Lionfish - Phase V

Owen Blaufuss (RBE), Kathleen Cochran (ME), Atharva Dikshit (RBE), Julia Meisser (ME/RBE), Justin Mitchell (ECE/CS), Brandon Snapperman (RBE)

Advised by Professor Craig B. Putnam,
Professor Bradley A. Miller, and
Professor William R. Michalson

April 27, 2022

Abstract

Lionfish, a species of venomous fish indigenous to the Indo-Pacific Ocean, are an invasive species along the southeast coast of the United States and the Gulf of Mexico. Due to the absence of any natural predators, their exponential rate of reproduction, and their ability to survive in a wide range of habitats, their population has been growing rapidly in the Atlantic Ocean. As a result, they are destroying the coral reefs and fish that live in and around the reefs, thus damaging the local ecosystem. The Lionfish MQP aims to mitigate the spread of the lionfish population by creating a remotely operated vehicle (ROV) to harvest lionfish. This year's team plans on expanding and improving the work done by the previous iterations of this project. The team's focus is to design and implement a new spearing mechanism, employ machine learning and computer vision to detect lionfish, and implement autonomous underwater navigation.

Acknowledgements

First and foremost, we would like to thank our advisors Professor Craig B. Putnam, Professor Bradley A. Miller, and Professor William R. Michalson for continuing to support, advise, and fund the Lionfish MQP.

We would also like to extend our gratitude to the WPI recreation building staff and lifeguards for allowing us to house our MQP in the Robot Pits and test our robot in the WPI pool.

Finally, we would like to thank the previous Lionfish MQP teams, particularly Stephanie Johnson, for sharing their work with us and helping us to continue this MQP.

Table of Contents

List of Figures

List of Equations

1 Introduction

Lionfish (Pterois miles or Pterois volitans) are a species of venomous fish with a striking striped appearance, indigenous to the Indo-Pacific Ocean. Due to unfortunate accidents in the 1990s, they spread to the southeast coast of the United States, the Caribbean, and the Gulf of Mexico, making them an invasive species: a non-native organism that has serious detrimental effects on the local ecosystem (Wurzbacher, 2011).

Figure 1: Pterois Miles/ Pterois Volitans (Lionfish) in the Ocean

Lionfish are particularly damaging to local ecosystems because of their rapid rate of reproduction, voracious appetite, and lack of natural predators in the region. They can live in a wide range of habitats, and their range has only expanded further in the past decade (Bottenus, 2017). As they spread, their negative impact affects not only the health of marine ecosystems but also the livelihood of coastal residents who rely on them.

Without predator species managing the lionfish population, human efforts have tried to take their place. This has been done through spearfishing competitions and training local sharks and other predators to hunt lionfish. Our MQP aims to help alleviate the spread of the lionfish infestation by continuing the work of the Lionfish project. The goal of the Lionfish project is to make a remotely operated vehicle (ROV) that kills and captures lionfish so they may be brought back to the mainland and sold to local restaurants, where they are a popular delicacy. An efficient, autonomous robot that could harvest large amounts of lionfish in one trip would be beneficial to both the environment and local economies where lionfish are a prevalent problem.

This year’s team seeks to expand and improve on the work of the earlier iterations of the project. Our team's goal is to revamp the electronics chamber, design and implement a new spearing and containment mechanism, clean up and improve upon the existing code to make it more reliable and easier to understand.

2 Background

2.1 Lionfish

Lionfish are a species originally indigenous to the Indo-Pacific Ocean but are now infesting and destroying ecosystems along the southeast coast of the United States, the Caribbean, and the Gulf of Mexico. Lionfish were first reported in the mid-Atlantic coast around the mid 1980's. These fish are members of the Pterois genus of the family Scorpaenidae which are a venomous predatory marine fish. Due to their venomous and predatory nature, lionfish have become an infestation that needs to be contained and controlled. Since Lionfish do not have any natural predators in the Atlantic, they have been able to rapidly increase in population size with females laying more than two million eggs per year (Côté et al., 2013). This invasive species has been found from the Gulf of Mexico all the way up to the coast of Maine. The massive population size of Lionfish has impacted the environment of the native reefs and fish who live on those reefs. This impact is compounding the damage already caused by climate change.

In their natural habitat the lionfish has many predators such as moray eels, groupers, sharks, and Bobbitt worms. (The Science, 2019) These predators have built up a tolerance or found ways around the venomous spines to eat the lionfish. In the Indo-Pacific, this predation keeps the lionfish population from rapidly increasing due to their faster than normal reproduction rate.

Without predation in the Atlantic, Lionfish populations have rapidly increased, necessitating human intervention to control their population. Currently the most effective method of controlling and decreasing the lionfish population is SCUBA divers spearfishing for them. This spearfishing is often incentivized using competitions with cash prizes. Some have even tried to make lionfish traps, or train sharks and local predators to hunt lionfish. (Busiello)

2.2 Project Requirements

2.2.1 Goals for Functionality

The goal of the Lionfish project is to build a battery-powered autonomous robotic vehicle to perform this hunting task. This vehicle would be brought to a lionfish-infested location by human operators and released into the water, upon which it will begin exploring the reef on its own, using cameras to navigate and search for lionfish. Once a lionfish is positively identified, the robot will maneuver close enough to kill it with a spear mechanism, then stow the fish in some type of containment device attached to the robot. It should then be able to reset or reload the spear mechanism and continue hunting lionfish until recalled to the surface or running out of storage space or battery. The end goal is to manage all these limiting factors such that the robot matches or exceeds a human diver's efficiency at killing and collecting lionfish. After resurfacing, the human operators will be able to retrieve the robot and all the lionfish collected during operation for sale, consumption, or disposal. The robot can then be recharged and re-armed to perform this task as many times as needed.

2.2.2 Challenges of Underwater Robotics

The underwater environment in which the lionfish robot will need to function poses some unique and interesting challenges. An underwater robot needs to maintain a constant buoyancy and be carefully balanced to have control over all six possible axes of motion (surge, heave, sway, yaw, pitch, and roll) possible when submerged. Any external components intended as a long-term solution for the robot must be made of materials which can resist saltwater corrosion and the pressure of submersion. Internal electrical components must be kept in a watertight area. Sensors should be monitoring for leaks, ready to safely surface the robot in the case of emergency and keep the delicate electronics dry.

2.2.3 Safety & Maintenance

A robot intended to explore a reef environment unsupervised must be equipped with robust safety features to minimize potential harm. Because the robot will be sharing space with both humans and native fish, there is a risk of the robot accidentally firing and potentially injuring an unintended target. Both code and physical safeguards must be in place to minimize this possibility. The reef itself must be protected as a valuable and vulnerable living thing, too. The robot needs to be maneuverable enough and aware enough of its surroundings to avoid running into the reef and potentially damaging the coral formations. The robot must not have any tethers or cords which could drag or become tangled.

The safety of the human operators of the ROV is also a serious consideration. The robot must not be able to fire its speargun if a human is present or while being reloaded. The lionfish collected must be contained in a way that allows handling of the robot without having to touch the fish and their venomous spines directly.

2.3 Robotic Platform

To simplify the challenges of functioning underwater, the robot has been built into an existing submersible robotic platform. The one currently being used is the BlueROV2 by Blue Robotics. It is configured with the heavy lift option which includes an upgraded configuration of eight thrusters providing full six-degree-of-freedom control and feedback stability (BLUEROV2 2021). “Adjustable gain levels allow the operator to have precision control at extremely low speeds as well as high power to overcome currents and carry heavy loads” (BLUEROV2 2021). This gives the BlueROV2 a higher level of performance to handle all the additional gear and weight the team will be adding to the robot. “The BlueROV2 is rated to a depth of 100 meters (330 ft); this rating is limited by several factors, including the crush depth of the 4” acrylic watertight enclosure tube, with some safety factor” (BLUEROV2 2021). The BlueROV2 is currently “controlled by a drone flight controller running the open-source ArduSub subsea vehicle control firmware. As part of the ArduPilot project, it brings to the ROV a vast number of features, capabilities, and an extensive user community. At the surface, the pilot controls the ROV through a laptop computer and gamepad controller. The open-source QGroundControl application acts as the user interface, providing the live video stream, sensor feedback and information, and the ability to change settings and configuration” (BLUEROV2 2021). The open-source software and modular frame design has enabled the team to have a level of customizability and flexibility that other platforms do not provide (BLUEROV2 2021).

2.4 Computer Vision

Computer vision is the branch of computer science that utilizes machine learning to analyze, sense and understand the robot's environment through digital images and videos (Seier, 2021). Unlike a human brain, the images and videos are converted into data that are then processed through a computer algorithm to recognize patterns.

Computer vision, although it may look trivial, poses several challenges during implementation (Brownlee, 2019). One such challenge is its restrictive capability. Human vision is usually able to correctly identify the object and extract valuable information in various lighting conditions. The same cannot be true for computers; with changes in the lighting conditions, a computer may register the same object as different in different lighting. Hence computer vision algorithms tend to solve extremely constrained problems. Nevertheless, there have been several advances in computer vision technology with numerous applications like optical character recognition (OCR), medical imaging, motion capture, fingerprint recognition and biometrics, etc. (Brownlee, 2019).

Closely integrated computer vision system aids the robot to "see" its environment and make autonomous decisions regarding its surroundings such as how far it is from an obstacle or whether the robot has encountered its target. Furthermore, computer vision models can be trained to process information faster than the human brain and explore areas with varying light spectrums like ultraviolet and infrared light, which would be invisible to the human eye (Seier, 2021). It also enables the robot to collect valuable data from areas that are remote or dangerous.

2.5 Collection Methods

Phase one of this project used a revolver-like mechanism that could hold eight custom spearheads on the robot at a time. The spearheads would surface after firing so the spearhead and lionfish could be retrieved at the surface. However, this provided a very limited yield per run, and retrieving the lionfish in the open water would likely prove extremely difficult. Because of this, subsequent iterations have investigated potential collection and containment mechanisms that would store the lionfish on-board the robot and would be automatically resettable or reloadable. We expanded on this research and fabricated prototypes for the most promising of these mechanisms.

2.5.1 Existing Technologies

Lionfish are hunted primarily through spearfishing. The most common tools for lionfish hunting are spear guns, pole spears, and Hawaiian slings (lionfish.co). Spearguns are usually propelled by pneumatics, springs, or rubber bands, and shoot a tipped spear shaft at the lionfish. Pole spears are made of long pointed shafts, usually with an elastic loop at the handle for propulsion. Hawaiian slings are a combination of the two, usually consisting of a long spear and a tube or handle through which it is shot using a rubber band.

