Metachronal Paddle Powered Remote Underwater Vehicle

Final Design Report

Members:

Andrew Spencer

Gage Greenhouse

Gary Lopez

Joshua Brown

Matthew Scott

William Schlotthauer

Date Submitted: May 1, 2020

Table of Contents

Project Overview

Metachronal Motion

The goal of this project was to design a remote operated paddle powered underwater vehicle that utilizes metachronal motion. This paddling motion is based on the metachronal paddling motions of shrimp and krill, and is defined as a wave pattern created by the sequential actions of structures. The primary advantage of metachronal motion in comparison to traditional hydrodynamics, is found at smaller scales, where metachronal motion produces higher drag in water. This advantage is why many pleopods use this motion to optimize swimming by giving the animal a weight support from the motion of the fluid. Metachronal motion produces more uniform flow vectors that provide not only a forward thrust, but are also capable of producing a lifting force. This type of motion can create numerous different flow vectors, it is even utilized by some species of krill to swim upside down. Figure 1 shows a comparison between a simple synchronized paddling on the left and metachronal paddling on the right.

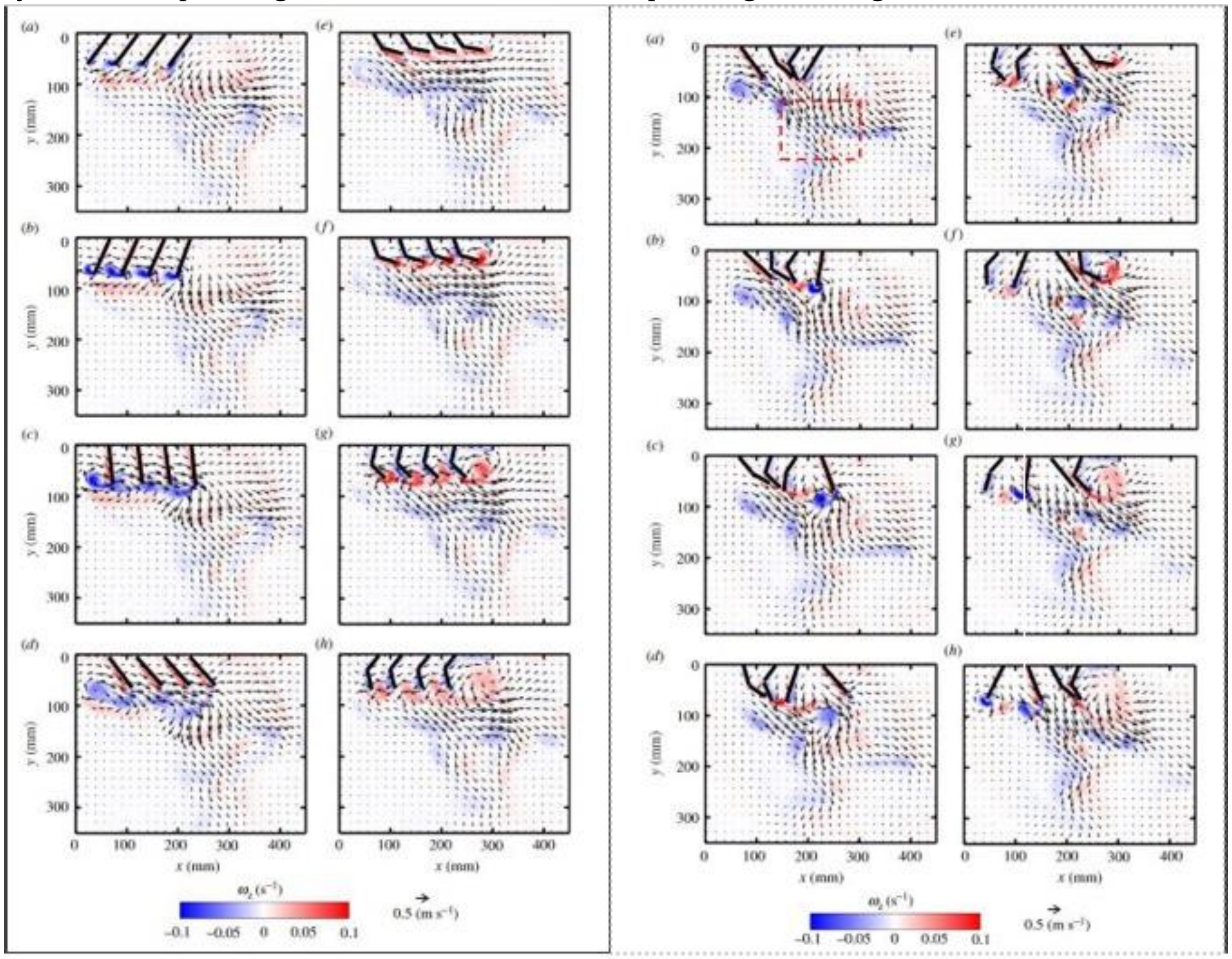


Figure 1

New Project Scope and Objectives

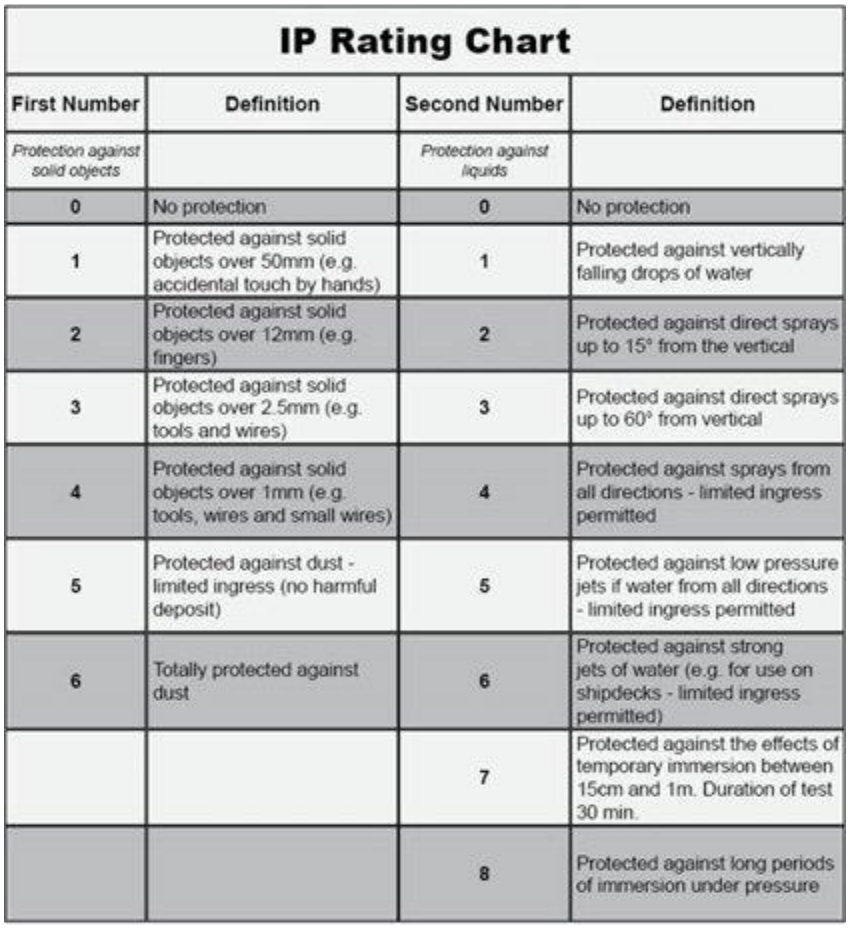
The goal of this project was to build an underwater vehicle that utilizes the concept proven by Dr. Santhanakrishnan and Mitchell Ford, but on a smaller scale, so that the vehicle can operate freely in water. To accomplish this, we were given several constraints and goals. The robot needed to be lightweight less than five pounds would be ideal, and less than ten pounds was required. The robot needed to be less than a foot in length to facilitate testing within the lab. The robot needed to reach a top speed of at least 0.5 m/s and accelerate to that speed within one body length of the robot. Dr. S. also required a camera be attached so that video can be recorded from the vehicle.

Applicable Codes and Standards

Since our project is an ROV, there aren't many codes and standards to be found, but we can abide by different general building codes and safety codes.

IEEE 1680 is a standard for environmental assessment of electronic products. This has to do with reduction or elimination of environmentally sensitive materials. It also includes codes for design for end of life, life cycle extension, energy conservation, end-of-life management, corporate performance, and packaging. To abide by these standards, we have design the ROV to be environmentally friendly. It is electrically powered so it does not give off any emissions. This ROV does however use acrylic cement. To abide by these standards, one must properly dispose of the cement after use and follow the proper cure time before use. This ROV is designed to not have an end of life, but to constantly be upgraded and modified for testing. We do however provide proper disassembly and disposal techniques. Energy conservation can be achieved by using renewable energy to power this ROV. IEEE standards also talk about futureproofing and corrosion resistance. Our ROV can constantly be upgrades to tackle the futureproofing standard. Our ROV is made mostly acrylic, this is corrosion resistant. There is however a steel spring inside the shaft seal that can corrode and will have to be replaced in regular maintenance intervals.

The American Society of Mechanical Engineers has one standard we can abide by. That standard is B18.2.1. It is Square and Hex bolt screws. This code describes that square and hex bolt screws should have a standard tolerance and sizing. This is more for manufactures than for us. We will be using square and hex bolts in the fastening of acrylic plates on the ROV.
The National Fire Protection Association has two standards and codes that we can use in our project. NFPA 70B and NFPA 70E. 70B is Electrical equipment maintenance. NFPA 70B describes that power should be disconnected when doing maintenance. This is to minimize the chances of shock when working. We will disconnect power and the Raspberry Pi when doing maintenance to our ROV. NFPA 70E is electrical safety in the workplace. This standard states that an arc flash and shock hazard sign must be posted if using an arc or powering an object. We have printed off a sign and will post it in Fablab 104 when the ROV is under power, or when/if we weld. This sign should be posted anywhere the ROV is at all times.
Ingress Protection rating describes waterproofing standards. Our ROV will incorporate IP67 Servos. The standard chart is shown below in figure 2.



IP Rating Chart

First Number	Definition	Second Number	Definition
Protection against solid objects		*Protection against liquids*	
0	No protection	0	No protection
1	Protected against solid objects over 50mm (e.g. accidental touch by hands)	1	Protected against vertically falling drops of water
2	Protected against solid objects over 12mm (e.g. fingers)	2	Protected against direct sprays up to 15° from the vertical
3	Protected against solid objects over 2.5mm (e.g. tools and wires)	3	Protected against direct sprays up to 60° from vertical
4	Protected against solid objects over 1mm (e.g. tools, wires and small wires)	4	Protected against sprays from all directions - limited ingress permitted
5	Protected against dust - limited ingress (no harmful deposit)	5	Protected against low pressure jets if water from all directions - limited ingress permitted
6	Totally protected against dust	6	Protected against strong jets of water (e.g. for use on shipdecks - limited ingress permitted)
		7	Protected against the effects of temporary immersion between 15cm and 1m. Duration of test 30 min.
		8	Protected against long periods of immersion under pressure

Figure 2

This rating is used to evaluate devices by how water and dust proof they are. In our case, an IP rating of 66 or better is preferred for our choice of servos due to them likely surviving a leak in our robot's hull.

DNV*GL is a Norwegian and German Society that has standards for ROV's. DNV*GL stands for Det Norske Veritas and Germanischer Lloyd.

1.1) ROV should be designed to where if a single failure occurs, no dangerous situation can occur.

This ROV is designed so that if a single failure occurs, no dangerous situation will occur. Power will be cut.

1.2) ROVs and their components shall be designed to meet the service conditions stated in the specification.

This ROV was designed to meet the requirements given to us and the conditions it will be tested in.

1.3) ROVs shall be designed and built to ensure safe operation and facilitate proper maintenance and the necessary surveys.

If the user manual and maintenance manual are followed, this standard will be accomplished.

1.4) ROVs shall be designed and constructed in such a way that sufficient possibilities for monitoring during dived travels are given. This can be achieved, e.g., by video systems and acoustic instruments.

Camera is attached to this ROV where a live video is possible. Depth senor can be added with ease. This is due to many extra ports on the Raspberry Pi.

1.5) ROVs shall be so equipped that the operator can be informed about the position and the operating condition of the vehicle.

Camera is attached to this ROV where a live video is possible.

1.7) Due care shall be taken to ensure that inadvertent movements cannot cause the remotely operated vehicle to destroy itself or equipment located at the work site or to become separated from its control and supply lines (e.g., by cable protector).

ROV will be in a tank. Area around tether shall be open and free of any human or object. Tether will be help by a person guiding it into the tank with some slack. ROV could hit sides of tank.

1.8) If electronics should fail, ROV should retain buoyancy.

If electronics fail, buoyancy is not changed. We have a static (3d printed) weight in the ROV to retain buoyancy that is not electronically controlled.

1.12) ROVs shall be so designed, that their operation causes no inadmissible environmental loads and endangering of the environment will be avoided as far as possible.

No emissions come out of this ROV. If user manual and maintenance manual are followed, proper cleaning after use to be specific, no environmental issue should be present.

6.2.5) The test pressure applied to vessels and apparatus with stress from internal pressure shall be equivalent to 1.5 times the maximum allowable working pressure.

This can be tested with the box test in a tank.

6.8.2) All electrical systems and equipment shall be inspected and tested before the ROV is put into service.

If The manuals given are read and used properly, this is accounted for. We have an assembly manual that states one must check all electrical and mechanical connections and test them before service.

IPC (Institute for Printed Circuits)

IPC-2221 Generic PCB

This is a standard for printed circuits that can be done at the ENDEVOR Lab. This standard can be followed if circuit is printed.

Body Design

Design Updates

The design we presented at the FDR featured an aluminum body, with internal layers, a bottom layer for the servos, and a top layer for the rest of the electronics, and paddles sticking out the sides, directly driven by the servos. Figure 3 shows this design.

