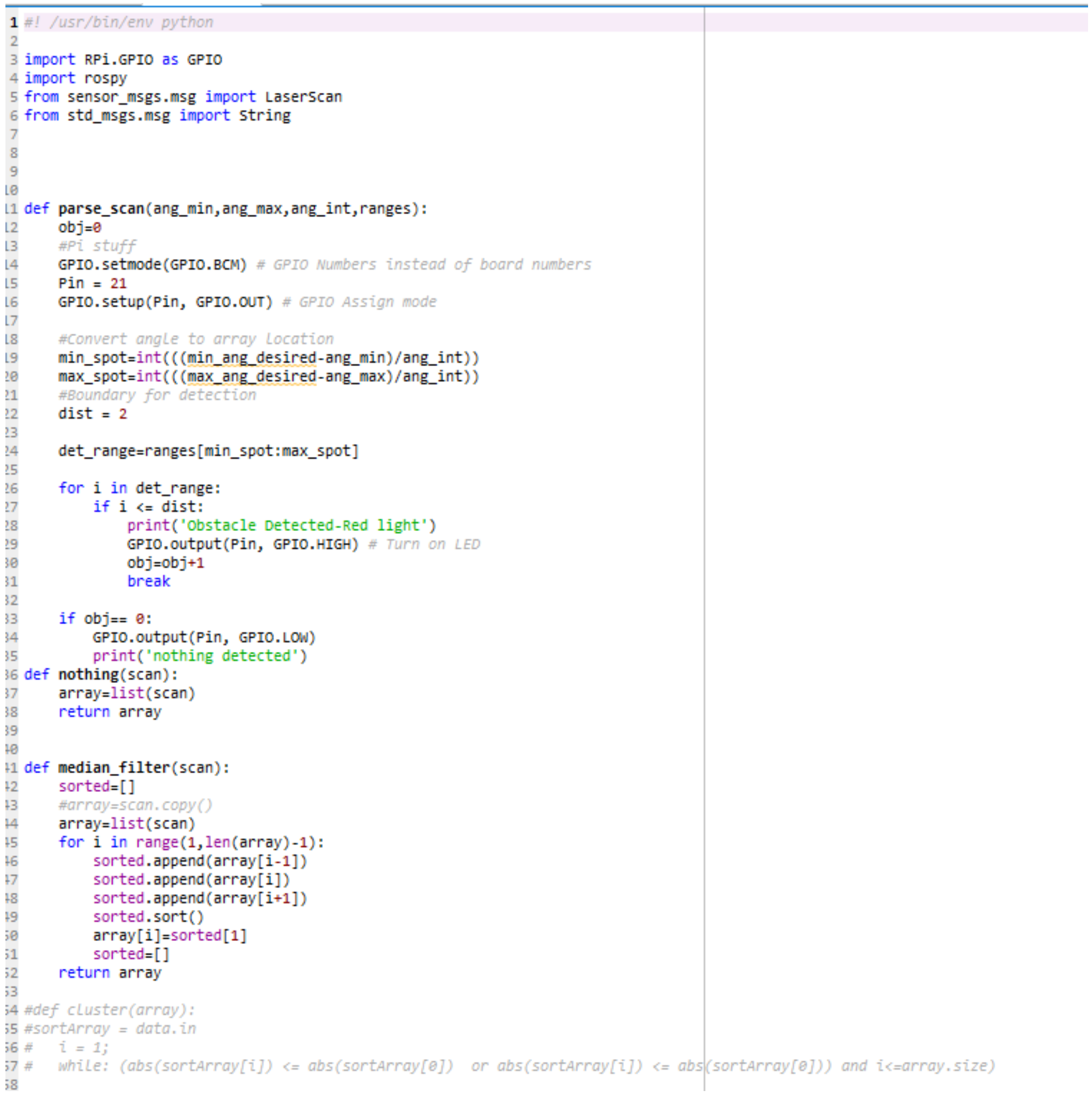




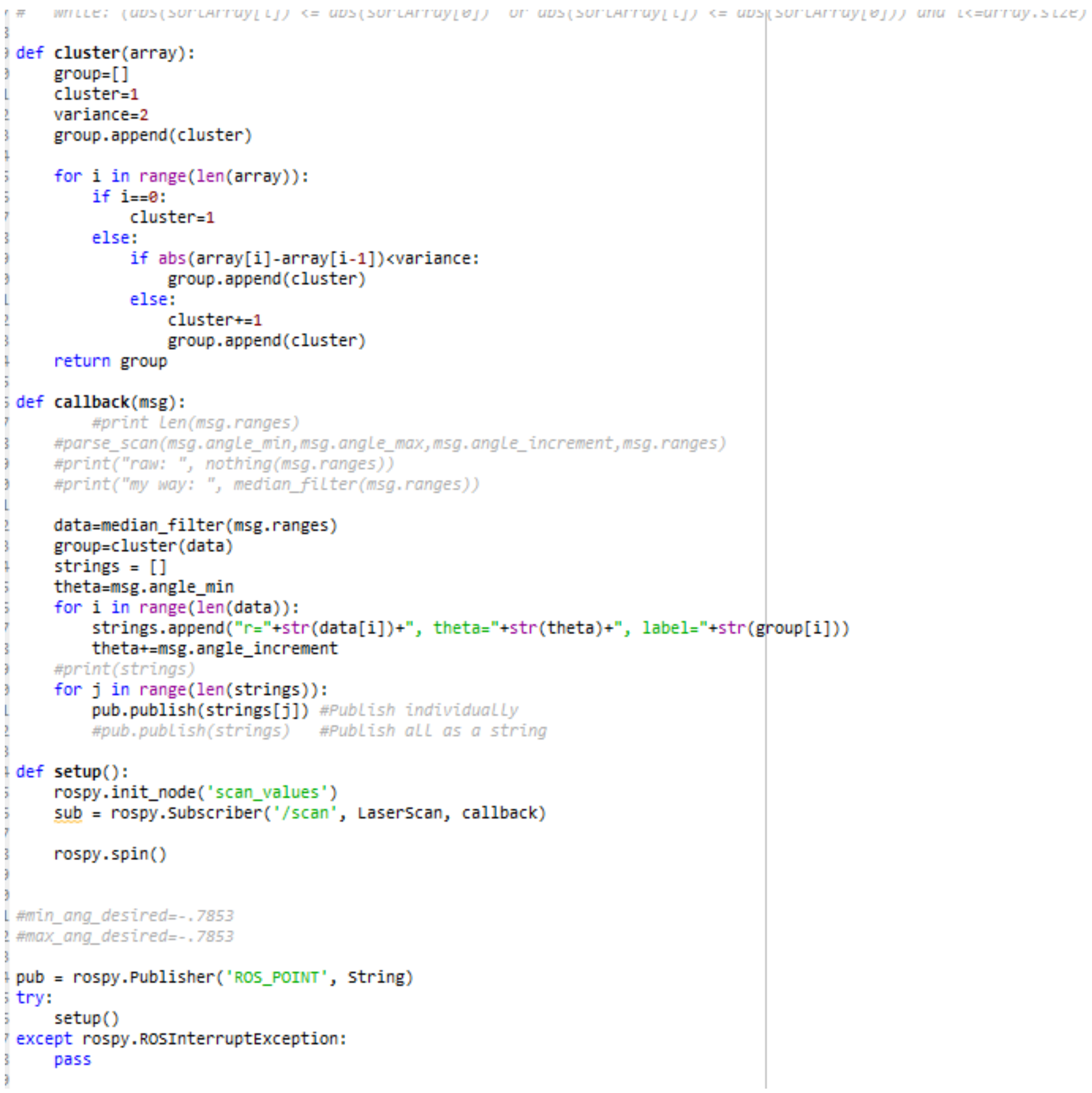
The **Figure C-1.**EES code and resulting graphs upon which the boat design was based.