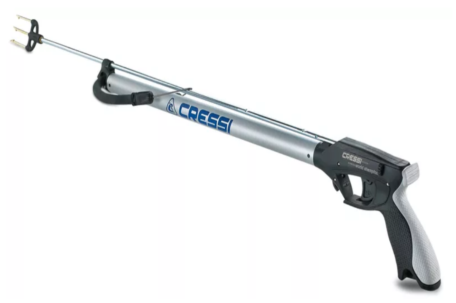


Figure 2: Pneumatic Speargun

(Cressi Apache Aluminum Speargun Lion-Fish Edition W/Stainless Steel Shaft/Sling)

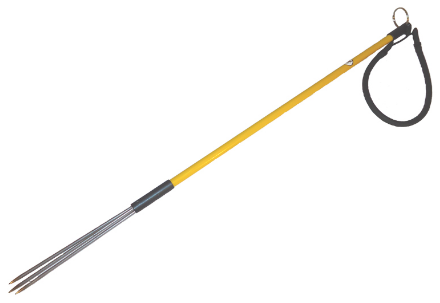


Figure 3: Traditional Spear

(30" fixed barbed paralyzer tip fiberglass pole spear)

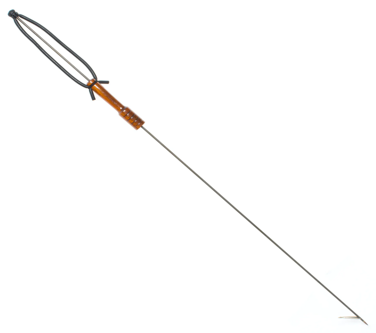


Figure 4: Hawaiian Sling

(Hammerhead Hawaiian Sling Spearfishing Combo Kit)

Various methods are used for containing the lionfish during dives. The spear-shaft hunting method uses the shaft of the spear or speargun to hold the lionfish, like a shish kebab. The shaft will often have a specialized tip and stopper to prevent the lionfish from sliding off the spear or too close to the hunter’s hand. This method causes very little drag, which would be beneficial for our design. However, it would pose a safety threat to humans interacting with the robot due to the exposed spines of the lionfish. In addition, it has the potential to cause interference with the robot’s cameras and moving parts, as well as the potential to attract predators. The lionfish are also more likely to break off and float away than they would be if they were kept in an enclosed containment unit. However, such containment units do exist, such as the Zookeeper™. The

Zookeeper™ is a cylindrical containment unit with a one-way valve that allows the lionfish to be inserted using the spear, then holds them securely in the containment unit while the spear is pulled back through to be used again. The main drawback of the Zookeeper™ is that it adds a significant amount of drag, which would make it difficult to use for our application and potentially cause issues for the robot when trying to navigate. Another containment method is the catch bag. Catch bags are usually made of a strong mesh and often have a one-way valve at the top like a Zookeeper™, but because they are made of mesh they provide significantly less drag when being moved through the water. While they can be highly effective, mesh catch bags can sometimes get in the way of the robot as well and may prove difficult to constrain as they become fuller. This is something that previous phases began to investigate, but due to COVID they were not able to completely vet all their prototypes.

2.5.2 Previous Iterations

The Phase I Lionfish MQP group created an eight-spear revolver mechanism that was designed to shoot buoyant spearheads at the lionfish. A rack and pinion system would cock the spear, then allow the spearheads to be fired at the lionfish. Once the lionfish was speared, the spearhead and the lionfish would float to the surface where they could be retrieved, and a new spearhead would rotate into the firing position (Lombardi et. al., 2018). However, this only allowed for eight shots per run, and collecting the lionfish on the surface would be very inefficient. Because of this, Phase III began investigating other possible spearing mechanisms.

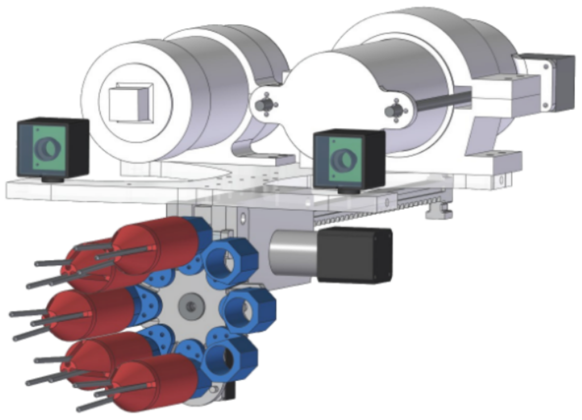


Figure 5: Phase I's Revolver Design

Phase III came up with two possible spear designs, which they dubbed the arc spear and the spin spear. The arc spear was based on the motion divers perform when inserting a lionfish into a containment unit, such as a Zookeeper (Abadjiev et. al., 2020).

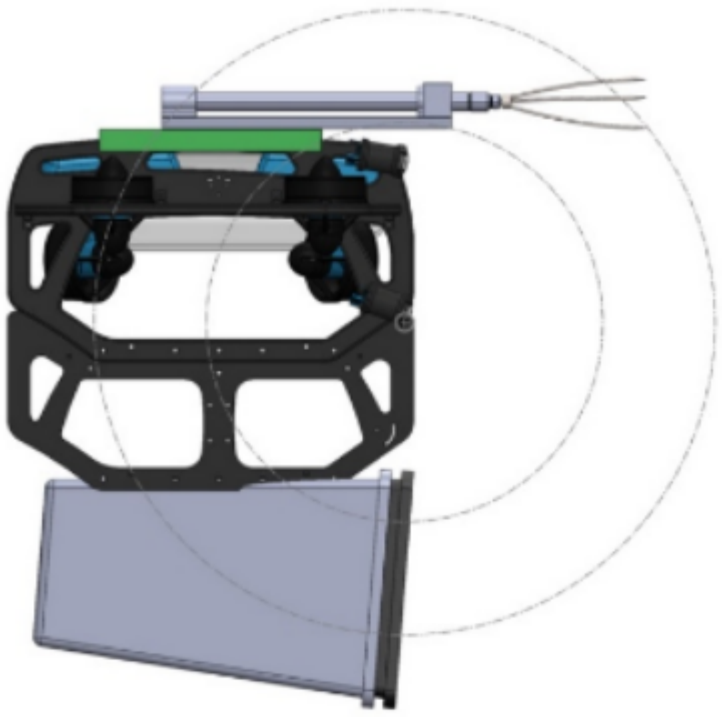


Figure 6: Phase III team's Arc Spear Design

The spin spear was packaged more tightly and utilized a linear slide to rotate 180 degrees to spear or store the lionfish. It was designed to spear the lionfish in the front and rotate backward to store them in a containment system in the back.

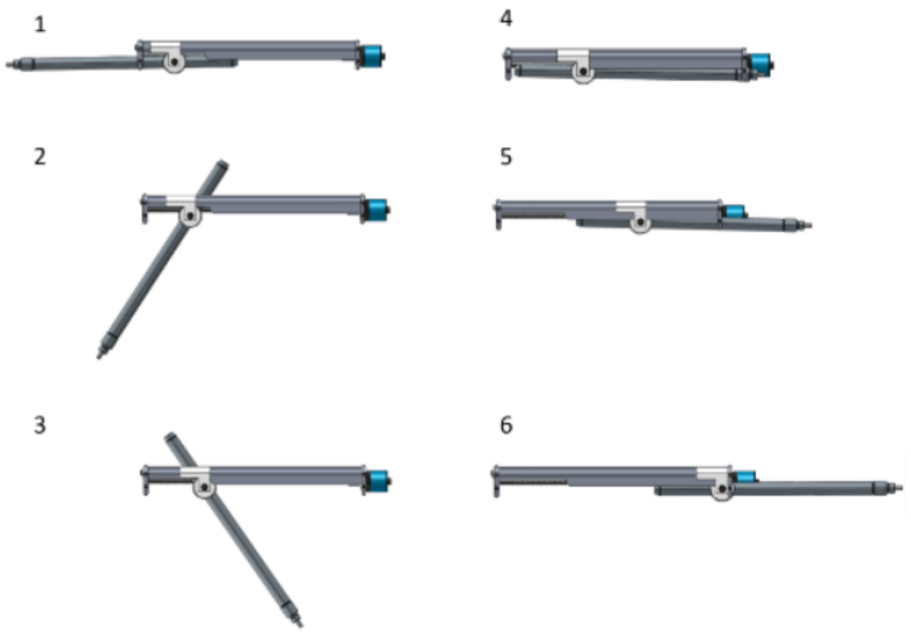


Figure 7: Phase III team's Spin Spear

Both designs used a pneumatic cylinder. However, the team later ran into issues with using a pneumatic system underwater. The air pressure required to run the pneumatics was less than the external water pressure, so the air could not be vented externally when the cylinder fired. This caused a damping effect on the cylinder and did not allow it to reach the 3 m/s firing speed that was calculated by a previous MQP team. To solve this problem, they needed to add an evacuation chamber and a pump that could vent the evacuated air into the water after every shot. However, our team found this design to be rather unnecessarily complicated. We began investigating ways in which we might be able to simplify the design and eliminate the necessity of a pneumatic system, as aside from being quite complex, it had the potential to create buoyancy issues and add more points of failure to the overall system.

Previous iterations also had various prototypes of containment methods for the lionfish. Phase III had several prototype designs, although due to COVID they were not able to definitively test which design would be the best. Their first design resembled a DIY Zookeeper and was made from a modified 12-gallon bucket from Home Depot. They found that this solution was very robust but produced a significant amount of drag and made navigating with the robot nearly impossible. In addition, the lid was not quite stiff enough and would sometimes have trouble pulling the objects being used to simulate lionfish off the spear. However, since they were not using real lionfish, simply objects meant to simulate lionfish, it may be worthwhile to do more extensive testing on such a prototype.

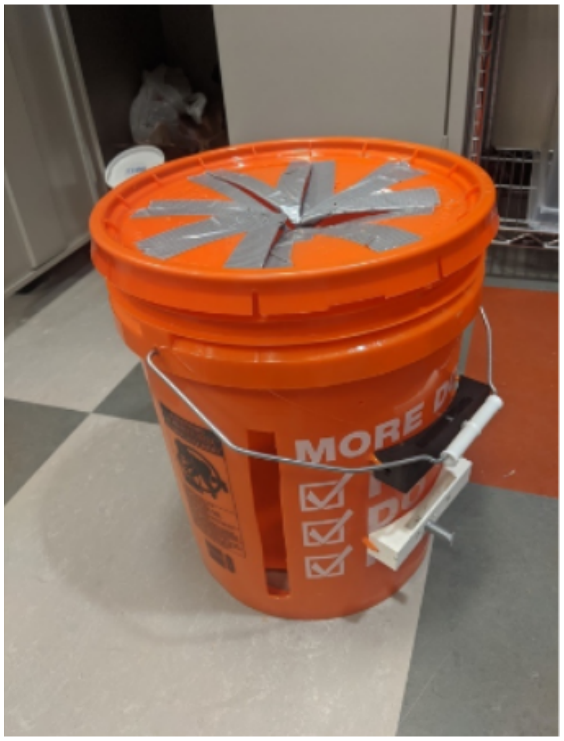


Figure 8: 12 Gallon Bucket DIY Zookeeper

Another design Phase III prototyped was a catch bag with a 12-gallon bucket frame. This design provided little to no drag but was far less robust than the bucket design. In addition, it had the risk of catching on obstacles, interfering with the propellers and camera on the robot, and allowing the spines of the lionfish to potentially break through, posing a danger to handlers. However, they state in their report that this may be remedied by using thicker mesh or chicken wire.