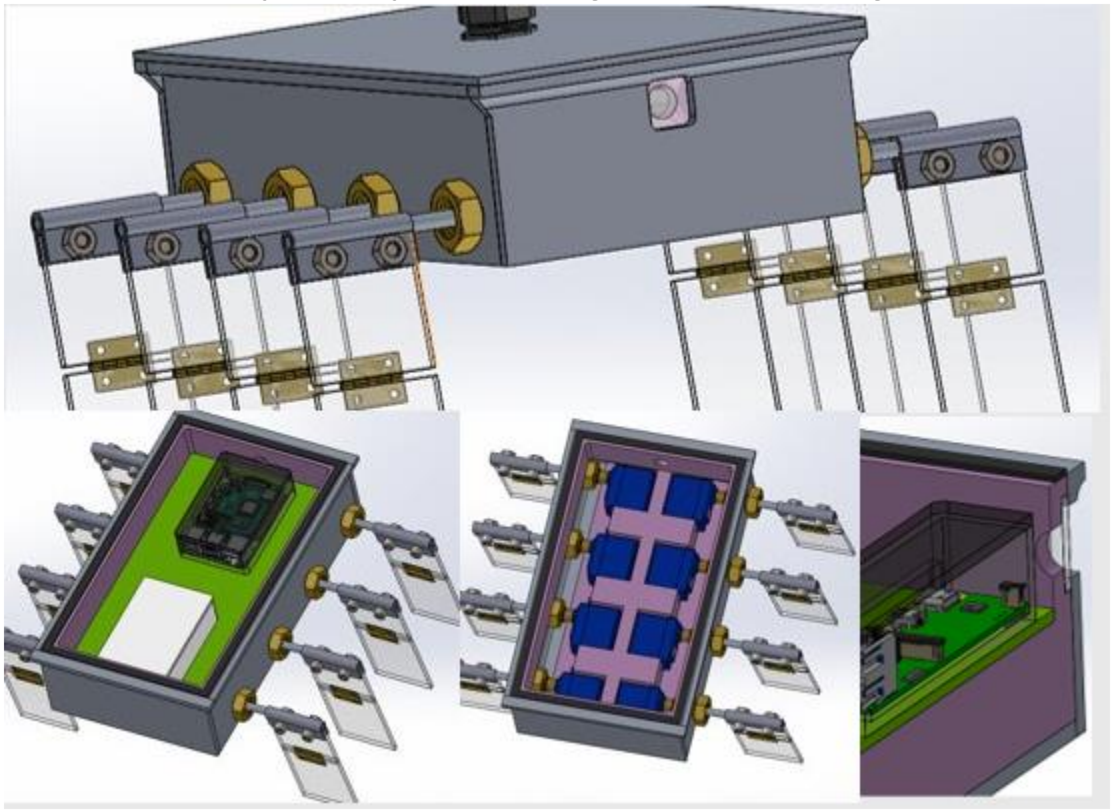


Figure 3

We received valuable feedback concerning various portions of this design including the limitations of welding aluminum together. We also were asked to provide fenders for the paddles to aid with metachronal motion, which coupled with an updated material altered our designs to a fairly large degree. The first step in updating our design was to finalize the material for our body. We narrowed our list down to the original aluminum, polycarbonate, and acrylic. Figure 4 shows the decision matrix for these materials.

		Aluminum		Polycarbonate		Acrylic	
Criteria	Weight	Score	W/ Weight	Score	W/ Weight	Score	W/ Weight
Cost	2	3	6	2	4	2	4
Machinability	3	2	6	1	3	3	9
Manufacturability	4	1	4	3	12	3	12
Durability	2	3	6	2	4	1	2
weight	3	1	3	3	9	3	9
Total			25		32		36

Figure 4

While aluminum was slightly cheaper, neither polycarbonate nor acrylic would break the bank. Aluminum was the most durable, and the panels for the design were easy to produce, but it was the heaviest material, and would prove difficult to weld together. Polycarbonate and acrylic have rather similar properties, and while polycarbonate is more durable and less brittle than acrylic, it is difficult to precisely machine and cannot be laser cut, whereas acrylic can be. The ability to quickly, easily, and precisely machine all of the panels with the use of the laser cutter available in Endeavor, is what ultimately made acrylic the most favorable option for the body material.

After deciding on acrylic we had to redesign the body using that new material. Our team developed three design concepts that were decided upon using the decision matrix in figure 5.

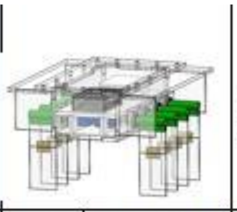
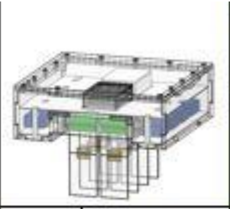
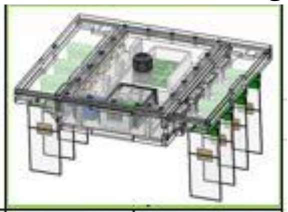
							
Criteria	Weight	Score	W/ Weight	Score	W/ Weight	Score	W/ Weight
Stream lined	2	2	4	1	2	2	4
Water-Proofability	3	3	9	2	6	2	6
Manufacturability	2	2	4	1	2	2	4
Bouyancy/weight	3	2	6	1	3	2	6
Champion Approval	5	2	10	1	5	3	15
Total			33		18		35

Figure 5

The first design is similar to the aluminum design from the FDR, with extended flanges on the front and top plates to act as a sort of fender to aid the metachronal motion. The second involved moving the paddles to the center of the robot. This design also changed from a flanged design for the lid to threaded inserts placed on the interior of the robot. This design featured a large amount of empty space on the top second layer, which was accounted for in the third design by widening the gap between the left- and right-hand side servos and removing the need for a second layer.

The first design shape is the least wide design, but due to the flanged top would not experience uniform flow over the top of the robot. Ultimately the second design was too complicated, and left Dr. S. with concerns on whether or not it would be capable of producing the correct flows for metachronal motion. Ultimately design 3 won out because it was the one Dr. S. liked the best.

New Design Details

This new acrylic features many features that have gone unchanged since the initial design phases, such as direct drive from the servos, a 3D printed servo mount, and a small body footprint. All of these are done to maximize output while keeping the mass of the robot small enough to overcome the inertia as it begins paddling. The length is 10.5 inches, which is about the minimum distance we can have our paddles spaced apart, while still being able to achieve metachronal motion without the paddles impacting one another. The width is 12.75 inches and the height of the body is 2.31 inches, these are both the minimum dimensions allowed that still allow for all of the necessary components to be placed in the robot. Figure 6 shows the design in more detail, while figure 7 shows an exploded view of the design.

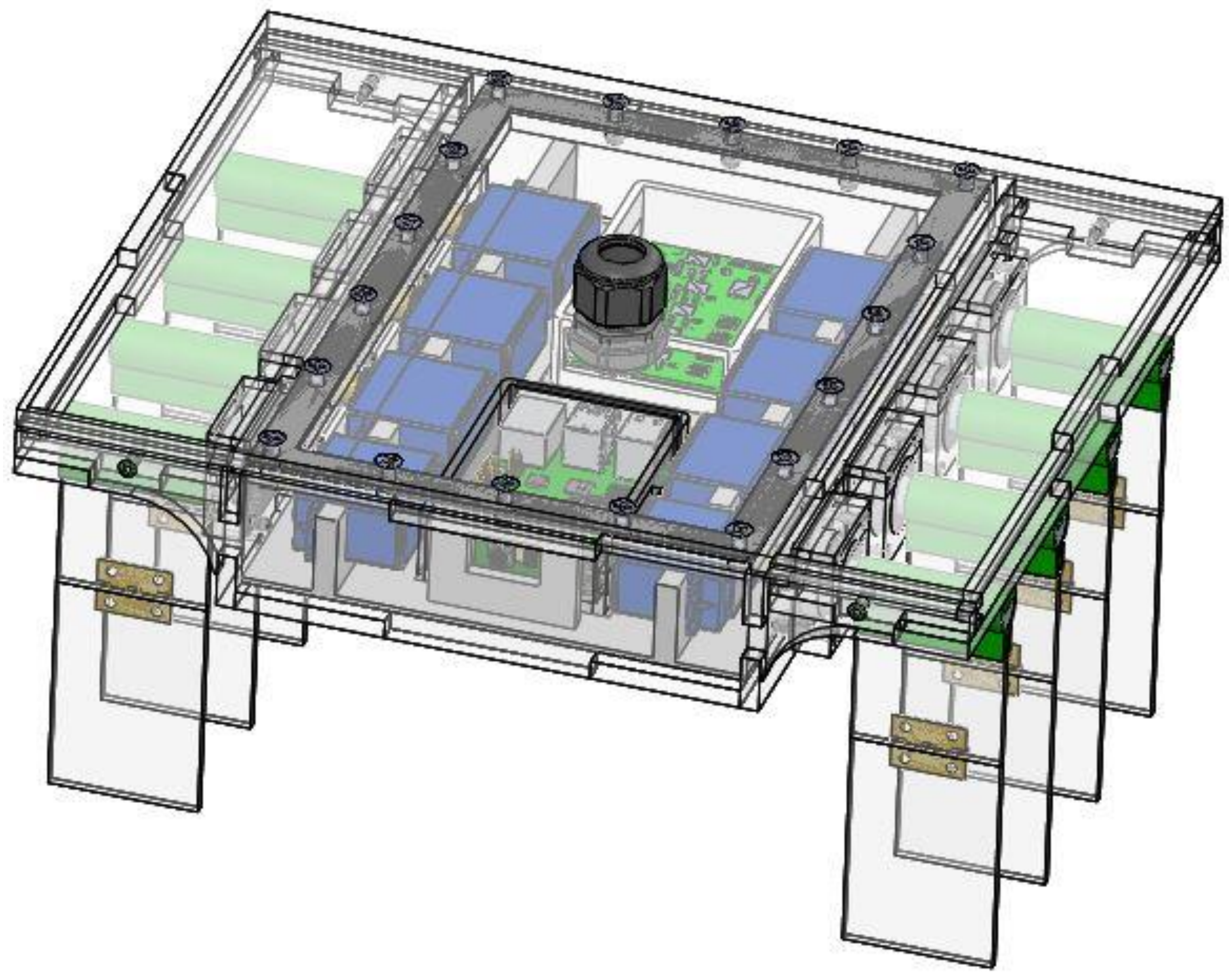


Figure 6

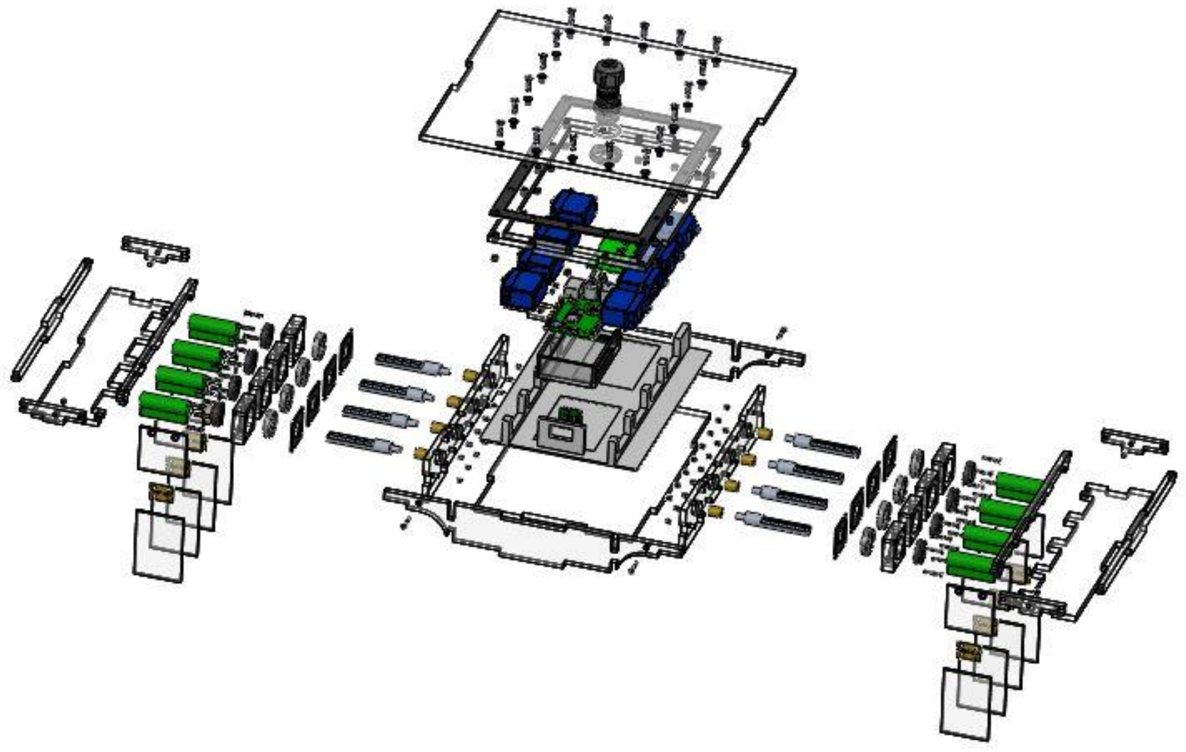


Figure 7

As can be seen in the figures above the design features laser-cut acrylic panels that are to be adhered together using IPS Weld-On 3 Acrylic Plastic Cement. The lid and shaft seals utilize a rubber gasket to ensure watertight seal and are secured with sealing screws, which are screwed into threaded inserts placed in the panels of the robot, while a PG19 cable gland is used to provide a waterproof seal around the tether entrance. A 3D printed insert serves as a mount for the servo motors and the camera. The servos are then able to directly drive the paddles. Situated above the paddles are two fenders that are constructed out of acrylic panels and are to be screwed into the end plates and help maintain the flow needed for metachronal motion. The shaft assembly is easily interchangeable and assembles by 4 screws compressing a coupler holding the seal and bearing on a gasket. The design user interface and control is handled by a Raspberry Pi through a custom written Python program leveraging Flask for a web interface. This allows the paddles to be fully configurable per side and can be set to be fully synchronized or individually independent. Power will be supplied through a tether cable comprised of three DC conductors and an Ethernet cable. Power is routed through the robot by the power supply board that steps the voltage down for the internal electronics. Figures 8 through 10 show the top, front, and right views of the design.