Appendix D: LiDar Code

This code reads in the raw data from the LiDar sensor through a message called sensor_msgs.msg. The message sends information in what is known as a point cloud. This is interpreted as a list with an integer value for every pulse the LiDar sends out. The integer value represents the distance from the sensor to the object the pulse bounced off of. The data is taken in and filtered through a 2D median filter which reduces noise in the data. The median filter achieves this by comparing each data point to its neighbors and replacing the initial point with the median. This is a filtering technique commonly used in photography to sharpen edges. A clustering algorithm is then used to compare data points and sort them into obstacle groups, each with a unique ID.



```
#! /usr/bin/env python

import RPi.GPIO as GPIO
import rospy
from sensor_msgs.msg import LaserScan
from std_msgs.msg import String




def parse_scan(ang_min,ang_max,ang_int,ranges):
    obj=0
    #Pi stuff
    GPIO.setmode(GPIO.BCM) # GPIO Numbers instead of board numbers
    Pin = 21
    GPIO.setup(Pin, GPIO.OUT) # GPIO Assign mode

    #Convert angle to array location
    min_spot=int(((min_ang_desired-ang_min)/ang_int))
    max_spot=int(((max_ang_desired-ang_max)/ang_int))
    #Boundary for detection
    dist = 2

    det_range=ranges[min_spot:max_spot]

    for i in det_range:
        if i <= dist:
            print('Obstacle Detected-Red light')
            GPIO.output(Pin, GPIO.HIGH) # Turn on LED
            obj=obj+1
            break

    if obj== 0:
        GPIO.output(Pin, GPIO.LOW)
        print('nothing detected')
def nothing(scan):
    array=list(scan)
    return array


def median_filter(scan):
    sorted=[]
    #array=scan.copy()
    array=list(scan)
    for i in range(1,len(array)-1):
        sorted.append(array[i-1])
        sorted.append(array[i])
        sorted.append(array[i+1])
        sorted.sort()
        array[i]=sorted[1]
        sorted=[]
    return array

#def cluster(array):
#sortArray = data.in
#    i = 1;
#    while: (abs(sortArray[i]) <= abs(sortArray[0])  or abs(sortArray[i]) <= abs(sortArray[0])) and i<=array.size)

```



```
#    while: (abs(sortArray[i]) <= abs(sortArray[0])  or abs(sortArray[i]) <= abs(sortArray[0])) and i<=array.size)

def cluster(array):
    group=[]
    cluster=1
    variance=2
    group.append(cluster)

    for i in range(len(array)):
        if i==0:
            cluster=1
        else:
            if abs(array[i]-array[i-1])<variance:
                group.append(cluster)
            else:
                cluster+=1
                group.append(cluster)
    return group

def callback(msg):
        #print len(msg.ranges)
    #parse_scan(msg.angle_min,msg.angle_max,msg.angle_increment,msg.ranges)
    #print("raw: ", nothing(msg.ranges))
    #print("my way: ", median_filter(msg.ranges))

    data=median_filter(msg.ranges)
    group=cluster(data)
    strings = []
    theta=msg.angle_min
    for i in range(len(data)):
        strings.append("r="+str(data[i])+", theta="+str(theta)+", label="+str(group[i]))
        theta+=msg.angle_increment
    #print(strings)
    for j in range(len(strings)):
        pub.publish(strings[j]) #Publish individually
        #pub.publish(strings)   #Publish all as a string

def setup():
    rospy.init_node('scan_values')
    sub = rospy.Subscriber('/scan', LaserScan, callback)

    rospy.spin()


#min_ang_desired=-.7853
#max_ang_desired=-.7853

pub = rospy.Publisher('ROS_POINT', String)
try:
    setup()
except rospy.ROSInterruptException:
    pass
```

Figure D-1. The python script that reads in the LiDar data and interprets it.