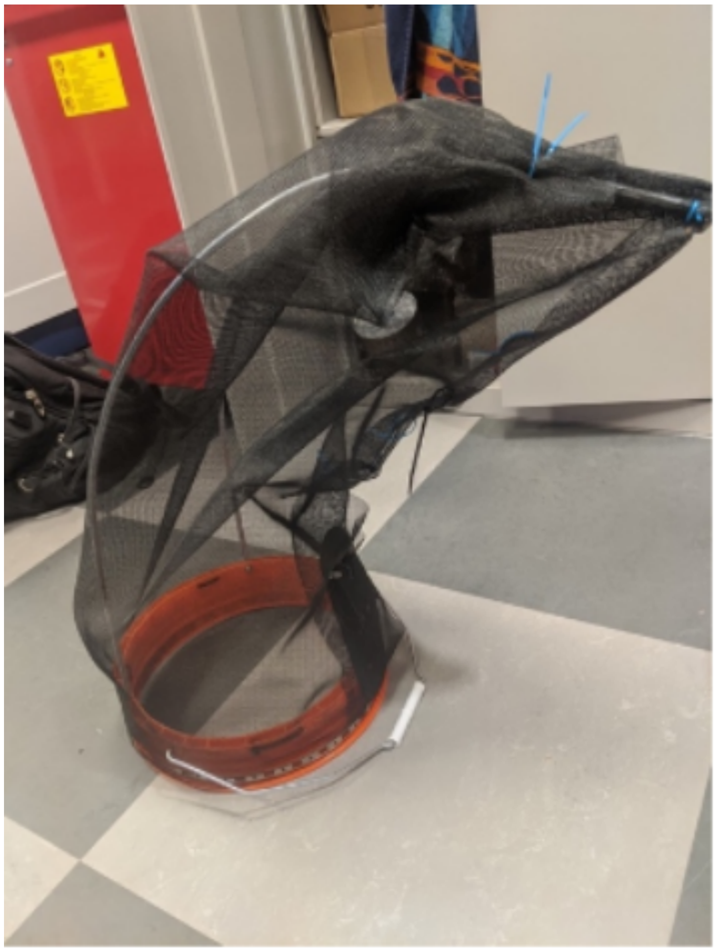


Figure 9: Catch Bag with Bucket Frame

The next prototypes combined the two previous prototypes and utilized a trash can, mesh, and a Tupperware lid to create a containment unit. The trash cans had various holes cut in them to reduce drag that were then covered in mesh, and the Tupperware lid was used as a one-way valve. This proved much more reliable than the 5-gallon bucket lid but had a large surface area and provided a bit more drag. They also found that the mesh at the bottom of the trash can was not very durable.

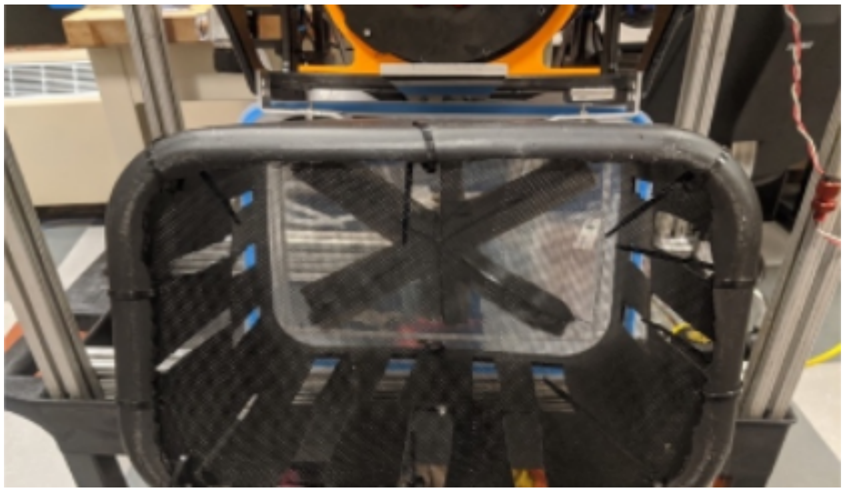


Figure 10: Trash Can with Slits Cut Out

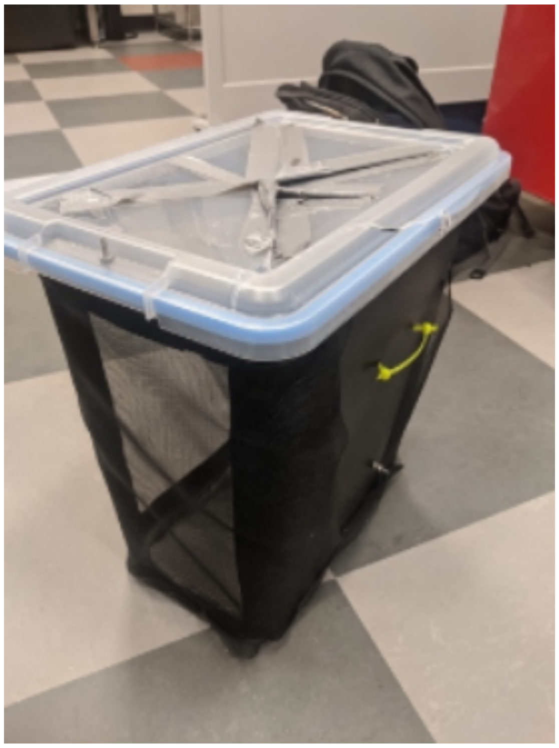


Figure 11: Trash Can with Full Window Cutouts

Phase IV designed a custom cage in Solidworks that would contain the lionfish with minimal drag on the robot. Their container had a pneumatically actuated hinge at the bottom to insert the lionfish in. However, due to COVID, the design was never prototyped. They did, however, perform flow analyses, as shown in Figure 13 (Gangaramane et. al., 2021).

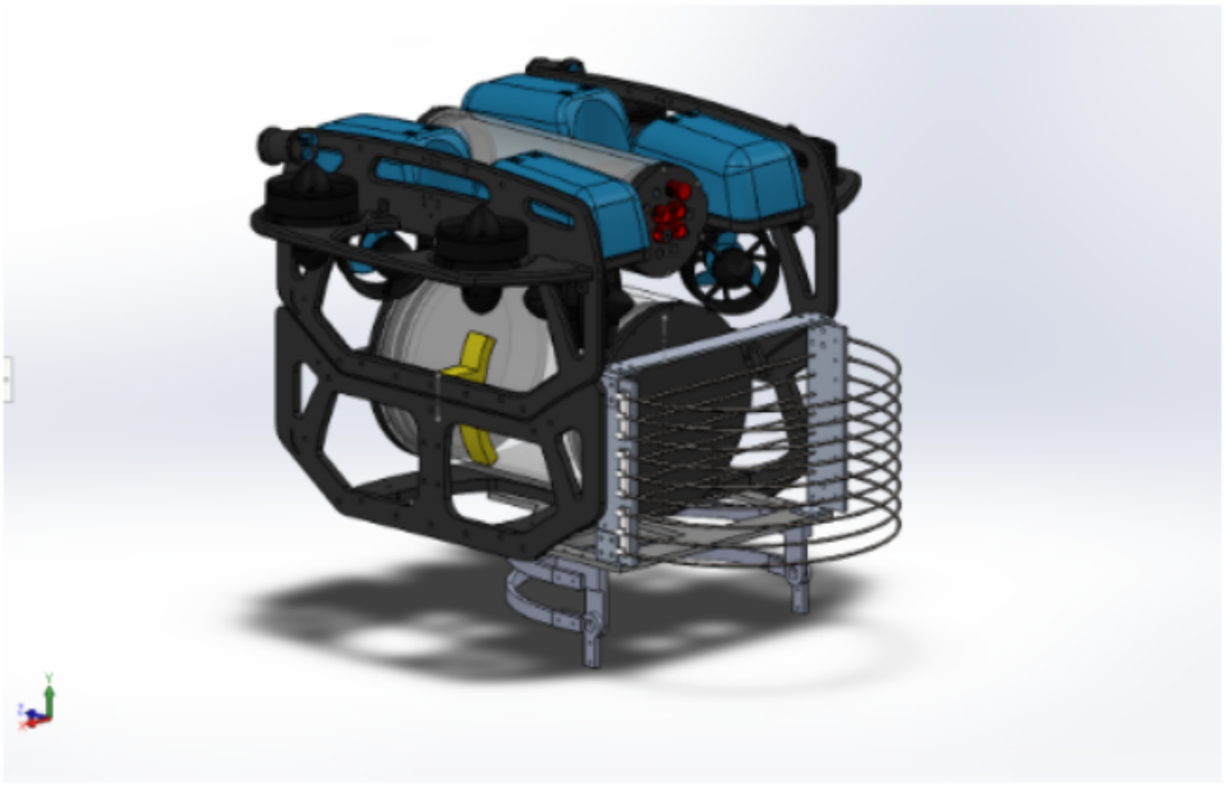


Figure 12: Isometric Rear View of Robot with Phase IV Container Design.

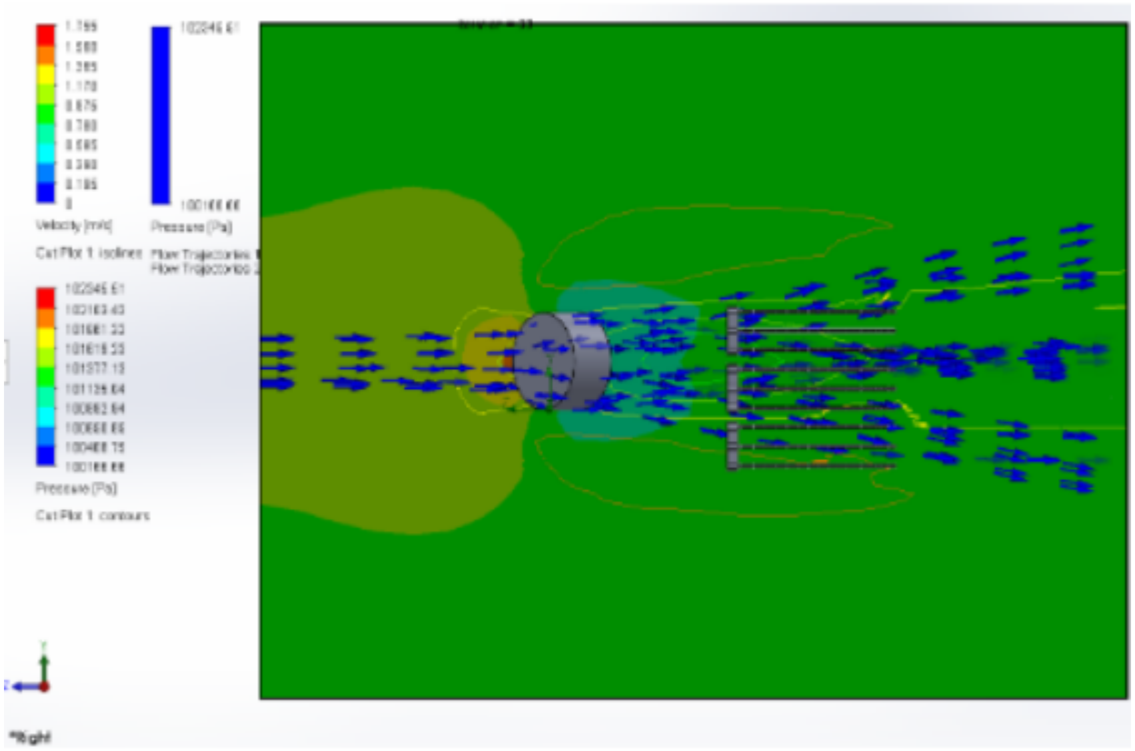


Figure 13: Flow of rear thrusters with empty tall cage. Cutaway at single thruster

2.5.3 Our approach

We discovered that most spearfishing spears use surgical tubing as a spring to propel the shaft forward and spear the fish. Phase III had initially decided this would be too mechanically complex to reset, but we determined that with some modifications a mechanical spring would be less complicated overall than a pneumatic system like Phase III's designs. We took inspiration from a Hawaiian sling style spear design on Amazon that used a spring attached to a handle with a trigger that would release the spear. A design like this could be automated with a winch and rigid or actuated mounting system easily, so we began designing prototypes for such a mechanism.