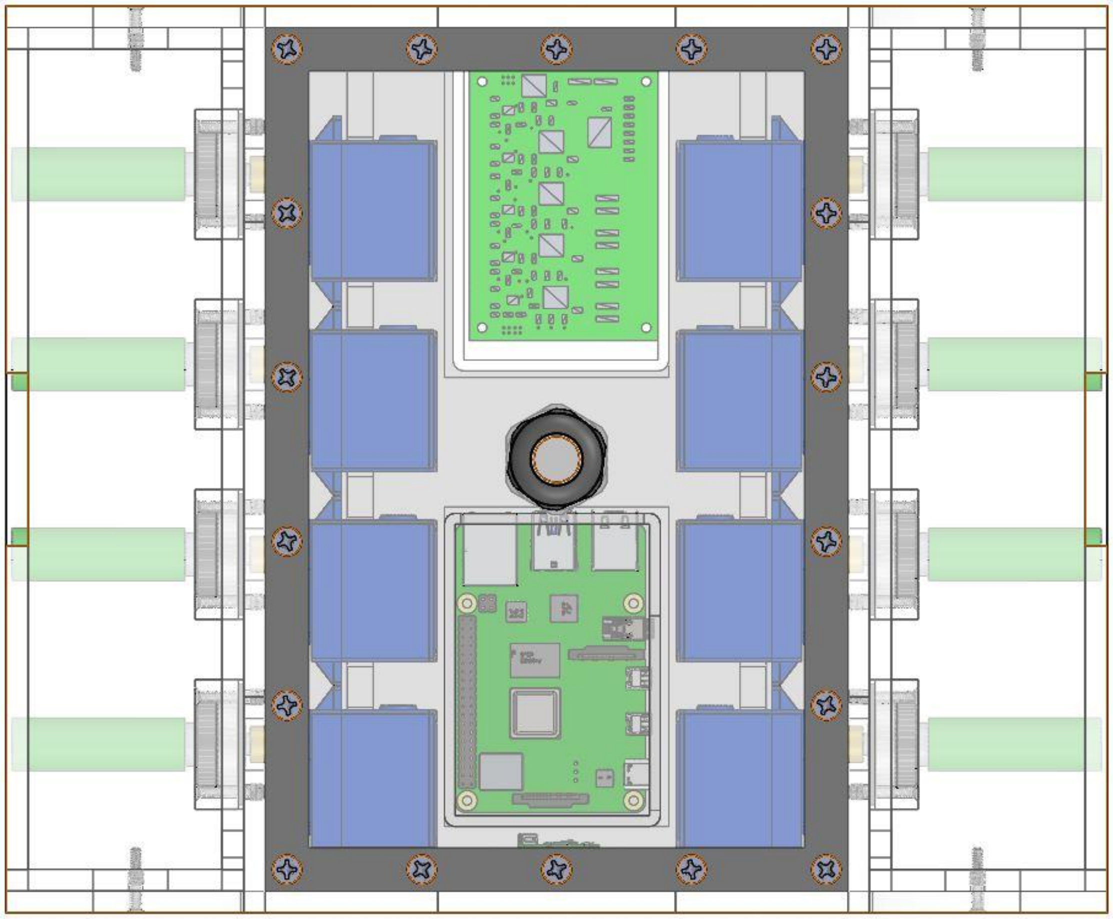


Figure 8

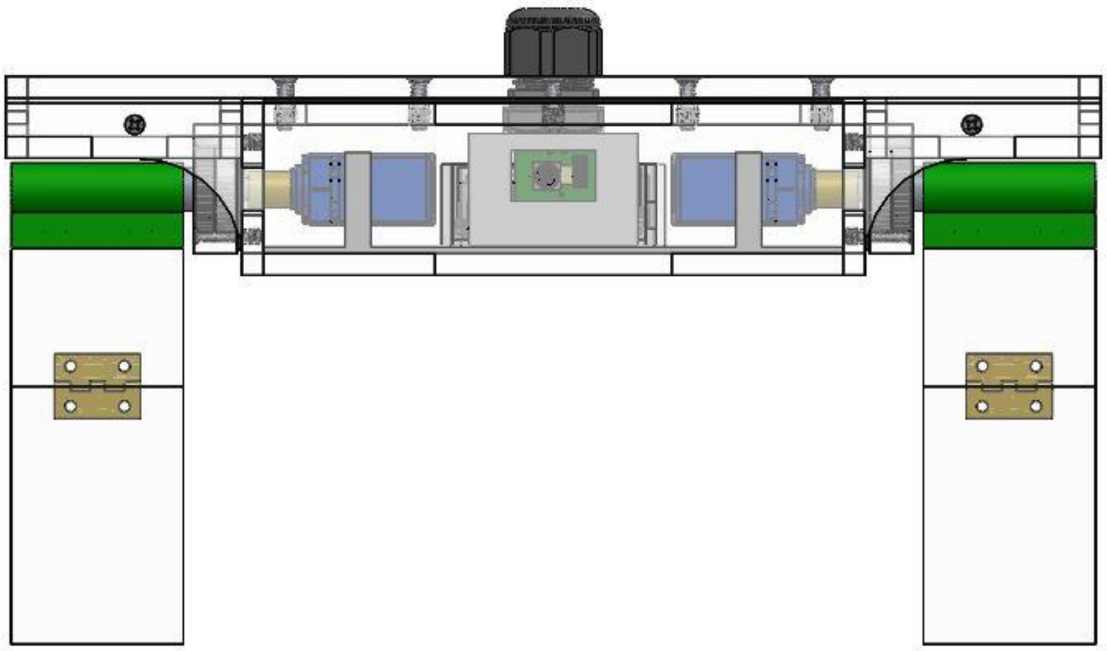


Figure 9

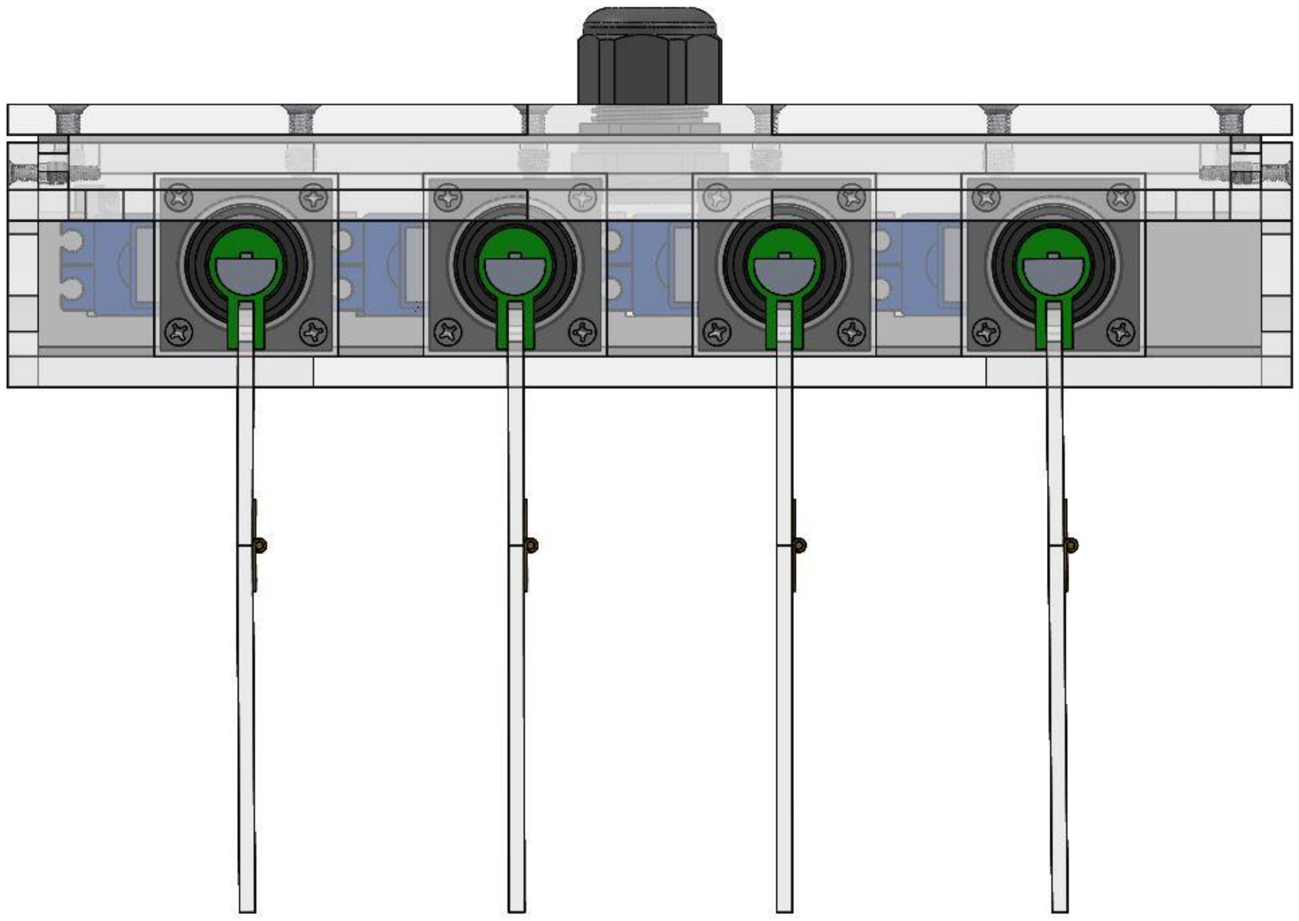


Figure 10

This design meets DNV*GL (Det Norske Veritas and Germanischer Lloyd) 1.8, where if the robot has electrical issues, it is still positively buoyant. We were able to determine this by testing to see if our design is lighter than the mass of the water it displaces. From given product weights (each servo is 60g) and from our Solidworks Model we know that our design weighs roughly 7.4lbs without additional ballasting, and we know that our design displaces (2.25" * 7.25" * 10.5") [Main Body] + 2x (25.27 in^3)[Fenders] + (26.2in^3) [Paddles] = 248 in^3 = .144 ft^3. Multiply that total by the density of water (62.4 lbs/ft^3) and we find that the mass of the water displaced is roughly 8.9 lbs. Since 7.4 is less than 8.9 our sub is positively buoyant. The design will be able to demonstrate metachronal motion while floating. Manipulating the 3D printed servo mount or by filling in the empty spaces within the fenders, neutral buoyancy can be easily achieved.

Limitations of Design and Future Work

Our design is not without limitation. One aspect of the design that can be improved upon is reshaping the design to be more aerodynamic. As a submarine, the design will require ballasting. An immediate recommendation is to increase the 3d printed pieces and fill in air gaps to increase the mass of the robot to achieve neutral buoyancy, while further along a team could add an active ballasting system to allow for the user to control the dive. Another limitation is that acrylic is brittle, which demands an extra degree of caution, and results in an increased risk in cracks to the body. Our design is not suited for high pressures, so the design will need to be reworked for deep-water use.

It is also recommended that the user interface be updated to be compatible with a controller. Other future work includes adding a feedback system which cuts servo power, makes robot positively buoyant, then shuts system power if leaking is detected internally. Another long term goal is autonomy. Future teams could also look into upgrading from a passive hinge system to an active hinge system, and research the effects of width between right-hand- and left-hand side paddles on metachronal propulsion.

Shaft Assembly Design

Design Updates

The shaft assembly for the robot was previously very expensive, and difficult to manufacture. The design needed a much simpler way of applying a waterproof application to a rotating shaft for the project to be completed with plenty of testing time before the end of the semester. The previous design uses cut brass tubing which has sleeve bearings and O-rings with wiper lids where the shaft will be pressed through the sleeve bearings with lubrication. The actual tubing that holds the apparatus is lathed to where thin bronze nuts and O-rinds were used to compress the system down and hold it in place. There are many issues with the design of this scheme which leads up to the new design.

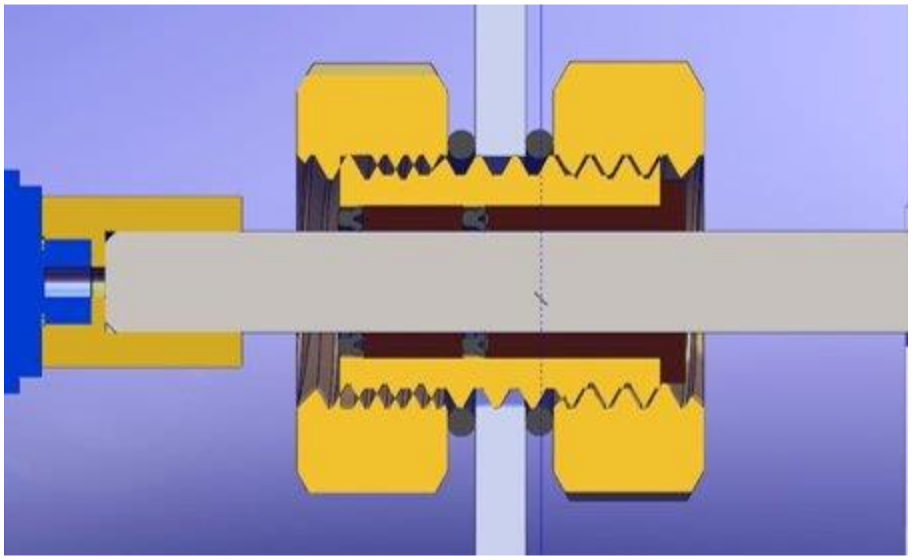


Figure 11

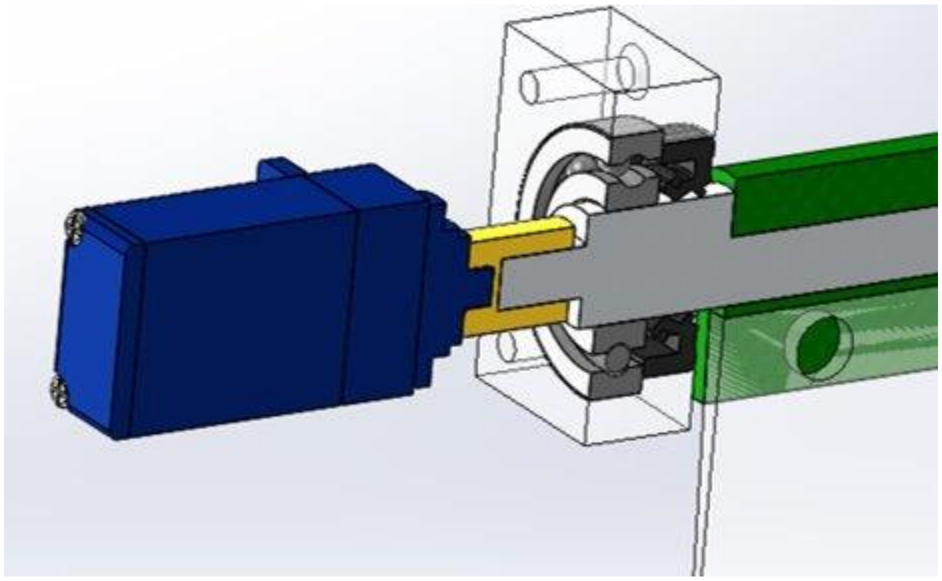
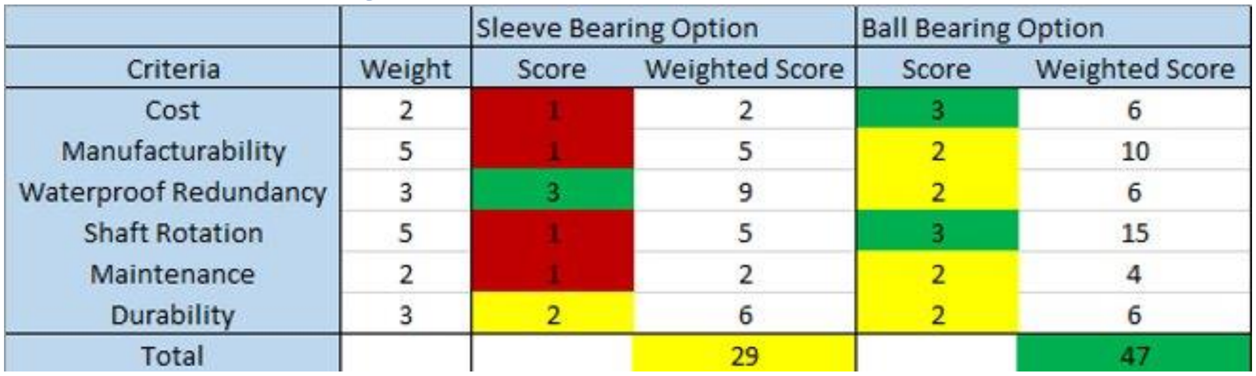


Figure 12

Figure 11 (top) and 12 (bottom). Figure 11 shows a cut view of the old design scheme using sleeve bearings spaced by O-rings with wiper lids to seal the shaft. Figure 12 shows the new application which uses a ball bearing and a spring loaded rotary shaft seal with wiper lids to waterproof the rotating shaft.