Appendix E: MOOS Mission

This is the mission file in which we set waypoints, speed, how to interact with waypoint reaching and more. The file is Generic for MOOS as most of the fine tuning parameters are spread out across the MOOS file system, found here: https://github.com/waddellt15/badgerboat

```
//--------    FILE: badgersea1.bhv   -------------

initialize   DEPLOY = false
initialize   RETURN = false

//-----------------------------------------------
```

```
Behavior = BHV_Waypoint
{
  name      = waypt_survey
  pwt       = 100
  condition = RETURN = false
  condition = DEPLOY = true
  endflag   = RETURN = true

  updates    = WPT_UPDATE
  perpetual  = true

               lead = 8
        lead_damper = 1
      lead_to_start = true
              speed = 3.5   // meters per second
       capture_line = true
     capture_radius = 5.0
        slip_radius = 15.0
             efficiency_measure = all

            polygon = 59.4,75.5:9.8,79.8:10.1,4.8

              order = normal
            repeat  = 100000

    visual_hints = nextpt_color=yellow
    visual_hints = nextpt_vertex_size=8
    visual_hints = nextpt_lcolor=gray70
    visual_hints = vertex_color=dodger_blue, edge_color=white
    visual_hints = vertex_size=5, edge_size=1
}

//----------------------------------------------
Behavior=BHV_Waypoint
{
  name       = waypt_return
  pwt        = 100
  condition  = RETURN = true
  condition  = DEPLOY = true
  perpetual  = true
  updates    = RETURN_UPDATE
  endflag    = RETURN = false
  endflag    = DEPLOY = false
  endflag    = MISSION = complete

           speed = 3.0
  capture_radius = 2.0
     slip_radius = 3.0
          points = 10.1,4.8
 }

//----------------------------------------------
Behavior=BHV_ConstantSpeed
{
  name       = const_speed
  pwt        = 200
  condition  = SPD=true
  condition  = DEPLOY = true
  perpetual  = true
  updates    = SPEED_UPDATE
  endflag    = SPD = false

     speed = 4
                        duration = 10
                        duration_reset = CONST_SPD_RESET=true

}
```

Appendix F: GPSD

This GPSD driver received gps data from the GPSD system. It then properly adjusted any needed variables and outputted them as current heading, speed, latitude, longitude and x/y referenced to the origin of the mission. This driver was the only interface for the robot to know where it was within the world and how to move.

```
 //---------------------------------------------------------
// Procedure: OnConnectToServer

bool GPSd::OnConnectToServer()
{
   registerVariables();
   return(true);
}


bool GPSd::Iterate()
{
  AppCastingMOOSApp::Iterate();
  GeodesySetup();
#if GPSD_API_MAJOR_VERSION >= 5

      gps_data_t *p_gpsdata = p_gpsd_receiver->read();
#else
    gps_data_t *p_gpsdata = p_gpsd_receiver->poll();
#endif
  p_gpsdata = p_gpsd_receiver->read();

  //m_buf << p_gpsd_receiver->data();     // grab the data buffer
  //cerr << "************************************************************" << endl;
  //cerr << "Got buffer: " << endl;
  //cerr << "************************************************************" << endl;
  //cerr << m_buf.str() << endl;
  //cerr << p_gpsd_receiver->data() << endl;
  //cerr << "************************************************************" << endl;
  if ((p_gpsdata != NULL) && (p_gpsdata->set))  {
    m_gps_mode              = p_gpsdata->fix.mode;
    m_gps_lat               = p_gpsdata->fix.latitude;
    m_gps_lon                = p_gpsdata->fix.longitude;
    m_gps_alt               = p_gpsdata->fix.altitude;
    m_gps_spd                 = p_gpsdata->fix.speed;
    m_gps_head                 = p_gpsdata->fix.track;
//Used for conversion from earth to local
  convertLL = m_geodesy.LatLong2LocalUTM(m_gps_lat,m_gps_lon,m_ny,m_nx);
  if (!convertLL) {
    reportConfigWarning("could not convert variables");
    return false; }
//
    Notify("zGeo_X",                  m_nx);
    Notify("zGeo_Y",                  m_ny);
    Notify("GPSD_Mode",      m_gps_mode);
    Notify("NAV_HEADING",      m_gps_head);
    Notify("NAV_LAT",  m_gps_lat);
    Notify("NAV_LONG",  m_gps_lon);
    Notify("GPSD_elevation",  m_gps_alt);
    Notify("NAV_SPEED",      m_gps_spd);
    Notify("NAV_X",          m_nx);
    Notify("NAV_Y",          m_ny);
    m_json_output = p_gpsd_receiver->data();
    Notify("GPSD_json", m_json_output);
  }

  p_gpsd_receiver->clear_fix();

  AppCastingMOOSApp::PostReport();
  return(true);
}
```