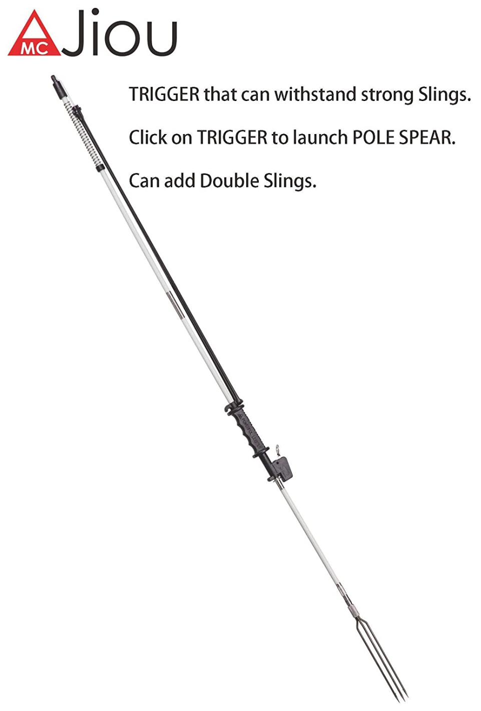

AJiou
MC
TRIGGER that can withstand strong Slings.
Click on TRIGGER to launch POLE SPEAR.
Can add Double Slings.

Figure 14: A-jiou Fishing Pole Spear (A-jiou Fishing Pole Spear Trigger Glass Fiber 5.5' Travel 3 Pieces Hawaiian Sling with 3 Prong Tip and Bag)

Another spear design we considered was a spring-loaded spear gun like the SEAC Polpone Sling Speargun, but we determined that this would be difficult to automate because the spear is shot out from the gun rather than being held like many traditional spears. A reel could be attached but resetting the speargun would still be very difficult to automate, so we decided not to use a speargun of this type.

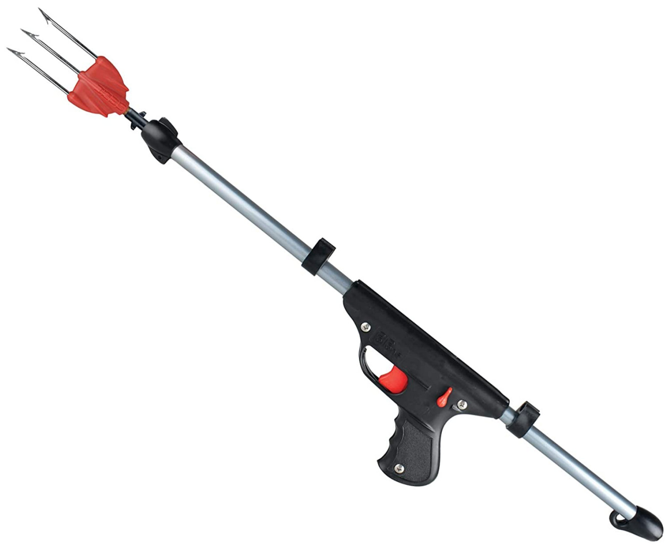


Figure 15: SEAC Polpone Sling Speargun

2.6 Autonomous Navigation

Autonomous navigation is the ability for a vehicle to determine its current location and then chart a course to a destination. Navigational autonomy spans a wide spectrum, from vehicles that only assist their human pilots to robots that require no human input at all. Robots that achieve full navigational autonomy do so through an artificial intelligence trained on either an optimal or a heuristic approach.

AI that takes the optimal approach will chart many paths to the goal and then select the best one by maximizing the advantages and minimizing the disadvantages of each path. People see this sort of AI in their everyday life in the form of services like Google Maps, which finds the optimal path from one location to another. The massive downside of robots using the optimal approach, is that the robot then requires access to pre-existing navigational data (i.e., the maps in Google Maps), as well as an accurate sense of where they are (i.e., a smartphone's GPS connection).

For many robotic applications, the optimal AI approach is unfeasible; robots could be operating without a GPS connection, or in an area with no pre-existing map. Hopefully, your Roomba can't go online and find a detailed map of your home's interior. So instead, the Roomba has to bounce around your living room and figure out for itself where your couch, coffee table and TV stand are. Other robots, such as one designed to solve a maze, follow simple rules, like follow the wall on the left. A lawn mowing robot would follow a simple zig zag motion pattern across the lawn, only deviating to avoid obstacles. Like these examples, our lionfish harvester is flying blind: it can't connect to the GPS satellites from 100 meters below sea level, and not every square inch of Atlantic Ocean reefs has been mapped.

Therefore, our project will need to use the heuristic approach to autonomous navigation, searching for lionfish by following a search pattern and simple rules. The system designed for our robot will need to be tailored to the goals of this project; we cannot have our robot bouncing into the reef like a Roomba. This is where our computer stereo vision and environmental sensors will be used: detecting obstacles so that we can navigate around them. Computer vision will also be responsible for detecting lionfish for harvesting, once it does, we will need to engage a secondary heuristic navigation AI. This AI will need to follow rules rather than patterns in order to position itself into an appropriate position for harvesting. Determining the appropriate rules and patterns for each of these AI will be the primary challenge of enabling the robot to navigate autonomously, underwater, and in three-dimensional space with six directions of movement

3 Methodology

3.1 Electronics Redesign

One of the first challenges we faced was removing and taking stock of the electronics in the robot. We had to pull the flanged seal out of the bulkhead to remove the electronics, which was difficult and a bit dangerous. The electronics were also mostly secured by zip ties, duct tape, and Velcro, and some of the boards were not currently being used. In addition, we wanted to make some changes to a few of the custom boards the robot was currently using and switch out the magnetic switch for a manual bulkhead switch. Because of all these factors, we decided to redesign the electronics layout and make it easier to access and remove. Initially we came up with two possible designs, one that would consist of two levels of removable drawers and one that would have a removable core.

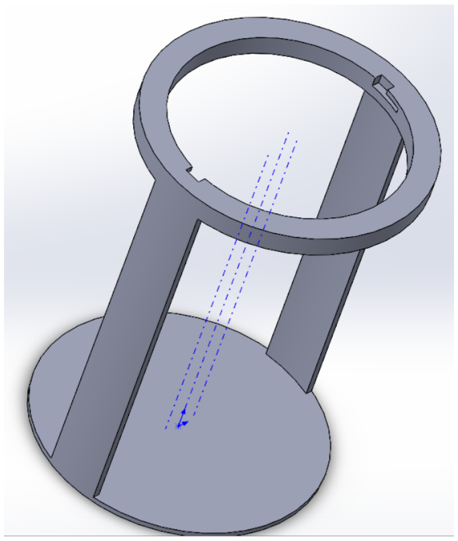


Figure 16: Base of the removable core concept

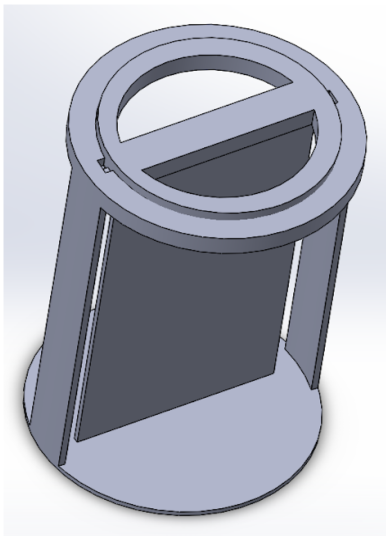


Figure 17: Full assembly of the removable core concept

Ultimately, we decided to use the design with the removable drawers, as it would be easier to manufacture and would have more usable space. Early versions of the design consisted of a large, one-piece shell that the two drawers would slide into.

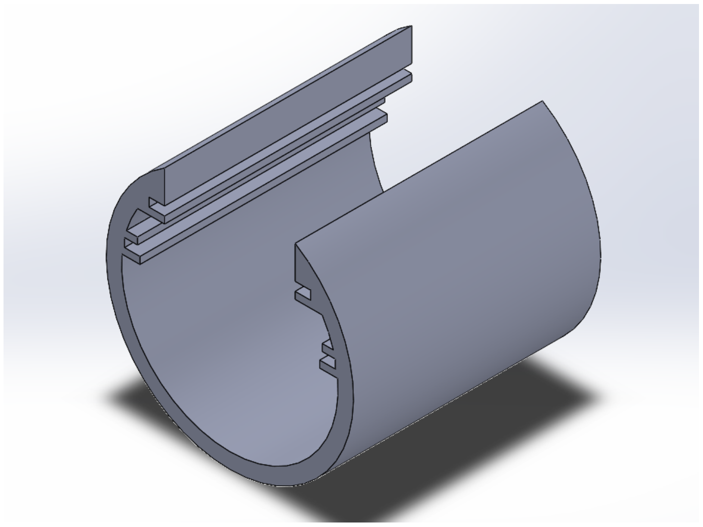


Figure 18: Original Drawer Shell Design

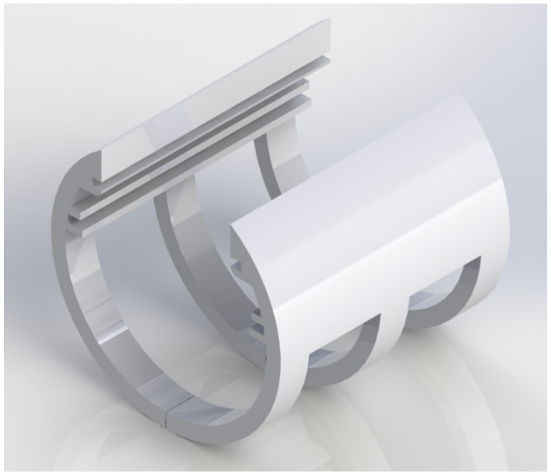


Figure 19: Shell Design Split in Half with Added Cutouts

However, we determined it would be easier to both install and manufacture if we split the shell in half and lightened it by cutting out sections of the bottom, as shown in figures 18 and 19.

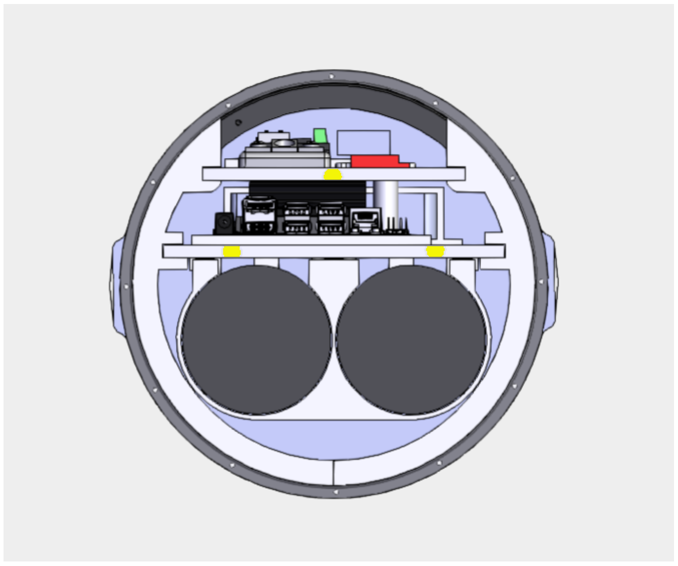


Figure 20: Early Version of Drawer Redesign

Once the electronics had been preliminarily laid out, we altered the drawer design once again to add extra support to the slots that held the electronics boards. This was to ensure that the structure would be strong enough to hold the weight of all the electronics and batteries without the risk of breaking. This was originally omitted due to concerns that the added support material would take up too much space from the electronics. However, once we determined roughly where all the elements would fit, we realized adding it in would not be an issue.