The new design is a much simpler design to manufacture and is a lot more cost efficient than the previous design. In the new design, a ½" acrylic sheet is laser cut into a 1 ½” by 1 ½" sheet that will hold both the plastic and glass balled ball bearing and the rotary shaft seal with steel spring on wiper lid. This apparatus is bolted onto the body with a gasket that seals the space between the acrylic body and the acrylic shaft holder in the subassembly. The aluminum shaft has minor manufactured processes done for it to fit onto the servo motor’s bronze shaft coupler (holds ¼" outer diameter shafts). The shaft is purchased as a ½" OD shaft. On one end of the shaft, the shaft is lathed down to ¼" to fit into the shaft coupler but remains ½" through the ball bearing and wiper lid shaft seal. On the other end of the shaft, A D-profile with a male end key in the middle is cut out of the shaft to fit a 3D printed sleeve that holds the actual paddles. It is recommended that the shaft is marked where these cuts are to be made after the servo motors and waterproof subassembly is in place. The paddles are laser cut where a pair of nuts and bolts are used to clamp the paddles down to the shaft sleeve, keeping the paddle in line during motion. The servo’s chosen for this assembly is the HS-656WP from servo city. The shaft coupler is purchased for the spline on the servo. This servo was chosen for multiple reasons and is discussed in the “Decision Matrix and New Design Details” section of this paper. The paddles are made from 1/8” acrylic sheets cut into two sections, the top section (2”x 2”) and the bottom section (3” x 2”). These sections are attached together by a small brass hinge that can be purchased at Lowes.

Decision Matrix and Design Justifications



		Sleeve Bearing Option		Ball Bearing Option	
Criteria	Weight	Score	Weighted Score	Score	Weighted Score
Cost	2	1	2	3	6
Manufacturability	5	1	5	2	10
Waterproof Redundancy	3	3	9	2	6
Shaft Rotation	5	1	5	3	15
Maintenance	2	1	2	2	4
Durability	3	2	6	2	6
Total			29		47

Figure 13

	Cost	Torque	Water Resistance	Current Draw	Speed
HS-656WP*	++	++	+++	+++	++
Amazon Special	+++	++	++	++	+
D646WP	+	++	+++	+++	++
Dr. Bai's	+++	+	+	+	+

Figure 14

Figure 13 (Top) and 14 (Bottom). Figure 13 shows a decision matrix used to determine which shaft sealing application to use after both designs were shown to the team. A scale between one and three (one being the worst, three being the best) was assigned to each major category. The weighting applies the importance of each section which will factor into a total score, where the higher score is the overall better option. Figure 14 shows a decision matrix for the servo options, where + is the worst score and +++ is the best score. The most + determines the chosen servo.

The shaft sealing techniques and the servo motors were given a list of criteria for which the components would be graded on. Unfortunately, due to unforeseen circumstances, the testing for some of the criteria were given "best educated guesses" for grading purposes. The chosen shaft sealing mechanism was the new idea where a ball bearing and rotary shaft seal is used to keep the shaft watertight, and the chosen servo was the HS-656WP due to its reasonable price, good torque, high water resistance, low current draw, and decent speed.

For paddle design analysis, the size of the paddles were based off multiple parameters. Ideally, an optimization code is to be configured to determine paddle dimensions based off velocity and position, but the design of the paddles were based off of a range of different paddle lengths that were determined from a code developed early in the design process. The code uses basic drag force assumptions applied to the legs with different lengths to determine how much stall torque will be applied at different angular velocities and stroke amplitudes. The range was set between 3 inches and 7 inches total for the size of the paddle. When designing the paddle, all of the servo motors and their stall torques were recorded and were also given a 10% offset from the stall torque to make sure the motor was still running (for example, if the stall torque is 2 Nm, 1.8 Nm is used to make sure the design is not meeting the stall torque since the current draw will spike and the motor will have low velocity then). For all the motors, the average stall torque was about 2.5 Nm, which 2.25Nm was used when determining the parameters. Testing of the motor for variable paddle sizes was one of the things that was going to be tested until unforeseen circumstances did not allow the team to do such. The final paddle size dimension was based off the model where the frequency of the paddles moving are compared to the torque of the motors specs. Testing of the frequency was also part of the testing of the paddles, but was not accomplished due to unforeseen circumstances as well. The guess frequency of the loaded servo motors since the testing was not accomplished was 1.5Hz, where the torque of the motor needed to be roughly 2.25 Nm. This concluded the 5” total length of the paddles to be the largest paddle from the simulation that allowed for the maximum velocity profile yet met the torque and frequency specs. The widths of the paddles were given as a set point to the body team to help them design the robot within the size constraints given (2” width allows them 8” of body width to stay within the given goal which adds up to about the raspberry pi width, servo widths and box thicknesses totaled together).

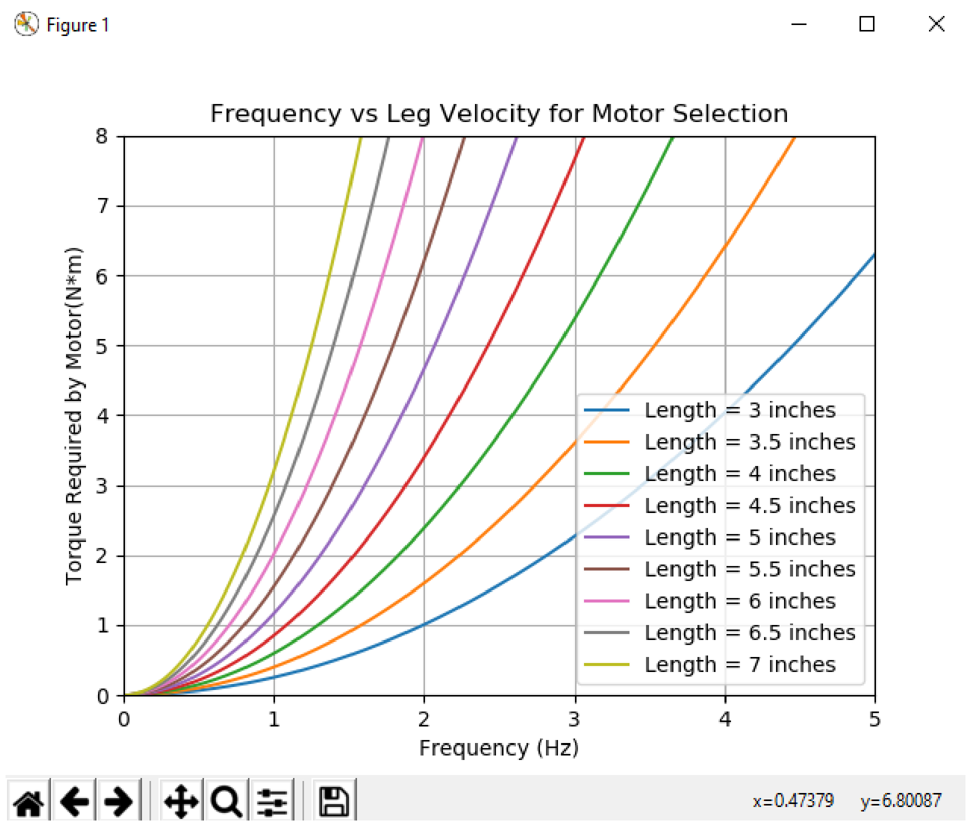


Figure 15

Figure 15. The diagram is the results of a python code that simulates the length of the paddles effecting to stall torque on the motors due to the force of drag. The code used to get the figure is shown in Appendix A.

Limitations of Design and Future Work

The sealing design seems like an easy, yet reasonable way to waterproof the shaft, but there are many limitations to the design. The first major limitation to the design is the code simulation. The code simulation, as shown in Appendix A, shows very basic dynamic equations and relationships to try and estimate certain parameters for the paddles rather than verifying them with experimental data. Ideally to accomplish this, an optimization code needs to be written that uses the code established from the “System Simulation” section of the paper to optimize paddle surface area while maximizing velocity and position. Other variables such as stroke amplitude could also be factored into the optimization as well. Also, as mentioned, experiments need to be ran prior to optimizing the code to validate the code.

For the shaft sealing design, the spring that holds the wiper lid in place is made from steel. If the robot is to corrode, the seals will need to be replaced. Also, the seals are rated for only 7 psi of pressure. This means after the robot is submerged below 16 feet of water, the seal will not properly be capable of water sealing the shaft. Ideally, there are other shaft seals for rotary applications that can withstand higher pressures, but do not support the shaft sizes used in this application. More research into different rotary shaft seals can be done to try and fit a larger shaft seal into place that can withstand higher pressures than the current application. Also research into different spring materials will help as a redundant factor for if the robot has shaft leaking issues.

The shaft sleeves are 3D printed at a fine setting for PLA. PLA can potentially wear, especially if constantly under a lot of stress constantly. More research into the shaft sleeve material could be very beneficial to improving the life expectancy on the sleeves, allowing the paddle sizes to be more interchangeable in the future.

Since the shafts were designed around 2" wide paddles, more research into shafts should be done to where if a new desired paddle width is to be tested, wider shafts do not have to be manufactured to properly hold the paddle. An idea is to alter the sleeve size and to place a "cap" over the hole that is exposed from the shaft sleeve when using for wider paddle applications. Also, a set screw connecting the sleeve to the paddle will help with the stability of the paddle and eliminating any potential slipping of the sleeve.

Power Supply, Distribution, and transmission Design

Design Updates

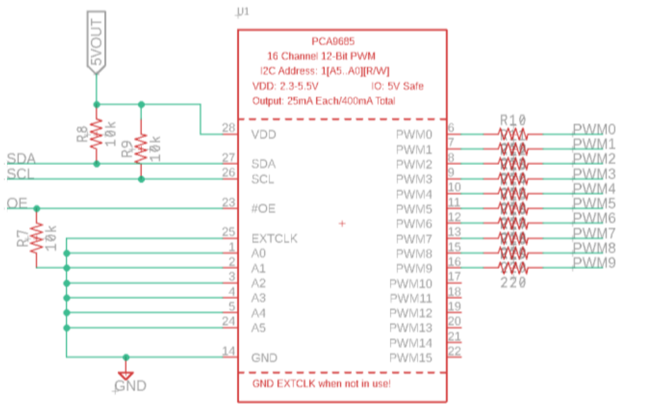
The power supply of the MPPRUV is handled by an external 24V power supply and an internal power distribution circuit. This was done in order to balance power supply with the anticipated power draw of the system. The other two options were Power over Ethernet (PoE) and a 5V or 48V DC power supply.

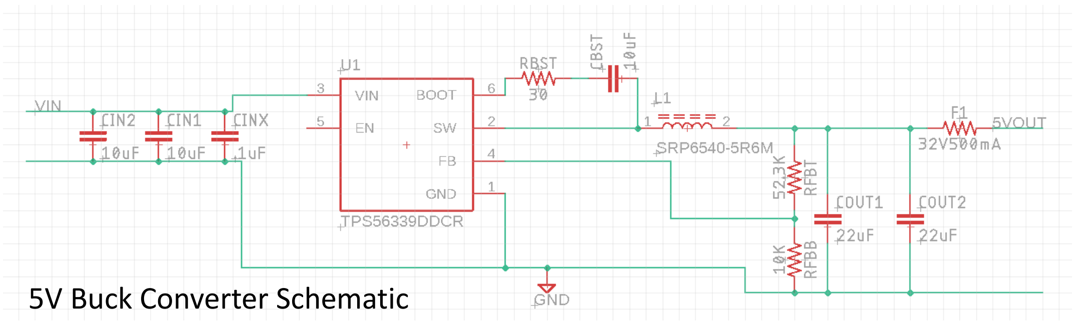
PoE was our first choice for power delivery as it would allow us to combine data communications and power into one, reasonably small and lightweight cable. We ran into issues with this design choice during research as we realized that the implementation would be much more complex than we believed we had time for. We also discovered that the absolute maximum power that we could supply with PoE was 90W, which was far below our worst-case power draw which came out to about 200W. It would require us to step the PoE voltage down to usable voltages for our DC equipment, have large magnetic components, and would take up a large amount of space in the final robot design. The advantages that the PoE would offer would be negated by the tether we had found in which we could have DC power and ethernet running in one cable.

In the end we chose a medium voltage, 24V DC power system that used buck converters to step the voltage down for our servos, Raspberry Pi, and servo controller. We also had two other options for our line voltage: 5V and 48V. 5V was too low to allow much power to flow to our robot over any distance, with the required power of 200W this would give us a minimum current of 40amp which would incur a lot of losses in the transmission line. The 40amps could be accommodated if the cable in the tether was very large and heavy, however this was not a feasible design. The other option was 48V, which in many cases would be better with lower losses in the transmission line. The reason we didn't go for 48V was that it was dangerously close to the minimum voltage required for power on safety checks and would likely have slowed down our development process. We did not research buck converters for this voltage, but from general research, using this voltage would increase heat output, component weight, and component size. After looking at our options we settled on 24V DC for the power through the tether as the voltage was high enough that we didn't need to use thick or heavy cables and the buck converters at this voltage were very efficient both with power and space.