Appendix G: Differential Thrust

This driver would edit the default MOOS controls of thrust and rudder positions to left and right motor thrust values for our differential thrust configuration. The code below is a portion of the driver that edited the mentioned values.

```
bool dfThrust::ThrustRudderToLR()
{
  // 1. Constrain Values
  //      DESIRED_RUDDER value to MAX_RUDDER
  //         - Anything more extreme than +/-50.0 is turn-in-place
  //      DESIRED_THRUST value to MAX_THRUST
  //         - Anything greater than +/-100.0% makes no sense
  double desiredRudder = clamp (m_des_rudder, (-1.0 * m_dMaxRudder), m_dMaxRudder);
  double desiredThrust = clamp (m_des_thrust, (-1.0 * MAX_THRUST), MAX_THRUST);

  // 2. Calculate turn
  //      - ADD rudder to left thrust
  //      - SUBTRACT rudder from right thrust
  double percentLeft  = desiredThrust + desiredRudder;
  double percentRight = desiredThrust - desiredRudder;

  // 3. Map desired thrust values to motor bounds
  //      - Range of DESIRED_THRUST: [-MAX_THRUST, MAX_THRUST]
  //      -          ...map to...
  //      - Range of valid thrust values: [-m_MaxThrustValue, m_MaxThrustValue]
  double fwdOrRevL   = (percentLeft  > 0.0) ? 1.0 : -1.0;
  double fwdOrRevR   = (percentRight > 0.0) ? 1.0 : -1.0;
  double pctThrustL  = fabs(percentLeft)  / MAX_THRUST;
  double pctThrustR  = fabs(percentRight) / MAX_THRUST;
  double mappedLeft  = pctThrustL * m_dMaxThrust * fwdOrRevL;
  double mappedRight = pctThrustR * m_dMaxThrust * fwdOrRevR;

  // 4. Offset using the progressive offsets
  //      - Based on the original DESIRED_THRUST value
  //      - Add offsets from left side motor
//  char cOffset = 'x';
//  if (m_thrustCommanded < 10)        { mappedLeft += m_Offset_LT10;         cOffset = '0'; }
//  else if (m_thrustCommanded < 20.0)  { mappedLeft += m_Offset_GTE10_LT20;    cOffset = '1'; }
//  else if (m_thrustCommanded < 30.0)  { mappedLeft += m_Offset_GTE20_LT30;    cOffset = '2'; }
//  else if (m_thrustCommanded < 40.0)  { mappedLeft += m_Offset_GTE30_LT40;    cOffset = '3'; }
//  else if (m_thrustCommanded < 50.0)  { mappedLeft += m_Offset_GTE40_LT50;    cOffset = '4'; }
//  else if (m_thrustCommanded < 60.0)  { mappedLeft += m_Offset_GTE50_LT60;    cOffset = '5'; }
//  else if (m_thrustCommanded < 70.0)  { mappedLeft += m_Offset_GTE60_LT70;    cOffset = '6'; }
//  else if (m_thrustCommanded < 80.0)  { mappedLeft += m_Offset_GTE70_LT80;    cOffset = '7'; }
//  else if (m_thrustCommanded < 90.0)  { mappedLeft += m_Offset_GTE80_LT90;    cOffset = '8'; }
//  else                              { mappedLeft += m_Offset_GTE90;         cOffset = '9'; }

  // 5. Deal with overages
  //      - Any value over m_MaxThrustValue gets subtracted from both sides equally
  //      - Constrain to [-m_MaxThrustValue, m_MaxThrustValue]
  double maxThrustNeg = -1.0 * m_dMaxThrust;
  if (mappedLeft  > m_dMaxThrust)
    mappedRight -= (mappedLeft  - m_dMaxThrust);
  if (mappedLeft  < maxThrustNeg)
    mappedRight -= (mappedLeft  + m_dMaxThrust);
  if (mappedRight > m_dMaxThrust)
    mappedLeft  -= (mappedRight - m_dMaxThrust);
  if (mappedRight < maxThrustNeg)
    mappedLeft  -= (mappedRight + m_dMaxThrust);

  m_des_L  = clamp (mappedLeft,  (-1.0 * m_dMaxThrust), m_dMaxThrust);
  m_des_R  = clamp (mappedRight, (-1.0 * m_dMaxThrust), m_dMaxThrust);
  return true;
}
```