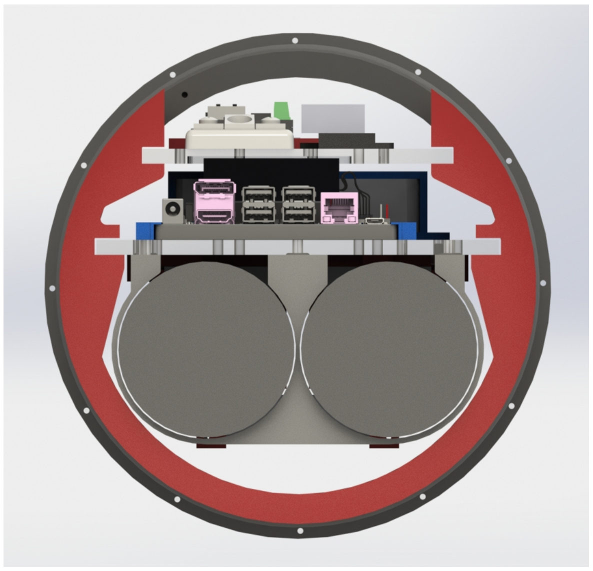


Figure 21: Updated Drawer with Added Supports

Once we had finalized the shell design, we finalized the mounting holes and mount designs for all the electronics. The batteries, relays, and circuit for the mechanical switch were mounted beneath the bottom drawer, while the Xavier and the network switch were mounted above the bottom drawer. The five-volt regulator, the tether interface, and the power distribution bars were mounted on the top drawer. We tried to keep any connections as close as possible, but inevitably due to the restricted space some of the connections needed to be remade with longer wires. Regardless, we ultimately decided to rewire much of the electronics to add color coding and make the wiring easier to follow.

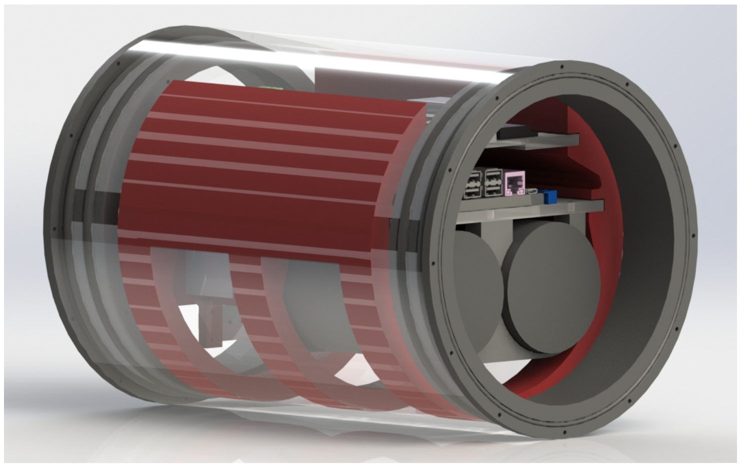


Figure 22: Isometric View of Final Drawer Design

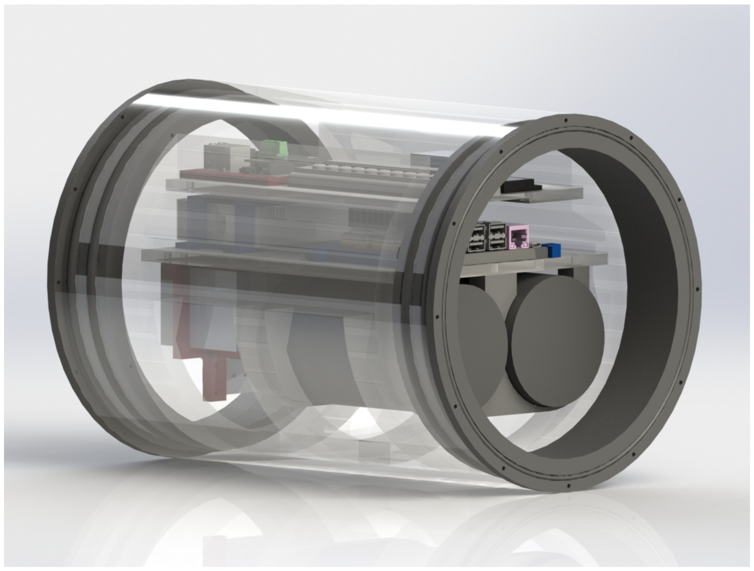


Figure 23: Isometric View of Electronics with Transparent Shell

3.2 Spear & Containment

3.2.1 Spear Mechanism

We knew from the start the spear mechanism was going to be a big design challenge, so we started brainstorming ideas early on. We looked for commonly used spearing devices online, as well as the ideas previous MQP groups explored.

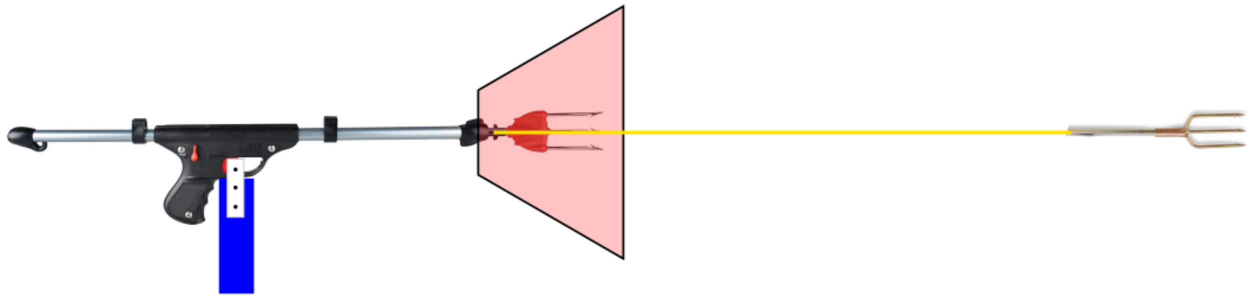


Figure 24: Early Spearing Mechanism Concept

One of the first designs we explored used a SEAC Polpone spring-loaded spear gun to fire a spear that could then be reloaded and re-used. However, we determined that creating a mechanism to reload the spear would prove extremely difficult and would likely require a lot of mechanical complexity. In addition, a real based speargun had potential to damage the reefs. Because of this, we began looking at pole spears and Hawaiian slings for inspiration.

Figure 25: Pole Spear (Lionfish pole spear with paralyzer tip)

Many of these types of spears had rubber surgical tubing attached to the end that a spear fisher would wrap around their wrist and then stretch by grabbing the shaft of the pole as far up as possible, then use the force of the rubber tubing to spear the lionfish on release. We had some concerns about the rubber bands getting in the way of the reloading and firing mechanisms or being difficult to reload, so we decided to use a constant force spring instead, as some of the team members had used them in robotics applications successfully before. For the initial prototype, we bought a constant force spring with a spring force of just under 29 lbs. This was because the research we found generally placed the force of the rubber bands used on spearguns at approximately 40 lbs., but the force of the rubber bands used by the Phase 1 team for their firing mechanism was only about 12 lbs. In addition, the constant force spring of the correct size was significantly less expensive at this strength than the ones of the same size with a higher force. Given these factors, we decided the 29 lb. constant force spring would be a good choice to use for the prototype, and we could purchase a stronger or weaker spring for later versions if it proved too weak or too strong for the job. Below is a render of the initial Spear Prototype, made using 3D printed parts, a constant force spring, and two linear bearings from McMaster Carr, along with a spear shaft purchased from Amazon.

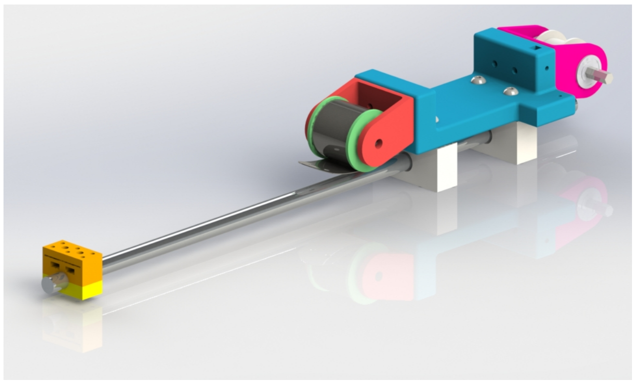


Figure 26: Spear Mechanism Preliminary CAD Prototype

Once the spear mechanism concept had been proven, it was time to create a reloading and firing system. There was much deliberation on the best way to go about doing this. The primary options were to use a winch or ratchet that could be released to allow the spear to fire, or to use braked gearbox. The winch seemed like it may be simpler, and it would allow the load to be taken off the motor when the spear was cocked. However, we realized that it would only allow the shaft to spin in the opposite direction at whatever speed the motor and gearbox would allow it to go, rather than being able to spin freely. This would not be fast enough to spear a lionfish, as we knew the gearbox ratio would have to be significant for the motor to have the power to pull back the spear. The braked gearbox, while more complicated, had the potential to allow for both a locking mechanism for the shaft to take the load off the motor and the ability to shift into a free-spinning mode for firing. However, this would require two shifters, which would be difficult to implement, especially without the use of pneumatics, which we had determined we would prefer to avoid. At first, we thought combining the two ideas might be feasible, using a ratchet to prevent back drive of the gearbox and a shifter to allow free spin. However, we realized that the spear mechanism would likely be required to have the ability to be disarmed without firing, which meant we would need the ratcheting mechanism to be automatically reversible, further complicating the design. Because of this, we ultimately decided to go with a double-shifting gearbox with one locking stage and one free-spinning stage. However, because we still wanted to avoid the use of pneumatics, we decided the best course of action would be to create a linkage using a servo that would replace the traditional pneumatic.

3.2.2 Containment Mechanism

While our primary mechanical focus this year was on designing and manufacturing the spear mechanism, we wanted to make sure it would be able to be integrated with a containment system. We decided to aim to give the spear mechanism a small footprint so future MQPs would have the ability to attach an aiming or actuation mechanism onto it if they chose to create a containment system that would require such a mechanism, but we came up with a preliminary design that future MQPs could refine, manufacture, and test if they so choose.