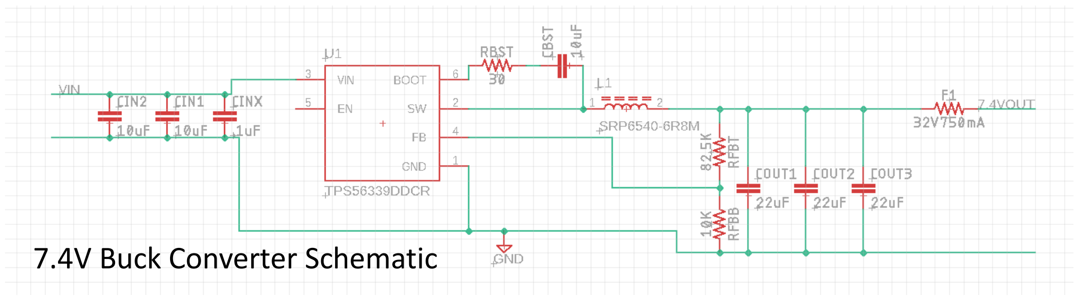
The power distribution and servo controller board are designed with our 24V tether power supply voltage, 5V Raspberry PI and servo controller voltages, and 7.4V servo voltages in mind. There are 5 buck converters as part of it, four of which output at 7.4V for the servos and one of which outputs at 5V for the Raspberry Pi and Servo Controller. Each buck converter is built implementing the Texas Instruments TPS56339 buck converter IC. This IC allows the conversion of voltages from 24V to any value from 1V to 12V, which is perfect for our 24V tether voltage and 5V and 7.4V power outputs. This also allows the output voltage to be changed to support many different optimal servo voltages for maximum performance. The servo controller is based on the PCA9685 from NXP and uses the design from Adafruit as reference. Using the Adafruit design as reference allows us to draw from the software libraries provided by Adafruit in our project, greatly simplifying the programming of the device by abstracting the software from the low-level hardware and programming required to handle I2C manually. The different schematics are shown below for the two different buck converters and the servo controller along with the overall schematic.

Servo controller
Adafruit PCA9685





5V Buck Converter Schematic



7.4V Buck Converter Schematic

Figure 16

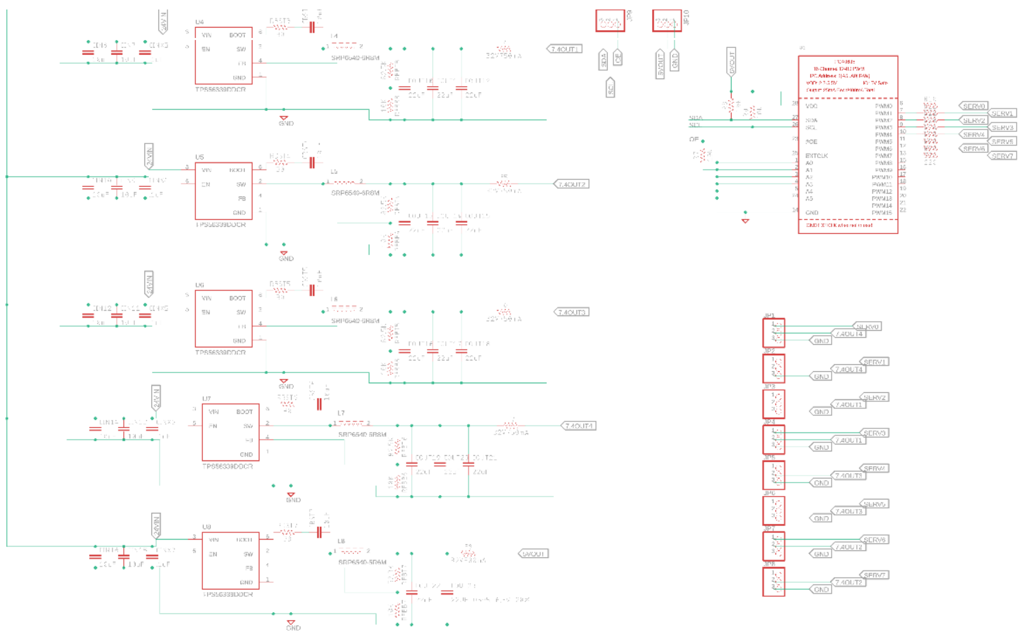


Figure 16

In our design we chose servos to drive our paddles over a few other options. Other notable options include stepper motors, linear actuators, and a very sophisticated planetary gear system. Overall servos were our best choice and will be explained in the following paragraphs.

Our first idea, and the one demonstrated to us, were stepper motors. Stepper motors were very quickly dismissed due to their large size and heavy weight, but the most positive aspects of stepper motors were their previous use in the lab model created my Mitchell Ford, their infinite variability, and capability for high speeds.

Our third idea, and the most complex, was the use of planetary gear systems along with a two-bar mechanism to generate the paddling motion. The best part of this is the electrical simplicity of the design, it would still use servos, but the servos could be small and not very powerful or fast. The actual power plant for this design would be a powerful brushless motor, one for each side with the ability to link both sides to ensure synchronization. The offset would be handled by moving the planetary gears to introduce a positive or negative offset in the position of the paddle. This idea was elegant but would have required the creation of very precise and complex parts and was not deemed to be feasible.

Our second idea, and another one we initially dismissed due to speed and power, was servos. Initially servos sounded like a great idea, we would have very fine control over their position at any given time, but their speed the torque would quickly become a problem with larger paddles or higher speeds. In the end, we decided to go with servos as no other option was as space efficient or as easy to handle hardware-wise. In the end the servos were the best choice for this project and will work great as we have managed to implement them.For the design of the actual power distribution board we first needed to decide on a size. We had already decided on implementing a raspberry pi into the machine and the MAE body team already had access to a raspberry pi model in solid works to design around. To make their life easy we told them to simply give us space for another raspberry pi to put the power distribution board and ensure we could run wires between them.

To keep with general good practice design principles, we had the power input on one side moving to the outputs, we decide to put the I2C and raspberry pi power location on the top of the board to put them as close to the Pi as possible. In this design with an input current of the board was going to be around 5 amps so to ensure the best connect we gave a long pad that you could strip a long portion of the wire and solder directly to the board. For grounding the different parts of the buck converters and servos we decided it was best practice to have a ground plane on the bottom of the board and use vias to connect the two sides. The became tricky when trying to connect the tether cable to the board, it makes the builders life a little easier we created a small ground pad on the top of the board with vias underneath it. We had a small concern about the vias holding the current to pass between the two sides, we overcame this by over sizing the vias and giving the builder the option to fill in the vias with extra solder.

$$trace\ area(mils^2) = \left(\frac{Current\ (3Amps)}{(0.048 * Temp\_rise(10℃)^{0.44})}\right)^{\frac{1}{0.725}}$$

$$trace\ width(mils) = \frac{trace\ area(mils^2)}{thicknessss(2oz) * 1.378\left(\frac{mils}{oz}\right)} = 26.9mils$$

Above is the calculations for the required trace width of the individual buck converters. A big assumption is that this calculation is with a 2oz copper board, this simply refers to the thickness of the copper. If a 1oz copper board used, then the required trace width comes out to 53.8mils instead of the 26.9mils that a 2oz copper board requires. This is important because the largest trace width I was able to fit onto the board was about 50mils, so this PCB requires a 2oz copper board to stay within safety tolerances. A note to make is the random constants that are present in the equation are from the IPC-2221 standard for PCB external traces.

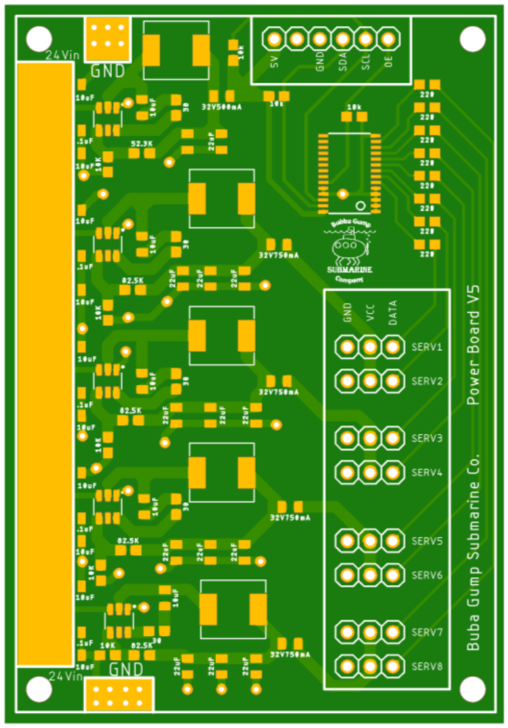


Figure 17 - Top

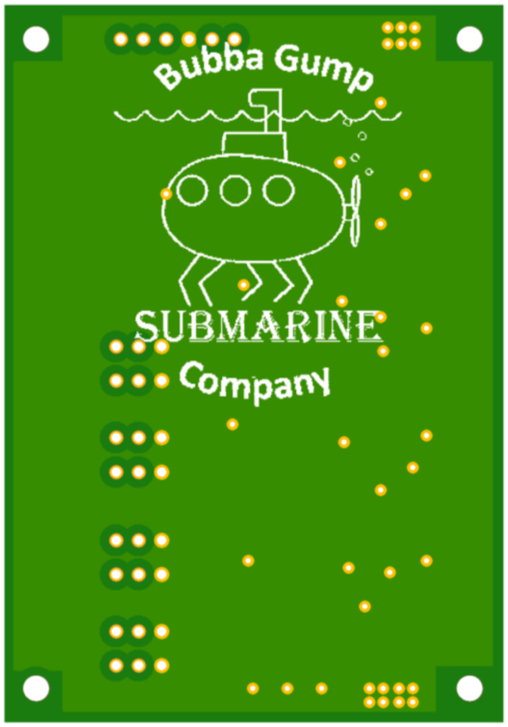


Figure 18 - Bottom

Decision Matrix and New Design Details

	Board Space	Complexity	Cost	Cable Complexity
Power Over Ethernet	🟥 +++	🟥 +++	🟥 +++	🟩 +
5V DC	🟩 +	🟩 +	🟩 +	🟥 +++
24V DC*	🟨 ++	🟨 ++	🟨 ++	🟨 ++

Above is the decision matrix for choosing the power delivery options and how we chose, that covered all of the needed areas.

We did not create a decision matrix for the power distribution board because we made it from scratch and thus were able to create it in a manner to best fit our needs. That being said in the future one safety change that I would consider making is putting direct connectors on the board, this would make for less open pads to be potentially shorted.

Software Design

Design Updates

In the beginning we had three options for writing software for this robot, the Robotics Operating System (ROS), hard programming in C/C++, and programming in Python. The first option that we pursued was ROS, it had excellent support for robotics and would have made making the robot autonomous a piece of cake. C and C++ was another option that we did not pursue due to the extreme complexity and time required to write all functions from scratch but would have been very performant. In the end we moved to Python as it had many functions built in and could still do low level communications with the servo controller.

ROS was the first option we tried, and we had some nice success with it. We managed to write out a publisher and subscriber such that we could control the robot in real time, but we ran into problems with high command rates as the program would break away from running the paddles to handle the latest command which slowed the program down dramatically. After this there was talk of moving the program to ROS Parameters, but this was not implemented as we switched away from ROS.

C and C++ was another option we considered. With these programming languages we could create a very high-performance application at the cost of a large and hard to handle program. Everything we wanted to do was very much possible with these programming languages and, if we were to create this program before 2010 would have been our first choice, but do to the simplicity and capabilities of Python, we didn't need to deal with C or C++.

In the end we settled on Python. We already had some code that could be carried over from the attempt at ROS as it ran on top of Python. We used python to great effect in this project. Combining both a web interface and the servo controller to create a comprehensive application that has both a good user interface and the low-level control necessary to run the servos directly within the application. We were again proven right in choosing this path through performance metrics as this application at most used 2.5% of the raspberry pi's computational power, even while running the servos at 100 times per second which is almost twice as high as necessary.

In order to keep the software burden on the Raspberry Pi, and the authors of said software, manageable, we decided to use the open source software Motion to handle the camera. Motion is a open source free software package originally designed to convert the Raspberry Pi into a motion activated security camera. From this package we disabled the motion detection and recording options and have it just serving a motion jpeg stream that can be opened in a web browser and viewed live with low latency. This software also can be expanded in the future with more cameras as it supports USB cameras as well, in most cases without much reconfiguration.

Decision Matrix and New Design Details

	Simplicity	**Ease of Use**	**Ease of Upgrades**	**Future Expandability**
Python with Flask*	+++	+++	++	+
ROS	+	+	++	+++

Note: Higher is better

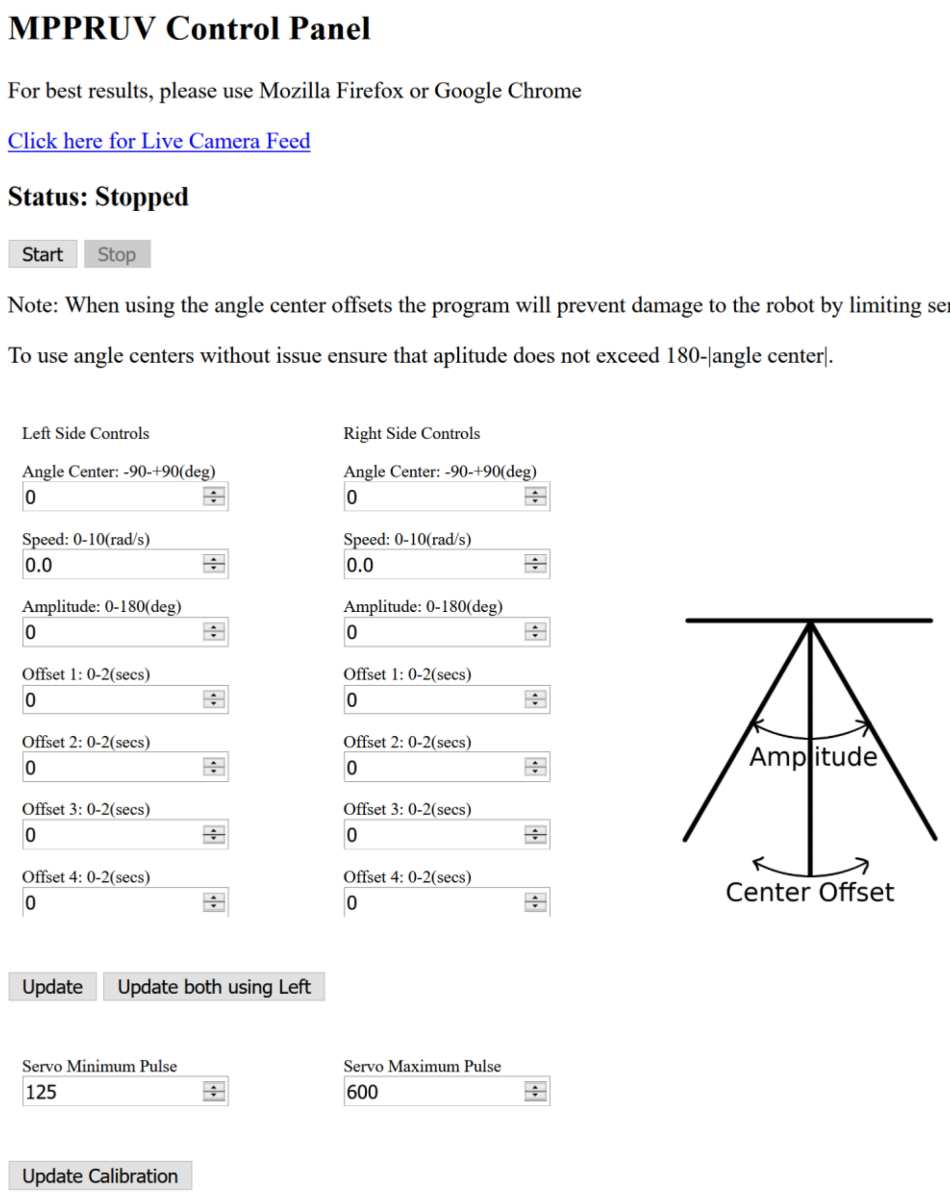
Limitations of Design and Future Work

The only issue that our final design will face is expandability in the future. With ROS it would have been very easy to make the RUV autonomous for add extra features that could be controlled by a controlling computer.