References

[1] CDC, "General information about harmful algal blooms," Centers for Disease Control and Prevention, Jul. 05, 2019. https://www.cdc.gov/habs/general.html (accessed Apr. 19, 2020).

[2]J. A. Coller, M. J. Sypnewski, S. B. Taylordean, C. J. Goodrum, and D. J. Singer, "The Design of an Autonomous Surface Vehicle for the 2018 Maritime RobotX Challenge," p. 9.

[3]“SR Utility Class,” SeaRobotics Corporation. https://www.searobotics.com/products/autonomous-surface-vehicles/sr-utility-class (accessed Apr. 27, 2020).

[4]“Inflatable Boat Lifespan.” https://newportvessels.com/inflatable-boat-lifespan/ (accessed Apr. 27, 2020).

[5]“XCAT Basic,” XCAT, Dec. 15, 2016. https://www.x-cat.com/en/xcat-basic (accessed Apr. 27, 2020).

[6]“Accessories,” Native Watercraft. https://nativewatercraft.com/product/biyak/ (accessed Apr. 27, 2020).

[7]“Blue Sky Boatworks 360 Escape 2019,” Caney Fork Outdoors. https://cfoutdoors.com/blue-sky-boatworks-360-escape-2019/ (accessed Apr. 27, 2020).

[8]“12 foot outrigger kit for 2 sides – Expandacraft.” https://expandacraft.com/product/12-foot-outrigger-kit-for-2-sides/ (accessed Apr. 27, 2020).

[9]“Kayak Series - Kayak Trolling Motor,” Newport Vessels. https://newportvessels.com/kayak-series-kayak-trolling-motor/ (accessed Apr. 27, 2020).

[10]“Ulterra 80 lb. / FP / MDI / i-Pilot Link - 60" | Minn Kota Motors.” https://minnkotamotors.johnsonoutdoors.com/featured-products/ulterra (accessed Apr. 27, 2020).

[11]“Vantage 80 lb. / Hand - 31" | Minn Kota Motors.” https://minnkotamotors.johnsonoutdoors.com/freshwater-trolling-motors/vantage (accessed Apr. 27, 2020).

[12]M. R. Benjamin, H. Schmidt, P. Newman, and J. J. Leonard, “An Overview of MOOS-IvP and a Users Guide to the IvP Helm - Release 13.5,” p. 301.

[13]“LiDAR sensor functionality and variants, 8022040,” p. 16.

[14]“Navigation Rules,” United States Coast Guard. p. 226.

[15]“Marine autonomous vehicles and the law: Assessing risks and liability - Professional Mariner - June/July 2018.” http://www.professionalmariner.com/June-July-2018/Marine-autonomous-vehicles-and-the-law-Assessing-risks-and-liability/ (accessed Jan. 27, 2020).

[16]“On Lake Michigan, Almost Time For Autonomous Boats? | WUWM.” https://www.wuwm.com/post/lake-michigan-almost-time-autonomous-boats#stream/0 (accessed Jan. 27, 2020).

[17]“Autonomous vehicles: The legal landscape in the US | Publications | Knowledge | Global law firm | Norton Rose Fulbright,” https://www.nortonrosefulbright.com:443/en/knowledge/publications/imported/2018/07/18/05. https://www.nortonrosefulbright.com/en/knowledge/publications/2951f5ce/autonomous-vehicles-the-legal-landscape-in-the-us (accessed Apr. 27, 2020).