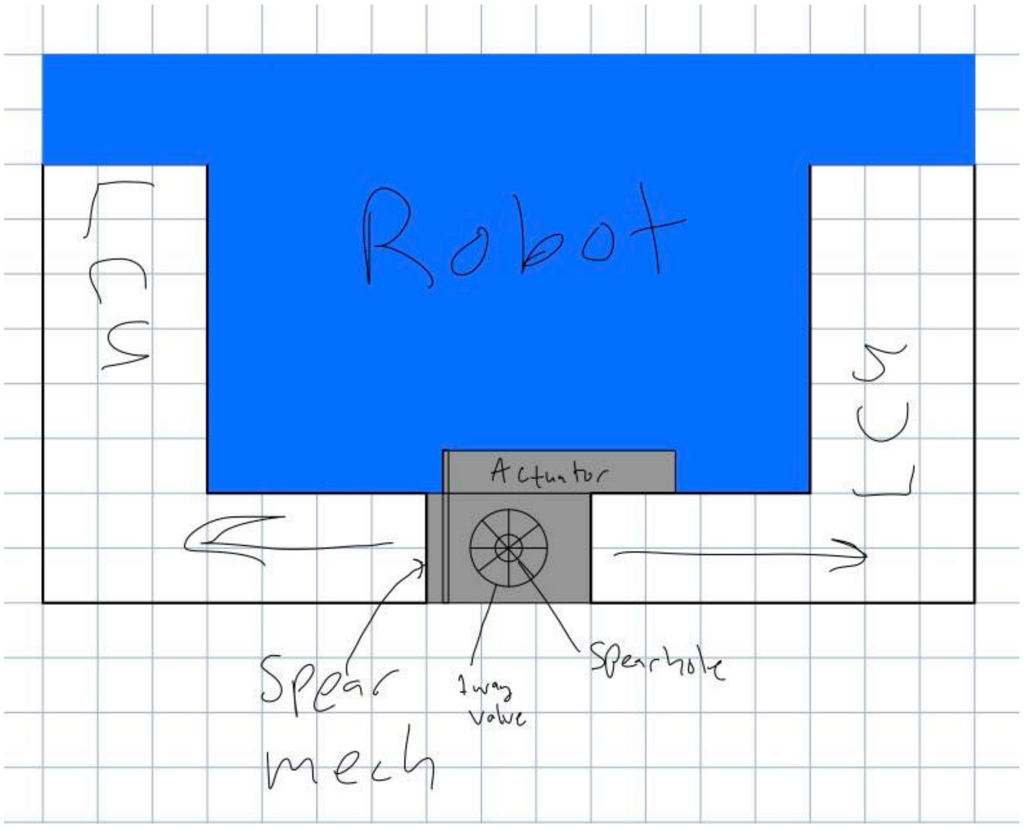


Figure 27: Front View Sketch of Containment Unit Design

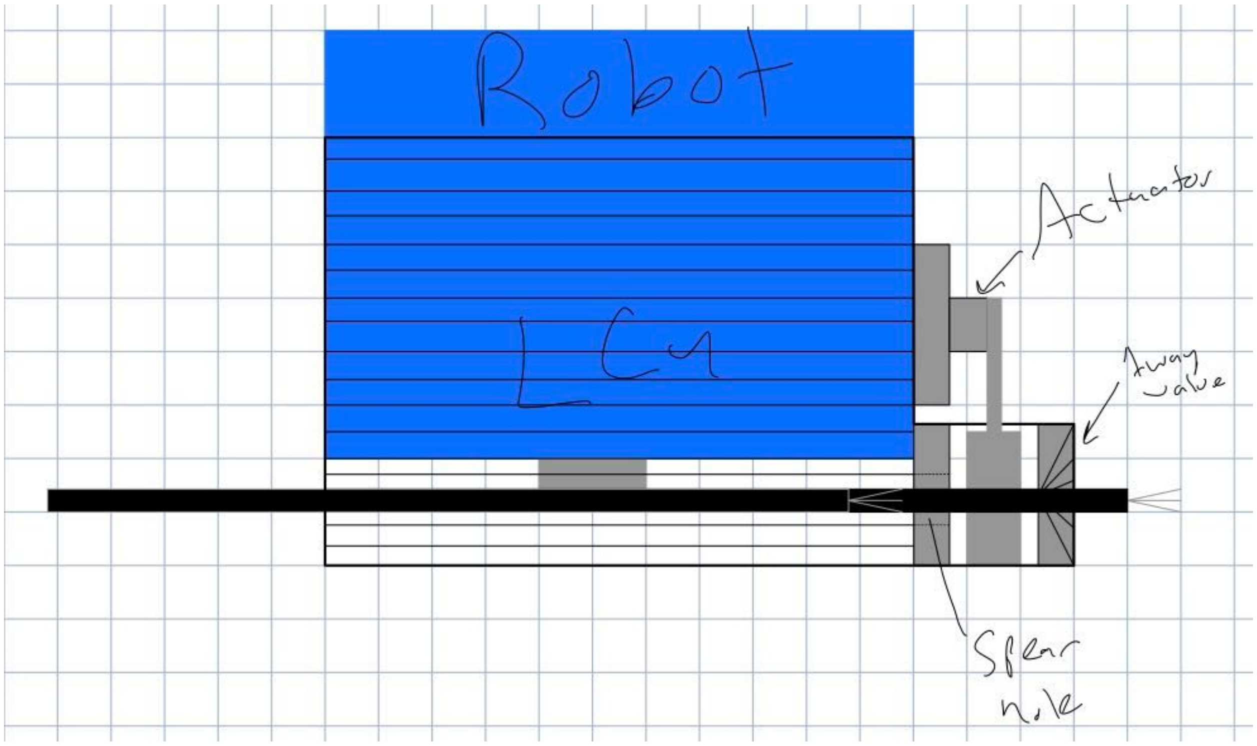


Figure 28: Side View Sketch of Containment Unit Design

Figures 27 and 28 show the preliminary sketch drawings created to represent the containment unit design. The design would utilize two of the LCUs designed by the Phase IV Lionfish MQP, but instead of being attached on the back, they would be mounted sideways and attached on the side. The Spear mechanism would then be mounted underneath the center of the robot and would normally remain cocked just in front of a one-way Zookeeper™ style valve. Then, when it speared a lionfish, it would retract into a rectangular tube at the front of the robot, pulling the lionfish through the one-way valve, and further retracting into a hole that was just big enough for the spearhead to get through so the lionfish would be pulled off. Then an actuator would push the lionfish off to the side, into one of the LCUs on either side of the robot. Each time a lionfish was speared, the side it is pushed to would be changed in order to keep the robot balanced. Our initial thoughts were that a linear actuator of some kind would push the fish along into the LCU, but the advisors suggested the use of a propeller on either side to push or pull the fish into the LCU on either side. This is something we suggest future groups explore, as it did not fit within the scope of our project timeline this year.

3.3 Communication

3.3.1 Internal Communications

There are three computing resources within the robot: a Pixhawk 4 and a Raspberry Pi 3B in the upper pressure chamber and the Nvidia Jetson Xavier in the bottom one. The Pixhawk 4 is a computer specifically built as an autopilot, running a real-time operating system (Apache NuttX OS) and designed with many ports for motors, sensors and other peripherals. Although the Pixhawk is well-designed for piloting the robot, it can only communicate with a single companion computer to handle telemetry. Enter the Raspberry Pi, which runs a program called ArduSub to act as the Pixhawk's companion computer over USB. ArduSub can handle certain piloting maneuvers, such as depth holding, and provides an interface for remote operation of the Pixhawk and anything connected to it.

The last computer in the robot, the Nvidia Xavier, handles the intense computations for object detection, stereovision, and autonomous navigation; it essentially replaces what would be a human pilot. Although we've installed the Xavier inside of the robot, it is still "remote" operation, and so it controls the robot by communicating with the Raspberry Pi. ArduSub connects to remote operating systems over MAVLink (Micro Air-Vehicle Link), a protocol designed for communicating with drones and other small aerial vehicles. Since our code is written in python, we use the pymavlink library, a python implementation of MAVLink. This allows the Xavier to send control signals and receive sensor data over an ethernet cable connected to the RaspberryPi. Using pymavlink to communicate with ArduSub is preferential than attempting to place the entirety of the robot under the Xavier's direct control. This is because we would rather use the well-developed code of ArduSub and NuttX OS to manage the robot's physical systems rather than relying on our own Python scripts.

3.3.2 External Communications

As mentioned before, ArduSub is designed for remote operation, and that's typically not done by a second computer as it is in our robot. Because we have kept ArduSub in our design, we are still able to perform remote human operation. This is done over Fathom ROV Tether cable, a waterproof Cat5 ethernet cable sold by Blue Robotics. However, this is not a normal ethernet connection, instead the tether relies on the HomePlug AV (IEEE-1901) standard for sending Ethernet through power lines. Because this standard uses only two wires, it requires an interface board on either end to convert it to and from standard ethernet connections. Inside of the robot, one of these interface boards connects the tether into ethernet and connects it to a network switch which also connects to both the Raspberry Pi and the Nvidia Xavier. On the topside, the tether is connected to an interface box, which uses the same board as inside the robot, but this time converts the ethernet connection further into a USB connection for use with a laptop. This laptop runs an open-source application called QGroundControl, which communicates with ArduSub to allow users to pilot the vehicle. This piloting is done with a gamepad controller, while heading, sensor data, and even live video feed are displayed on QGroundControl. As much fun as it is to drive the robot around in the water, there is another reason for keeping QGroundControl functional, it provides a powerful interface for tuning and calibrating the robot's systems, things that would be infeasible to do with code alone.

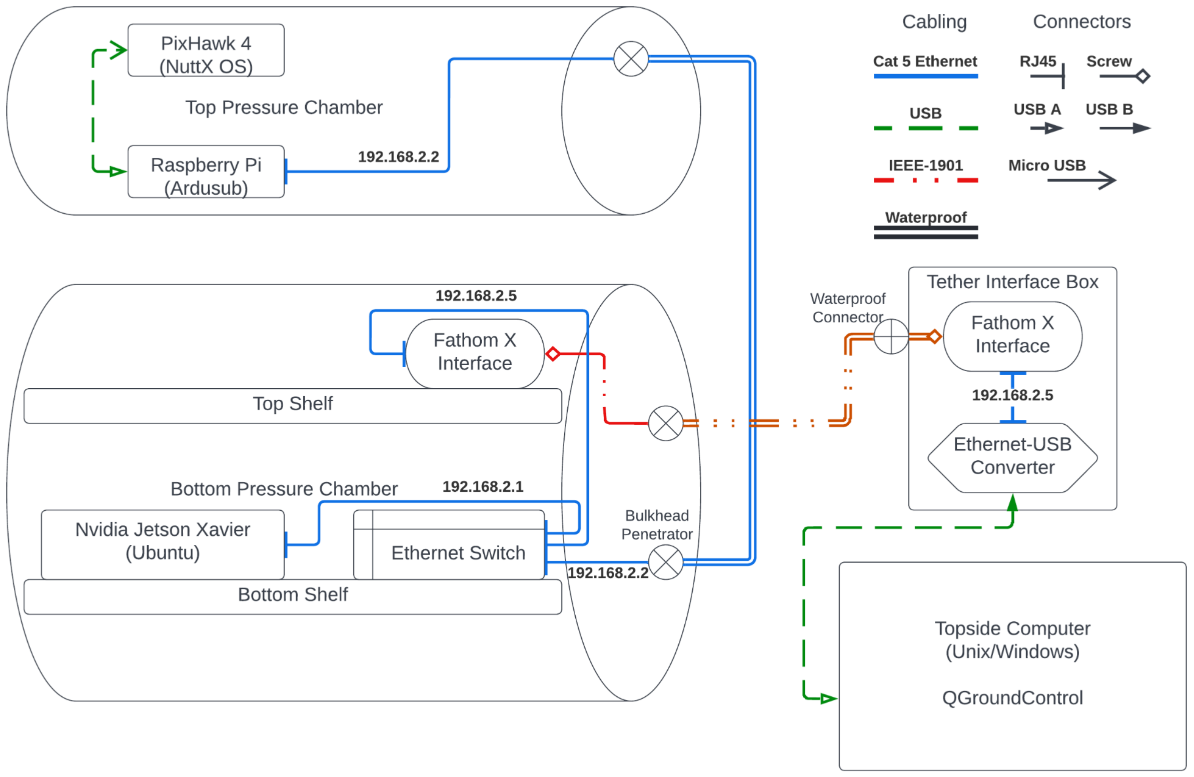


Figure 29: Full diagram of robot communications, internal locations are approximately accurate