User Interface

Design Updates

The web interface was created using Python Flask, a Python package used to create web applications. We expanded the web application such that it could control and start processes on the Raspberry Pi itself. The web interface is created using an HTML webpage as a template with all the variables subbed in at request time. The variables are handled by an HTML form that allows the change of variables and a button to update the variables on the Raspberry Pi. The starting and stopping action are handled in the same form and as such allows the updating of variables with the start button (though this is not part of the specifications of the software). The web interface also allows the tuning of the pulse widths for the servos to ensure the proper calibration of the 180-degree maximum paddle amplitude.



MPPRUV Control Panel

For best results, please use Mozilla Firefox or Google Chrome

Click here for Live Camera Feed

Status: Stopped

Note: When using the angle center offsets the program will prevent damage to the robot by limiting servo travel.

To use angle centers without issue ensure that aplitude does not exceed 180-|angle center|.

Left Side Controls		Right Side Controls	
Angle Center: -90-+90(deg)	0	Angle Center: -90-+90(deg)	0
Speed: 0-10(rad/s)	0.0	Speed: 0-10(rad/s)	0.0
Amplitude: 0-180(deg)	0	Amplitude: 0-180(deg)	0
Offset 1: 0-2(secs)	0	Offset 1: 0-2(secs)	0
Offset 2: 0-2(secs)	0	Offset 2: 0-2(secs)	0
Offset 3: 0-2(secs)	0	Offset 3: 0-2(secs)	0
Offset 4: 0-2(secs)	0	Offset 4: 0-2(secs)	0

Servo Minimum Pulse: 125

Servo Maximum Pulse: 600

Figure 19 The website of the servo Controller

System Simulation

Design Overview

Because fabrication and testing of the robot could not be accomplished during these unfortunate times, our team had to figure out a way to see if our design meets the goals given at the beginning of the course. A system simulation was determined to be the best way to see if the system properly meets the goals given without having any data from testing a built prototype. The system is built on Python 3.7 using the import of "odeint" to solve a set of dynamic equations. First, a free body diagram of the system needs to be drawn to determine what forces act on the paddles.

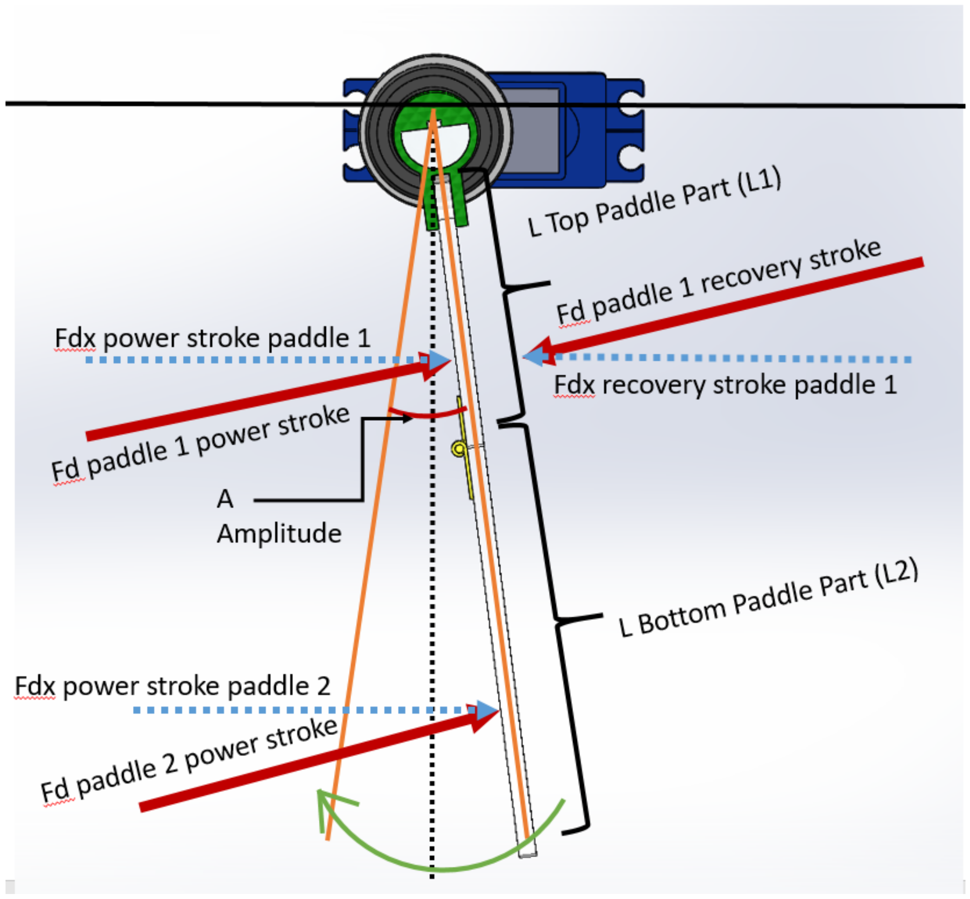

L Top Paddle Part (L1)
Fd paddle 1 recovery stroke
Fdx power stroke paddle 1
Fdx recovery stroke paddle 1
Fd paddle 1 power stroke
A
Amplitude
L Bottom Paddle Part (L2)
Fdx power stroke paddle 2
Fd paddle 2 power stroke

Figure 20

Figure 21. A free body diagram is drawn from theory to show all major forces acting on the system. The green arrow on the bottom references the power stroke of the paddle. The x-components of forces are used to determine the x position and velocity at different times. The magnitude of forces are calculated at 2/3 the length of the paddle segments since the linear velocity increases as the length of the paddle increases (triangular force distribution). The stroke amplitude is defined as the angle between the paddle's final positions for recovery and power strokes outlined by the angled orange lines. The paddle is broken into two sections, a top section which will not hinge on the recovery stroke due to it being directly attached to the shaft, and the

bottom section where the recovery stroke's force is ignored due to the hinge allowing the support to fold. In real world application, there will be a fraction of drag due to this hinging but will be minimal for overall simulation results.

The diagram allows for a system of ordinary differential equations to be derived to solve for the position and the velocity of the system over a function of time. These results can be graphically displayed to see if the velocity of the robot meets the goal of .5 m/s within one length of the robot.

$$\theta = Asin(\omega t + \varphi)$$

$$\dot{\theta} = A\omega \sin(\omega t + \varphi)$$

$$V_{paddle} = \dot{\theta}\frac{2L}{3}$$

$$F_d = \frac{1}{2}\rho C_d A V^2$$

$$\begin{cases} \dot{x} = V \\ \ddot{x} = \frac{1}{m}(F_1 + F_2 - F_b) \end{cases}$$

Figure 21

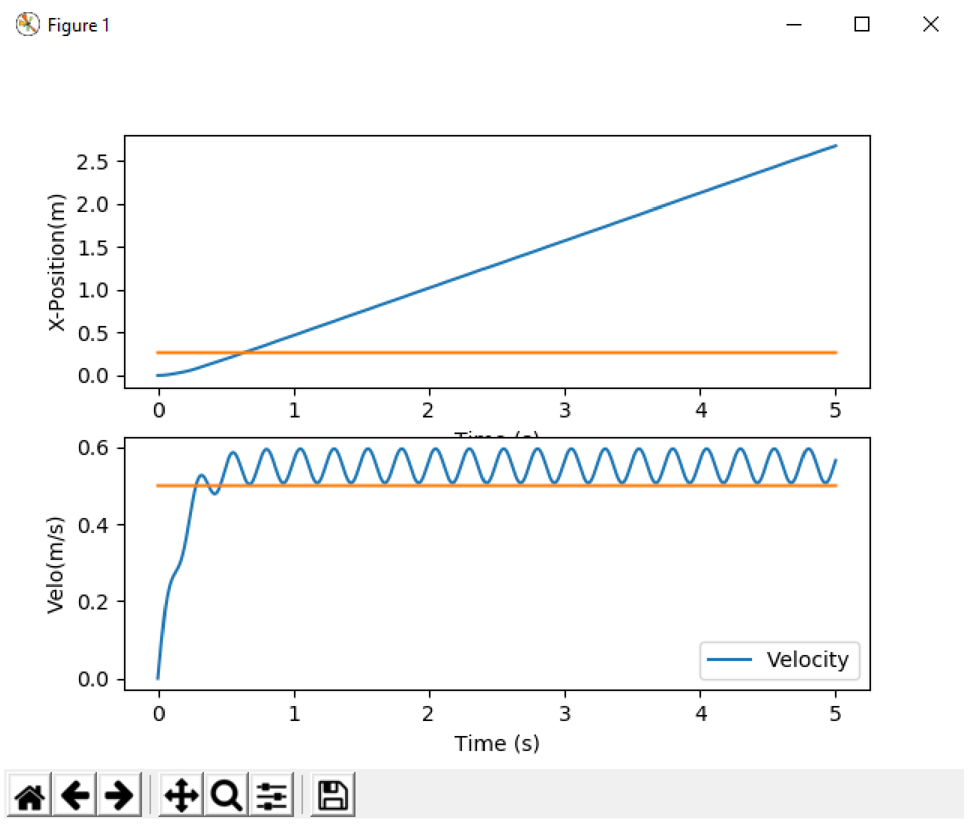


Figure 22

Figure 23a (Top) and Figure 23b (Bottom). Figure 23a displays the system of differential equations that the system simulation code solves. The code is shown in Appendix B. Theta which is the angle of the paddle from the vertical “zero position” is defined as well as it’s derivative with respect to time. These values will display positive and negative, where the code will use the derivative to determine the magnitude of the forces (positive or negative values). The linear velocity can be derived as a function of the angular velocity function over the length of the paddle. The force of drag used is then thrown into the differential equations of x dot equaling the velocity of the robot, and the acceleration is the sum of forces in their respective polarities. Figure 23b shows the position in the x axis (top) and the velocity (bottom) as a function of time. The horizontal orange lines show the given requirements, where the velocity must exceed .5 m/s before the length of the robot is travelled (10.5”).

The simulation results allow the user to enter custom values for all variables into an easy to use Python class to simulate the system with different variable values. The simulation seems to display a reasonable graphical display for the system. The values used to calculate the results in Figure 23 are;

- M (mass) = 3.17 kg (7 lbs)
- Body Frontal Face – W (width) = .324 m (12.75”), H (height) = .0508 m (2”)
- Top Paddle Face – L (length) = .0508m (2”), W (width) = .0508m (2”)
- Bottom Paddle Face - L (length) = .0762m (3”), W (width) = .0508m (2”)

- A (stroke amplitude) = 100 degrees
- ω (frequency) = 2π
- Φ (phase offset per paddle) - Paddle 1 = 0, P2 = π/2, P3 = π, P4 = 3π/2
- Cdp (coefficient of drag paddle) = 1.3
- Cdb (coefficient of drag body) = 3

The generic simulation code used can be found in Appendix B.

Limitations of Design and Future Work

The overall code gives a decent analysis of the system theoretically but has not been tested experimentally to determine the validity of the results. The overall simulation follows a trend, which compared to previous research, seems valid, but cannot be verified without testing the actual prototype. The code is also a two-dimensional problem in which requires four dimensions (x,y,z,t) for proper analysis. The code also assumes basic drag forces and dynamic relationships which could be less accurate than hydrodynamic models. Future research needs to be done on such hydrodynamics and the code should be applied for different research structures to determine the best code fit for the working prototype.

An optimization code which follows this code would help design body and paddle parameters as the frontal areas of the body and paddles can be varied, maximizing velocity and position as outputs. The code can use the basic dynamic model and can potentially open a new area of research on how to optimize dimensions for optimal performance of the vehicle.

Cost Analysis

For our project we found it easier to separate our cost analysis list into two categories. With one category being used for the parts that will be used for the body and the waterproofing of the body. The other category being for the list of parts that will be used for the electronics and the operation controls for the robot. We separated the parts into these categories in order to help estimate how much of our budget would be spent in each categories area.

Mechanical

For this category we placed all the parts and components we would need to manufacture the body and to install our waterproofing measures. This includes the acrylic to build the body and the cost of the gaskets and lip seals we needed to waterproof our robot. We also included the materials for the paddles and the 3D parts that we will need to manufacture. The final cost for these materials came out to be $539.47. The list for the mechanical components is provided in Appendix D.