3.4 Operational Considerations

Every emergency is currently handled the same way by the robot: surface. If leaks are detected by the leak sensors, or if a sensor dies, the robot will surface. This isn't part of our design; this is a built-in feature of ArduPilot. Using QGroundControl, a user can define responses to emergencies such as leak detection. These responses range all the way from warning the human pilot to automatically entering surface mode, which tells the autopilot to ascend as rapidly as possible. The issue is that the Pixhawk has very limited knowledge about the bottom pressure chamber, there are no connected sensors there. Our solution is to connect the bottom chamber's leak sensors to the Nvidia Xavier, which can then inform the autopilot if there's a leak. While this works well for leaks, it does not help if we lose communications or the Nvidia loses power. So instead of using the leak emergency in ArduSub, we can take advantage of a different emergency: topside communication lost. In typical usage, QGroundControl is constantly sending "heartbeat" signals once per second; if ArduSub goes too long without receiving one of these signals, an emergency trigger. Usually, this response is just a warning message, but

QGroundControl lets us reconfigure ArduPilot so that the robot will surface once topside communication is lost. Now in our design, the Nvidia Xavier is going to be the computer sending these heartbeat signals. This way if there's any problem in the bottom pressure chamber, even when communication or the Xavier is knocked out, the robot will still engage surface mode. Even if all else fails, and the robot is left completely dead in the water, we have designed the robot to have positive buoyancy so that it will always float itself to the ocean's surface.

3.5 Computer Vision and Object Detection

Computer vision is a crucial asset to detect and harvest lionfish. To implement computer vision, we utilized the webcam - present in the top chamber - as the webcam is attached to a servo, which further increases the range of the camera feed, and outputs video at 1080p resolution. This increases the chances of our machine learning (ML) model to detect the lionfish underwater. We trained our model to detect both lionfish and divers to restrain any misfires by the robot. When the model detects the object in the frame, it constructs a bounding box in the shape of a rectangle around the object and classifies it as either person or lionfish. If the object is a lionfish, the robot calculates the centroid of the bounding box by intersecting the diagonals of the rectangle. This ensures that the robot always hits the target and there are fewer wasted shots.

The previous MQP team used COCO - Common Objects in Context - (a pre-trained ML model) to detect objects in everyday life. This model could identify objects like TV, person, bicycle, car, dog, eyeglasses, etc. However, upon further testing, we realized that the model would consume a lot of computing power to identify the objects, thus resulting in low frame rates from the camera feed. As the program needs to detect only a person and lionfish (and ignore other life forms), we created our own ML model and trained it to detect a person and a lionfish. Furthermore, the previous iteration did not show the detection accuracy of the model. It was imperative to know how confident the model is regarding the identified object to minimize any potential misfires. We hence created a modular object detection program that would train the ML model only once on the desired objects and then use the trained model to detect the objects with the help of the camera. We did a preliminary test of the object detection program by running it on the pre-trained COCO model (Figure 30 and 31) and observed no significant frame drops.

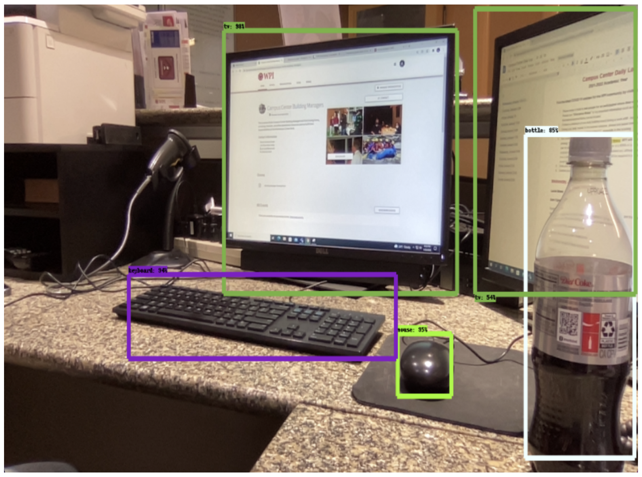


Figure 30: Preliminary test of object detection with pre-trained COCO model

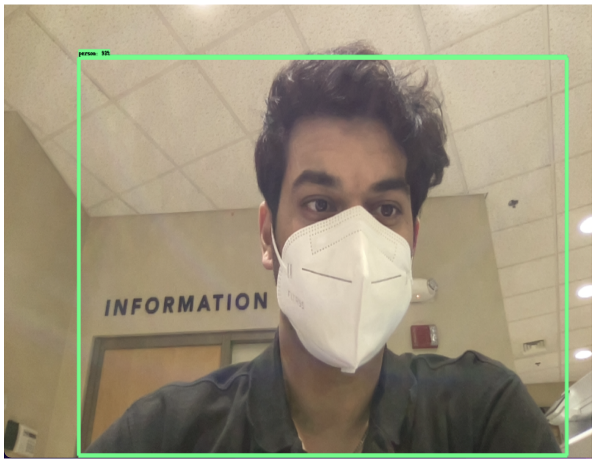


Figure 31: COCO model detecting person

Since we would be training our own ML model to detect Lionfish and divers underwater, the important distinction factor between the diver and the Lionfish would be the shape. We would need to have a higher detection rate for detecting a diver than a Lionfish. As the diver would be further away from the robot than the Lionfish, we could potentially lose a lot of details in detecting the diver. Thus, we would train our model to focus on the features like diver equipment (fins, scuba regulator and tank) and the inherent shape of human beings.

We then created the training model such that one can train it on any object if enough training data is provided (in the form of videos or images). As the model could be trained on any machine, and later be imported to the robot, we chose a computer that has high computing and graphical power to run the model and produce the necessary data in a small amount of time. Thus, we relied on Apple's neural engine present in the M1 MacBook Pro to accelerate the machine learning model training. We further utilized TensorFlow to create our ML model as it was easy to integrate with OpenCV and python. However, the Object Detection library used for detecting and classifying the objects was compatible with TensorFlow1, and MacOS Big Sur and Monterey only supported TensorFlow2. Moreover, to efficiently utilize Apple's neural engine, TensorFlow updated their package to include "tensorflow-macos" and "tensorflow-metal" that supports the ARM64 architecture. But none of the extended packages had support for the object detection library and thus were not useful to us.

To tackle these hurdles, the team utilized Anaconda and created a virtual environment with python 3.5. We were then successfully able to download TensorFlow1 version 1.15 (which is the highest version available of TensorFlow1) and other libraries essential for computer vision. It is to be noted that this was done only to train the machine learning model. The future teams do not need to run the machine learning program for the robot to work. This program is loaded separately than the Stereovision so that it does not get run repeatedly and use up the available space in RAM. If the project asks for the robot to be trained on objects other than people and Lionfish, it is advised to run this program on a Windows operating system, with python 3.6 or lower. It is also advised to use the LabelImg software as it easily makes and labels the bounding boxes of the images used for training and testing the model. The model would create the bounding boxes around the objects that it identified and would give a percentage in front of the identified object to show how confident the model is at identifying the object.

3.6 Movement and Control

3.6.1 ROV Control

The ROV uses three modes of operational control: manual, semi-autonomous and fully autonomous. Manual control is done by connecting a laptop up to the tether and activating manual mode via QGroundControl to control the robot via Xbox or PlayStation controller. The second way is by connecting the laptop to the ROV and controlling it through typing commands into the terminal in a semi-autonomous manner. The last way the ROV is controlled is fully autonomous by just having it operate solely through code. Full autonomous operation can be broken down into two movement command and control structures Joystick emulation and Vector control. Joystick emulation is based on RC channel commands imitating manual control using joysticks; this method works but is susceptible to changes in orientation and control scheme via QGroundControl. Vector control works by giving the ROV an axis (x, y, z, r) to travel upon which bypasses the RC channel control structure eliminating the need to imitate a controller. The vector control movement commands/methods are named by the axis they control, so the method only needs to be given a power level and duration to move along the desired axis.

3.6.2 Code Control

The first step to creating a fully autonomous robot was creating the classes and functions necessary for autonomous operation. We created the Robot folder to contain the codeController, controller, lionfishRobot, robotUtils, navigationCLI, sensorData, dryTest, and poolTest classes. The controller class is used during terminal control where it controls the robot based upon the commands typed in the terminal. The codeController class handles movement commands just for autonomous code control. When an instance of the class is created the code initializes all motors to power level zero. The movement commands are separated into two different command and control structures: joystick emulation and vector control. Before using any of the movement functions the runOnce function has to be called/run to act as translator between the Robot class and the CodeController class, so that the Robot class variable names align with the CodeController class variable names. It also sets all thruster motors to power level zero (off) stopping all motion if the robot was under power. The first two movement functions constantTranslation and timeTranslation operate under the joystick emulation command and control structure. The constantTranslation function takes in two key values which will control direction and power of the robot’s movement. The direction is handled by a simple string input: turn, up, down, forward, and backward which will determine the direction. The power value is also a float variable representing a percentage where 0 is off, 100 is full power in one direction and -100 is full power in the opposite direction. This function sets a direction and power that the robot will follow until told otherwise. This is great for continuous movement based on sensor data. The timeTranslation function takes in three key values which will control the direction, duration and power of the robot’s movement. The direction and power values are the same as the constantTranslation’s values. The only difference is the addition of the duration value to give a terminating condition. The duration is the length of time that the motors will run in seconds; it is a float variable. The timeTranslation function is very helpful to fine tune and test movement control and to create simple search patterns. The runTimed function is under the joystick emulation command and control structure, and it takes in a power level as a float and a duration also as a float. It uses the power and duration for three consecutive movement commands to drive forward, dive and lastly turn. The runTimed function is for dry and pool testing, but primarily for pool testing because of the possible length of time the motors could run with a max time duration given by an operator. The next group of three functions are the runTimedRCForward, runTimedRCDive, and runTimedRCTurn functions. They all take in the same values as the runTimed function power and duration both float and are under the joystick emulation command and control structure. The runTimedRCForward function is pre-set to only send the values for the RC channel FORWARD which locks the robot to going either forward or backward. The runTimedRCDive function is pre-set to only send the values for the RC channel THROTTLE which locks the robot to the motion of either diving or ascending. The runTimedRCTurn function is pre-set to only send the values for the RC channel YAW which locks the robot to turning either right or left. These three functions were used during dry and pool testing, but they can also be used in normal movement code. The next group of three functions are the runVectorForward, runVectorDive and runVectorTurn functions. They all take in power as a float and duration as an integer and are under the vector command and control structure. The runVectorForward is pre-set to only send values for x-axis motion and limits the robot’s movements to forward and backward motion. The runVectorDive is pre-set to only send values for z-axis motion and limits the robot’s movements to diving and ascending. The runVectorTurn is pre-set to only send values for r-axis motion and limits the robot’s movements to turning right and left. Same as the previous group of three functions, they were used during dry and pool testing, and can be used in normal movement code. The last movement-based function in the codeController class is emergencySurface which will turn off all the motors and allow the robot’s positive buoyancy to raise the robot to the surface in the case of an emergency.