Electrical

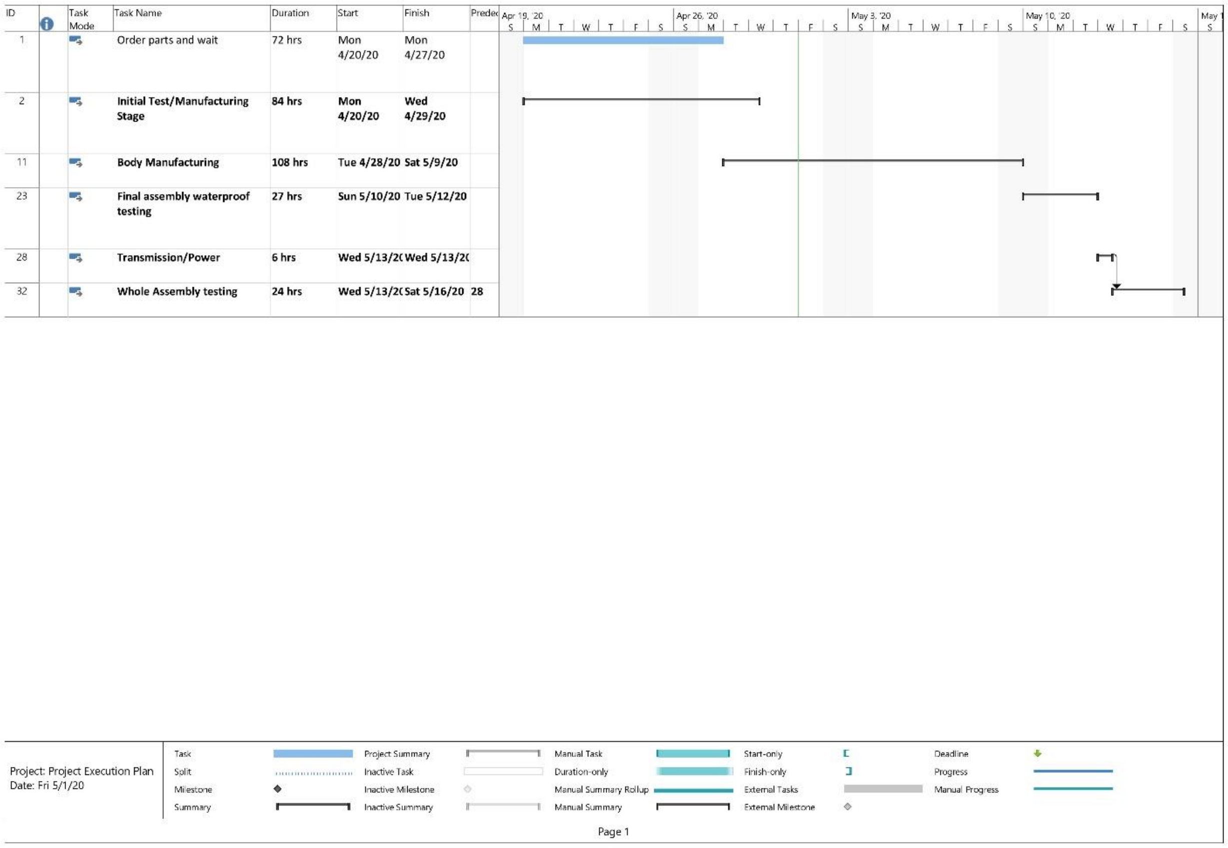
For this category we placed all the parts and components that we would need to power and operate the robot. The electrical components estimate will include costs for the parts necessary to power our robot and ensure that it operates correctly. It will include the costs for the three servo options that we have used for testing and it includes the final servo option that we chose to move forward with. It will include the cost for the materials needed manufacture our control boards and any costs for wires and components necessary for it. We have also included the costs for the raspberry pi, power supply, and our camera. The final cost for these materials came out to be $1000.00. The list for the electrical components is provided in Appendix D.

Overall

After we finalized the calculations of the cost for our parts and materials, we ended up being under our budget of $4000 for our project. By being under budget we are allowed much more room for contingency which gives us financial room to test our systems more and allow us to improve the design. The final total cost estimate for the electrical components came to $1000 and the final total cost for the mechanical components are $539.47. The combined cost for our electrical and mechanical components is $1,539.47. This is less than half of our budget of $4000. This will allow us to use a 100% contingency plan with still room for more improvements if we see fit. A 100% contingency will make the total cost be $3078.94 and will allow us to replace any part if or improve any subsystems if necessary. This will also give us room to test our methods as we see fit. This contingency will give us ample room for any mistakes that may occur during our manufacturing process. A complete parts list and calculation is provided as Appendix D.

Testing Plan

Figure 24 shows an overview of the testing and assembly GANNT Chart. A full Microsoft project file has been submitted alongside this report for detailed viewing.



ID		Task Mode	Task Name	Duration	Start	Finish	Predec
1			Order parts and wait	72 hrs	Mon 4/20/20	Mon 4/27/20	
2			**Initial Test/Manufacturing Stage**	**84 hrs**	**Mon 4/20/20**	**Wed 4/29/20**	
11			**Body Manufacturing**	**108 hrs**	**Tue 4/28/20**	**Sat 5/9/20**	
23			**Final assembly waterproof testing**	**27 hrs**	**Sun 5/10/20**	**Tue 5/12/20**	
28			**Transmission/Power**	**6 hrs**	**Wed 5/13/2(**	**Wed 5/13/2(**	
32			**Whole Assembly testing**	**24 hrs**	**Wed 5/13/2(**	**Sat 5/16/20**	**28**

Project: Project Execution Plan
Date: Fri 5/1/20

Page 1

Figure 234

Mechanical

Mechanical testing will consist of five stages. Stage one will consist of cutting and assembling a small box made of acrylic, similar to the one in figure 25. This will be used to test the sealing of the weld-on 3 acrylic cement. Once the cement is verified another box will be built. This one will have 0.5” holes in two sides. This will allow us to test the cable glands by running the tether in one side and out the other leaving the ends of the tether out of the water and out of harm’s way while we test the glands. This box or a nearly identical box can then be made to test the shaft seals design. The sealing capabilities of these need to be tested and verified early in the process before building commences.

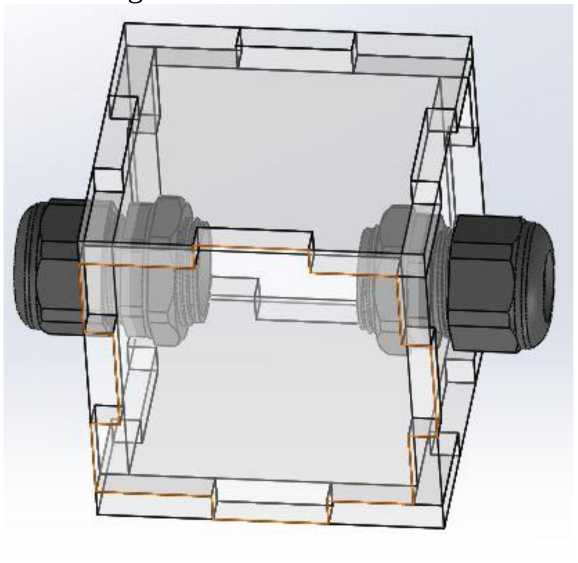


Figure 25

Later, once the body is assembled a further test will be conducted on the body, with all mechanical components assembled correctly, but without any of the electrical components. If this works, we will add the electrical components and do an unpowered test. After that test is successful, the robot will be powered on, and the fully realized design will be tested one last time to ensure everything works correctly, and there is no leakage.

Electrical

Electrical testing will be very straight forward. Once the power supply board is completely soldered, it will need to be powered up and tested to ensure that all voltages are present, and the servo controller operates as expected. First closely inspect tall of the solder joint to ensure proper connection. Test the connections after the visual conformation by securing the board to a nonconductive mat and connecting the board to the bench power supply. From here measure the output voltages with an oscilloscope or multimeter to ensure that they are at the proper 5 and 7.4 volts respectively. If the voltages are not where they should we recommend visually inspecting the connections, then proceeding to connect the oscilloscope to view the waveform ripple from the buck converter. If the output is still not at the desired voltage, then troubleshoot by narrowing down which buck converter is the problem and check the connections and part orientations. For troubleshooting the I2c connection, connect the power board to the Raspberry pi and see if the chip turns on and the outputs are desired, you can connect an oscilloscope to ensure the SDA and SDL are the correct waveforms.

Hazard Analysis

There are several hazards involved with this project including shock, pinch, cut, tripping, irritant, and biohazards. There is a shock hazard from electricity powering this ROV. Especially when in water. A pinch hazard due to the mechanical mechanism driving the paddles. A cut hazard from fabrication and manufacturing. A tripping hazard from the tether. An irritant hazard from Weld-on 3. which is an acrylic cement used to join the acrylic body together. A biohazard from transporting it from water source to water source. Every single one of these hazards and be prevented. To combat shock hazard, we can get the ROV checked by electrical engineering professors, and we can also practice safe wiring and grounding standards. To combat pinch hazard, we can cut power to the ROV whenever working on it,and keep our hands away from the ROV when in operation. To combat cut hazards, we can wear proper safety equipment (gloves and safety glasses) and practice proper methods as well as getting the proper training on said tool. To protect against tripping on the tether, we will have only one person holding the tether and no one else around it. It will also be corded off. Using proper PPE like a mask, gloves, and eye protection, this can be minimized. To prevent a biohazard, we will wash and thoroughly clean the ROV before and after every test and especially before and after it is moved to a different water source.

Design Goals and Did They Meet

From the system simulation, the goal of achieving .5 m/s within the body length of the robot with the design that is currently simulated met the expectations. Figure 23 shows the intersection between the position and velocity functions, and shows the velocity reaching the .5m/s goal before the robot achieves 10 ½" (the length of the robot). The system is theoretical, so experimental data could potentially alter the result of this goal.

Appendices

Appendix A

Code (Python 3) is shown for paddle length analysis. Equations are displayed under the functions.

```
import matplotlib.pyplot as plt

from numpy import *

def Calculations(Length, Width, Kv, Density, Cd):

    lmeter = Length * .0254

    wmeter = Width * .0254

    v = linspace(0,5,1000)

    Fd = .5 * Density * v**2 * Cd * lmeter * wmeter

    Torque = (Fd * lmeter)

    return v,Fd,Torque

def Velocity(Kv,Length):
    lmeter = Length * .0254
    RN = linspace(10**0,10**3,1000)
    velo = RN*Kv/lmeter
    return RN, velo
def Frequency(l,v,SA):
    lmeter = l*.0254
    f = (90/pi)*(v/(lmeter*SA))
    return f
def lengthsatisfy(v,f,SA):
    l = (90/pi)*(v/(f*SA))*39.37
    return l
def freq_vs_torq(density,SA,L,Cd,W):
    lmeter = L * .0254
    wmeter = W*.0254
    f = linspace(0,10,1000)
    t = .5 * density * ((pi/90)*SA*lmeter*f)**2 * Cd * wmeter * lmeter
    return f,t
def PaddleAnalysis():
    Density = 997.05 #kg/m^3
    Kv = .8927 * 10**-6 #m^2/s
    Llowest = 3 #inches for all lengths
    L0 = 3.5
    L1 = 4
    L2 = 4.5
    L3 = 5
    L4 = 5.5
    L5 = 6
    L6 = 6.5
```

```
Lhighest = 7
Width = 2 #inches
Cd = 1.28 #flat plate
SA = 120 #Stroke angle in degrees
frequency = 3 #Hz
#Import From Calculation for all lengths
v, Fdlowest, Tlowest = Calculations(Llowest,Width,Kv,Density,Cd)
v, Fd0, T0 = Calculations(L0,Width,Kv,Density,Cd)
v, Fd1, T1 = Calculations(L1, Width, Kv, Density, Cd)
v, Fd2, T2 = Calculations(L2, Width, Kv, Density, Cd)
v, Fd3, T3 = Calculations(L3, Width, Kv, Density, Cd)
v, Fd4, T4 = Calculations(L4, Width, Kv, Density, Cd)
v, Fd5, T5 = Calculations(L5, Width, Kv, Density, Cd)
v, Fd6, T6 = Calculations(L6, Width, Kv, Density, Cd)
v, Fdhighest, Thighest = Calculations(Lhighest, Width, Kv, Density, Cd)
fl = Frequency(Llowest,v,SA)
f0 = Frequency(L0, v, SA)
f1 = Frequency(L1, v, SA)
f2 = Frequency(L2, v, SA)
f3 = Frequency(L3, v, SA)
f4 = Frequency(L4, v, SA)
f5 = Frequency(L5, v, SA)
f6 = Frequency(L6, v, SA)
fh = Frequency(Lhighest, v, SA)
f, tl = freq_vs_torq(Density,SA,Llowest,Cd,Width)
f, t0 = freq_vs_torq(Density, SA, L0, Cd, Width)
f, t1 = freq_vs_torq(Density, SA, L1, Cd, Width)
f, t2 = freq_vs_torq(Density, SA, L2, Cd, Width)
```

```
f, t3 = freq_vs_torq(Density, SA, L3, Cd, Width)
f, t4 = freq_vs_torq(Density, SA, L4, Cd, Width)
f, t5 = freq_vs_torq(Density, SA, L5, Cd, Width)
f, t6 = freq_vs_torq(Density, SA, L6, Cd, Width)
f, th = freq_vs_torq(Density, SA, Lhighest, Cd, Width)
#calculate frequency
length = lengthsatisfy(1.75,frequency,SA)
print("Length to satisfy frequency; l >= {:.2f} inches".format(length))
plt.plot(v,Fdlowest, label="Length = {:} inches".format(Llowest))
plt.plot(v, Fd0, label="Length = {:} inches".format(L0))
plt.plot(v, Fd1, label="Length = {:} inches".format(L1))
plt.plot(v, Fd2, label="Length = {:} inches".format(L2))
plt.plot(v, Fd3, label="Length = {:} inches".format(L3))
plt.plot(v, Fd4, label="Length = {:} inches".format(L4))
plt.plot(v, Fd5, label="Length = {:} inches".format(L5))
plt.plot(v, Fd6, label="Length = {:} inches".format(L6))
plt.plot(v, Fdhighest, label="Length = {:} inches".format(Lhighest))
plt.xlim(0,2.5)
plt.ylim(0,70)
plt.xlabel("Velocity of Legs (m/s)")
plt.ylabel("Force of Drag on Legs (N)")
plt.title("Force drag on Legs")
plt.legend()
plt.grid()
plt.show()
plt.plot(v, Tlowest, label="Length = {:} inches".format(Llowest))
plt.plot(v, T0, label="Length = {:} inches".format(L0))
plt.plot(v, T1, label="Length = {:} inches".format(L1))
```

```
    plt.plot(v, T2, label="Length = {:} inches".format(L2))
    plt.plot(v, T3, label="Length = {:} inches".format(L3))
    plt.plot(v, T4, label="Length = {:} inches".format(L4))
    plt.plot(v, T5, label="Length = {:} inches".format(L5))
    plt.plot(v, T6, label="Length = {:} inches".format(L6))
    plt.plot(v, Thighest, label="Length = {:} inches".format(Lhighest))
    #plt.plot(x,servohighest,'k',label="Highest servo torque",linewidth=2)
    plt.xlim(0,2.5)
    plt.ylim(0, 13)
    plt.ylabel("Motor Torque Needed (N*m)")
    plt.xlabel("Velocity of Leg (m/s)")
    plt.title("Torque needed for motor")
    plt.legend()
    plt.grid()
    plt.show()
    plt.plot(v, fl, label="Length = {:} inches".format(Llowest))
    plt.plot(v, f0, label="Length = {:} inches".format(L0))
    plt.plot(v, f1, label="Length = {:} inches".format(L1))
    plt.plot(v, f2, label="Length = {:} inches".format(L2))
    plt.plot(v, f3, label="Length = {:} inches".format(L3))
    plt.plot(v, f4, label="Length = {:} inches".format(L4))
    plt.plot(v, f5, label="Length = {:} inches".format(L5))
    plt.plot(v, f6, label="Length = {:} inches".format(L6))
    plt.plot(v, fh, label="Length = {:} inches".format(Lhighest))
    #plt.plot(x,servofrequency, 'k',linewidth=3, label="Servo Frequency Required = {:.2f} 
Hz".format(frequency))
    plt.xlim(0, 2.5)
    plt.ylim(0,8)
    plt.ylabel("Frequency (Hz)")
    plt.xlabel("Velocity of Legs (m/s)")
    plt.title("Frequency vs Leg Velocity for Motor Selection")
    plt.grid()
    plt.legend()
    plt.show()
    plt.plot(f, tl, label="Length = {:} inches".format(Llowest))
    plt.plot(f, t0, label="Length = {:} inches".format(L0))
    plt.plot(f, t1, label="Length = {:} inches".format(L1))
    plt.plot(f, t2, label="Length = {:} inches".format(L2))
    plt.plot(f, t3, label="Length = {:} inches".format(L3))
    plt.plot(f, t4, label="Length = {:} inches".format(L4))
    plt.plot(f, t5, label="Length = {:} inches".format(L5))
    plt.plot(f, t6, label="Length = {:} inches".format(L6))
    plt.plot(f, th, label="Length = {:} inches".format(Lhighest))
    plt.xlim(0, 5)
    plt.ylim(0, 8)
    plt.ylabel("Torque Required by Motor(N*m)")
    plt.xlabel("Frequency (Hz)")
    plt.title("Frequency vs Leg Velocity for Motor Selection")
    plt.grid()
    plt.legend()
    plt.show()
PaddleAnalysis()
```

Appendix B

Code (Python 3) is shown for body simulation analysis. Equations are displayed under the functions.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from CollisionCheck import motion
class Element():
    def __init__(self,L,W, Cd):
        self.L = L
        self.W = W
        self.A = L*W
        self.Cd = Cd
class Krill():
    def __init__(self,m, Body,Paddle_1,Paddle_2):
        self.m = m
        self.body = Body
        self.p1 = Paddle_1
        self.p2 = Paddle_2
class Medium():
    def __init__(self,ρ):
        self.ρ = ρ
class Motion():
    def __init__(self,A,ω,ϕ):
        self.A = A
        self.ω = ω
        self.ϕ = ϕ
class Simulator():
    def __init__(self,Krill,Motion,Medium,X0=np.array([0,0]),tf = 5):
        self.t = np.linspace(0,tf,1000*tf)
        self.X0 = X0
        self.mot = Motion
        self.med = Medium
        self.krill = Krill
    def F_d_paddle1(self,bodyvel,theta,dtheta):
        fd = (0.5*self.med.ρ*self.krill.p1.Cd*self.krill.p1.A*((dtheta*2*self.krill.p1.L/3)-
bodyvel)**2)*np.cos(theta)*np.sign(dtheta) #look into body velo offset (add body velo - flow
velo)
        return sum(fd)
    def F_d_paddle2(self,bodyvel,theta,dtheta):
        fd = (0.5*self.med.ρ*self.krill.p2.Cd*self.krill.p2.A*((dtheta*(self.krill.p1.L +
2*self.krill.p2.L/3)-bodyvel)**2)*np.cos(theta))
        return sum(fd)
        def F_d_body(self,vel):
        return (0.5*self.med.ρ*self.krill.body.Cd*self.krill.body.A*(vel)**2)
        def sys(self,X,t):
        x, xdot = X[0], X[1]
        deriv = np.zeros_like(X)
        theta = self.mot.A*np.sin(self.mot.ω*t + self.mot.ϕ)
        omega = self.mot.A*self.mot.ω*np.cos(self.mot.ω*t+self.mot.ϕ) #theta dot
        #look at equilibrium
        deriv[0] = xdot #position to velo
        deriv[1] = (2*self.F_d_paddle1(xdot,theta,omega)/self.krill.m +
2*self.F_d_paddle2(xdot,theta,omega)/self.krill.m - self.F_d_body(xdot)/self.krill.m)
#acceleration from f=ma
        return deriv
    def simulate(self):
        self.X = odeint(self.sys,self.X0,self.t)
    def graph(self):
        plt.figure(1)
        plt.subplot(211)
        plt.plot(self.t,self.X[:,0],label="Position")
        plt.plot(self.t, .2667+0*self.t)
        plt.xlabel("Time (s)")
        plt.ylabel("X-Position(m)")
        plt.subplot(212)
        plt.plot(self.t,self.X[:,1],label="Velocity")
        plt.plot(self.t,0*self.t+.5)
        plt.ylabel("Velo(m/s)")
        plt.xlabel("Time (s)")
        plt.legend()
        plt.show()
if __name__ == "__main__":
    Medium = Medium(ρ = 1000)
    Body_1 = Element(L = 0.0508, W=0.324, Cd = 3)
    Paddle_1 = Element(L= 0.0508, W =0.0508, Cd = 1.3)
    Paddle_2 = Element(L =0.0762, W=0.0508, Cd = 1.3)
    Krill = Krill(3.17,Body_1, Paddle_1, Paddle_2)
    Motion = Motion(A=np.radians(100),ω=2*np.pi,ϕ=np.array([0,np.pi/2,np.pi,3*np.pi/2]))
    Sim =Simulator(Krill,Motion,Medium,X0=np.array([0,0]),tf = 5)
    Sim.simulate()
    Sim.graph()
```

Appendix C: Cost Estimate Lists

Item Description/Name	Part #	Quantity	Cost/Unit	Cost for all Units
12in x 12in x 0.25 in Acrylic sheet	4615T37	2	$ 12.43	$ 24.86
24in x 24in x 0.25 in Acrylic sheet	4615T47	1	$ 45.14	$ 45.14
12in x 12in x 0.5in acrylic sheet	4615T51	2	$ 24.66	$ 49.32
24in x 24in x 0.125 in Acrylic sheet	1615T94	1	$ 23.61	$ 23.61
4-40 thread- threaded inserts	94459A260	1	$ 9.57	$ 9.57
Installation tip for 4-40 threaded inserts	92160A115	1	$ 15.55	$ 15.55
8-32 Thread-Threaded inserts	94459A310	1	$ 9.84	$ 9.84
Installation tip for 8-32 threaded inserts	92160A123	1	$ 15.55	$ 15.55
Soldering Iron for threaded inserts	7662A696	1	$ 30.30	$ 30.30
4-40 Thread-100deg CS-3/4in len Sealing Screws	98070A520	4	$ 6.57	$ 26.28
8-32 Thread-100deg CS-5/8in len Sealing Screws	98070A340	2	$ 9.43	$ 18.86
Nonwhitening Acrylic Cement	7517A2	2	$ 18.00	$ 36.00
3M polyurethane sealant	67015A75	2	$ 28.85	$ 57.70
Buna-N gasket sheet 24in x 24in x 1/16 in	8525T22	2	$ 10.27	$ 20.54
Tight tolerance 6061 aluminum rods-0.5in Dia	9062K31	1	$ 17.19	$ 17.19
Plastic ball bear - 0.5 in diameter	6455K64	8	$ 10.26	$ 82.08
Rotary Shaft seal	1199N61	8	$ 3.15	$ 25.20
Cable Gland (PG19)	LKM-CGL	1	$ 11.89	$ 11.89
Hatchbox ABS 3D Printing Filament		1	$ 19.99	$ 19.99
			Subtotal	$ 539.47

Item Description/Name	Part #	Quantity	Cost/Unit	Cost for all Units
5V Buck Converter	TPS56339	1	$1.28	$1.28
	SRP6540-5R6M	1	$1.49	$1.49
	C0805C104M5RACTU	2	$0.10	$0.20
	C1608X5R1E106M080AC	2	$0.76	$1.52
	C2012X7S1A226M125AC	2	$0.97	$1.94
	RC0603FR-0730RL	1	$0.10	$0.10
	CRCW060310K0FKEA	1	$0.10	$0.10
	CRCW060352K3FKEA	1	$0.10	$0.10
	MFU0805FF00500P100	1	$0.63	$0.63
7.4V Buck Converter	TPS56339	4	$1.28	$5.12
	SRP6540-6R8M	4	$1.49	$5.96
	C0805C104M5RACTU	8	$0.10	$0.80
	C1608X5R1E106M080AC	8	$0.76	$6.08
	C2012X7S1A226M125AC	12	$0.97	$11.64
	RC0603FR-0730RL	4	$0.10	$0.40
	CRCW060310K0FKEA	4	$0.10	$0.40
	RMCF0603FT82K5	4	$0.10	$0.40
	MFU0805FF00750P100	4	$0.63	$2.52
Resistors to set 7.4V to 6V	CRCW060364K9FKEA	4	$0.10	$0.40
Extra Fuses:		0		$0.00
5V	MFU0805FF00500P100	2	$0.63	$1.26
7.4V	MFU0805FF00750P100	8	$0.63	$5.04
Servo Controller	PCA9685	1	$2.46	$2.46
	C0805C104M5RACTU	1	$0.10	$0.10
	CRCW060310K0FKEA	3	$0.10	$0.30
	CRGCQ0603F220R	10	$0.10	$1.00
Headers	M20-9990345	10	$0.18	$1.80
12 Gauge Wire interconnects	GB007	1	$18.66	$18.66
Jumper wires for Raspi	4330127279	1	$5.99	$5.99
12 Gauge Wire		0		$0.00
	22759/32-12-0	2	$1.20	$2.40
	55PC0211-12-9CS2502	2	$1.57	$3.14
PCB Quote	JLCPCB	1	$30.00	$30.00
Servo Options	HS646WP	9	$42.99	$386.91
	SER0030	1	$31.87	$31.87
	ANNIMOS	1	$38.99	$38.99
Raspberry Pi	Raspberry PI 4GB SBC	2	$66.88	$133.76
Tether	CVS LK1CAT6SF 12/3	100	$2.35	$235.00
Raspberry Pi Camera	3099	1	$29.95	$29.95
Power Supply	2219815	1	$30.29	$30.29
			subtotal	$1,000.00

Item Description/Name	Part #	Quantity	Cost/Unit	Cost for all Units
12in x 12in x 0.25 in Acrylic sheet	4615T37	2	$ 12.43	$ 24.86
24in x 24in x 0.25 in Acrylic sheet	4615T47	1	$ 45.14	$ 45.14
12in x 12in x 0.5in acrylic sheet	4615T51	2	$ 24.66	$ 49.32
24in x 24in x 0.125 in Acrylic sheet	1615T94	1	$ 23.61	$ 23.61
4-40 thread- threaded inserts	94459A260	1	$ 9.57	$ 9.57
Installation tip for 4-40 threaded inserts	92160A115	1	$ 15.55	$ 15.55
8-32 Thread-Threaded inserts	94459A310	1	$ 9.84	$ 9.84
Installation tip for 8-32 threaded inserts	92160A123	1	$ 15.55	$ 15.55
Soldering Iron for threaded inserts	7662A696	1	$ 30.30	$ 30.30
4-40 Thread-100deg CS-3/4in len Sealing Screws	98070A520	4	$ 6.57	$ 26.28
8-32 Thread-100deg CS-5/8in len Sealing Screws	98070A340	2	$ 9.43	$ 18.86
Nonwhitening Acrylic Cement	7517A2	2	$ 18.00	$ 36.00
3M polyurethane sealant	67015A75	2	$ 28.85	$ 57.70
Buna-N gasket sheet 24in x 24in x 1/16 in	8525T22	2	$ 10.27	$ 20.54
Tight tolerance 6061 aluminum rods-0.5in Dia	9062K31	1	$ 17.19	$ 17.19
Plastic ball bear - 0.5 in diameter	6455K64	8	$ 10.26	$ 82.08
Rotary Shaft seal	1199N61	8	$ 3.15	$ 25.20
Cable Gland (PG19)	LKM-CGL	1	$ 11.89	$ 11.89
Hatchbox ABS 3D Printing Filament		1	$ 19.99	$ 19.99
5V Buck Converter	TPS56339	1	$1.28	$1.28
	SRP6540-5R6M	1	$1.49	$1.49
	C0805C104M5RACTU	2	$0.10	$0.20
	C1608X5R1E106M080A	2	$0.76	$1.52
	C2012X7S1A226M125A	2	$0.97	$1.94
	RC0603FR-0730RL	1	$0.10	$0.10
	CRCW060310K0FKEA	1	$0.10	$0.10
	CRCW060352K3FKEA	1	$0.10	$0.10
	MFU0805FF00500P100	1	$0.63	$0.63
7.4V Buck Converter	TPS56339	4	$1.28	$5.12
	SRP6540-6R8M	4	$1.49	$5.96
	C0805C104M5RACTU	8	$0.10	$0.80
	C1608X5R1E106M080A	8	$0.76	$6.08
	C2012X7S1A226M125A	12	$0.97	$11.64
	RC0603FR-0730RL	4	$0.10	$0.40
	CRCW060310K0FKEA	4	$0.10	$0.40
	RMCF0603FT82K5	4	$0.10	$0.40
	MFU0805FF00750P100	4	$0.63	$2.52
Resistors to set 7.4V to 6V	CRCW060364K9FKEA	4	$0.10	$0.40
Extra Fuses:		0		$0.00
5V	MFU0805FF00500P100	2	$0.63	$1.26
7.4V	MFU0805FF00750P100	8	$0.63	$5.04
Servo Controller	PCA9685	1	$2.46	$2.46
	C0805C104M5RACTU	1	$0.10	$0.10
	CRCW060310K0FKEA	3	$0.10	$0.30
	CRGCQ0603F220R	10	$0.10	$1.00
Headers	M20-9990345	10	$0.18	$1.80
12 Gauge Wire interconnects	GB007	1	$18.66	$18.66
Jumper wires for Raspi	4330127279	1	$5.99	$5.99
12 Gauge Wire		0		$0.00
	22759/32-12-0	2	$1.20	$2.40
	55PC0211-12-9CS2502	2	$1.57	$3.14
PCB Quote	JLCPCB	1	$30.00	$30.00
Servo Options	HS646WP	9	$42.99	$386.91
	SER0030	1	$31.87	$31.87
	ANNIMOS	1	$38.99	$38.99