3.6.3 Testing Movement Control

Movement control verification and testing happened on land first in dry testing and after successful dry tests occurred pool testing would commence. When dry testing started there were

three objectives. To establish reliable commands for the forward/backward, turning and diving actions. To have variable control of the power of the thrusters via percentage and to be able to control the start and stop of the motors based on time. During the initial part of the dry testing code, it would test communication, armed status, and initialization of the robot code.
The first test would be on the file structure of the code by initializing the two key classes Robot and CodeController. During the test the robot would print out a before and after message to let the operator know when and where a test would fail. After the file structure passes it will run the armRobot method to try to arm itself and if it is successful, it will double check by running the check_Armed method. After arming the robot, the robot will test code initialization by running the runOnce method in the CodeController class. It aligns all the Robot class variable names to the CodeController variable names while also setting all motors to power level of zero to stop all motion if the robot is under power. If runOnce passes then a series of statements will be printed to inform the operator what will be tested by stating what directions, power levels and time durations are being used along with any general information on the test or series of tests. In addition, the general information statements at the beginning before each movement control test a quick statement will be printed to let the operator know what the power level and direction are before starting. After running through all the movement tests the robot will disarm itself as a safety precaution.

4 Results

4.1 Electronics Redesign

4.1.1 Internal Electronics

When we first started working with the robot, the two years of incremental previous work had taken their toll on the internal electronics, and a total overhaul was needed.

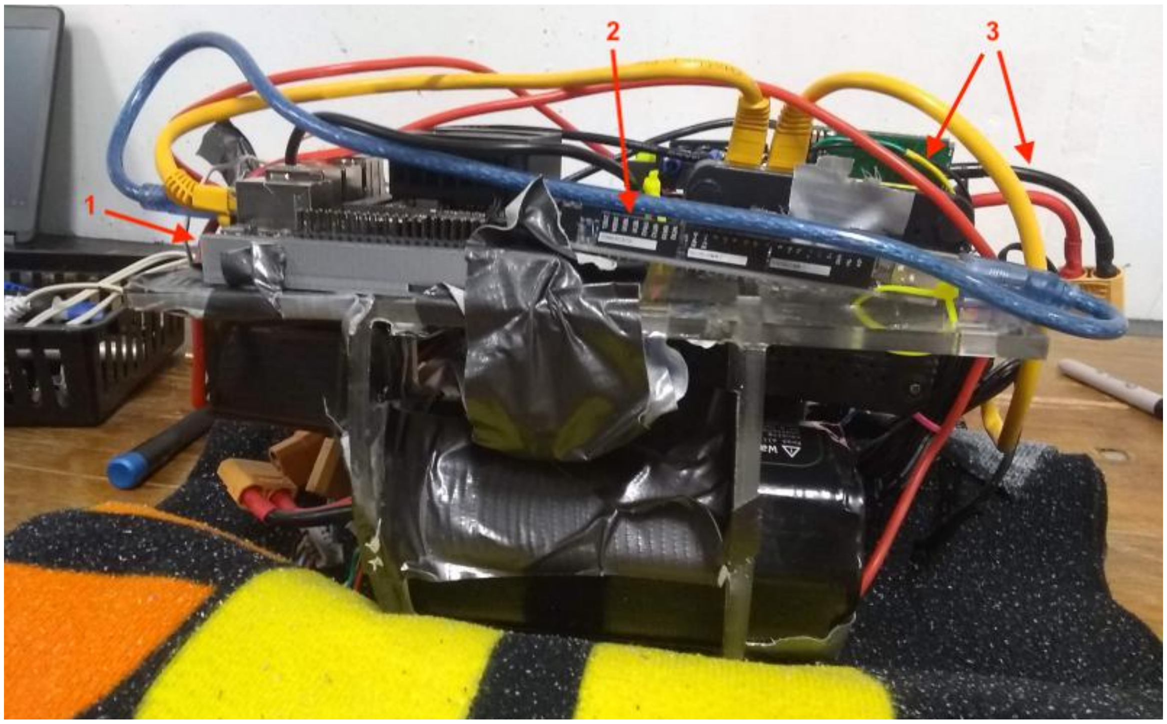


Figure 32: Side view of robot electronics when we started, with issues labeled

Figure 32 shows the robot's internals as we first saw them, demonstrating exactly why so much work was dedicated to correcting this. While examining the structure more closely, it became evident that a lot of these electronics were either unused or half disassembled. And while the polycarbonate shelf had mounting holes, very few of them seemed to line up with the components. After reading through the previous teams' reports and weekly meeting agendas we managed to piece together an explanation for the state of the robot, for brevity we will only cover three of them here according to the labels in Figure 32:

1. The Nvidia Xavier is duct-taped in place. This is because the shelf was designed to mount the Nvidia Nano, but when Phase IV upgraded to the Xavier the mount no longer fit
2. This Arduino Mega is completely unused. While Phase III had code on here related to the EchoSounder sensors, we saw no evidence of code on the Xavier that received this data as was described.
3. These wires don't go anywhere. The green and yellow ones to the left are cut off pieces of wire from the Phase III team's magnetic reset switch for the Nvidia Nano. The old battery connector to the right simply was not removed after Phase IV installed a power switch.

While it is understandable that after so many revisions any project's assembly quality would suffer, we as a team felt that its initial state was unacceptable going forwards. As discussed in section 3.1, therefore we prioritized usability and ease of retrofitting in our design.

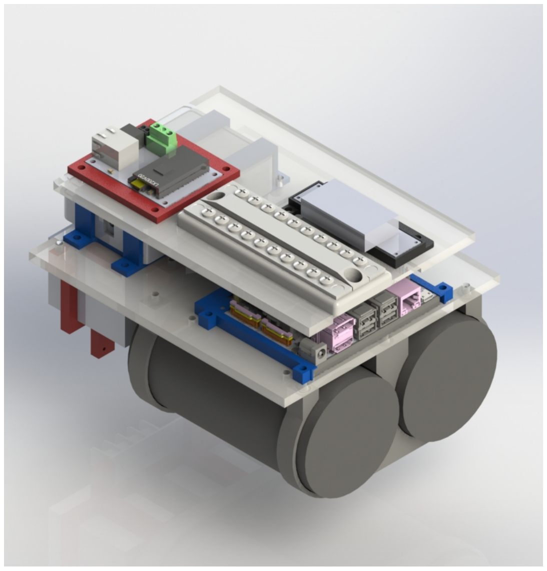


Figure 33: Final Shelf CAD Rendering

We have succeeded in fabricating the shelf design as shown in Figure 33, and it has proven itself as we worked with it in the past few months. The shelves are very simple, Lexan with some holes drilled into them, allowing a future team to rearrange or add components by simply cutting and drilling holes in a new piece of Lexan. Hopefully, the clearer and easier-to-modify layout of the new internal design will encourage future teams to continue using best practices when adding additional components.

Not everything went smoothly, of course. A careful comparison of the CAD versus the final layout will reveal that the 5V regulator on the top shelf is rotated 90º clockwise from its expected position. This is because the clearance between the input pins of the regulator and the power rails next to it made wiring difficult. While at first, we were able to force it into place, this caused mechanical strain which damaged the regulator; a short circuit destroyed it and the ethernet switch after a few weeks of use. After replacing the regulator with a spare and spending $10 on a new ethernet switch, we learned from our mistakes and rotated the regulator to loosen the cable strain. Also note that there is unused space on the top shelf and many spare terminals on the power bars. This was intentional, to provide space for expansion. For example, if a future team decides that a new microcontroller is needed for the Echosounders, this space can be used. Although the extra space cannot fit the old Arduino Mega, a smaller board such as from the Arduino Nano or Adafruit Feather series would certainly fit there. The result of the electronics bay redesign was an overall success, and it is our hope that future teams will benefit from this effort.

4.1.2 Power Switch Replacement

When discussing this project with members of the WPI community, one of the most frequent tongue-in-cheek questions we received was “did you remember to give the killer robot an off switch?” Of course, it has one, but that wasn't always the case. Phase III had wired a momentary reed switch to quickly power cycle the old Nvidia Nano in case the software froze. Phase IV took this idea and upgraded to a latching reed switch to operate relays that could turn the robot on and off properly.

However, in our initial work the magnetic system proved unreliable, often failing to power off the robot or rapidly power cycling the entire robot. Worse still, there were no freewheeling diodes connected to the power relays despite being shown in the circuit diagrams. This means that the circuit was vulnerable to large spikes in voltage from the relay's coil inductance whenever the power was turned off. These power fluctuations were such a concern that we actually scanned all of the storage on all of the computers for data corruption, though thankfully none was found. Initially we looked at improving the existing system, to make it safer and more reliable, but quickly ran into more quality issues.

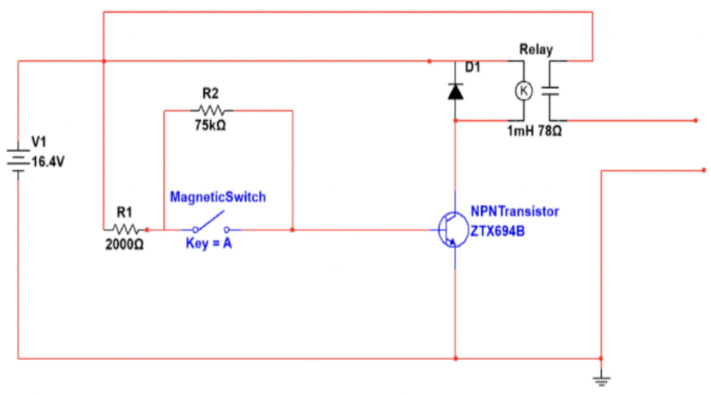


Figure 35: Phase IV Magnetic Switch (Multisim Diagram)

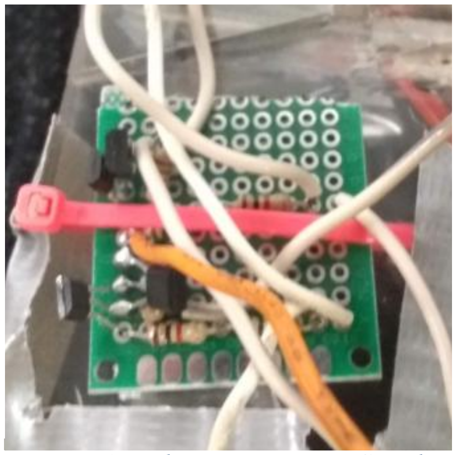


Figure 34: Phase IV Magnetic Switch (Reality)

As Figure 35 shows, the physical construction of the circuit ended up being difficult to decipher. This is because there are two copies of the circuit on the piece of proto board. Each instance of the circuit is responsible for one relay, and each relay connects power to one of the two pressure chambers. Because there are two magnetic reed switches there, it was possible to power on only one pressure chamber, and although this happened infrequently it was certainly an undesirable behavior

As part of our goals to improve usability and reliability, we ended up abandoning the magnetic switches altogether. Our alternative was a single mechanical switch that operates through the watertight bulkhead of the robot. This switch is sold by Blue Robotics specifically for this purpose, and we found we had parts of one in storage. For the first half of the year, a bulkhead penetrator bolt was modified to work with the switch. Eventually we replaced this repaired switch with a brand new one out of an abundance of caution. The new mechanical switch works very well, and has significantly improved the safety and usability of the robot.

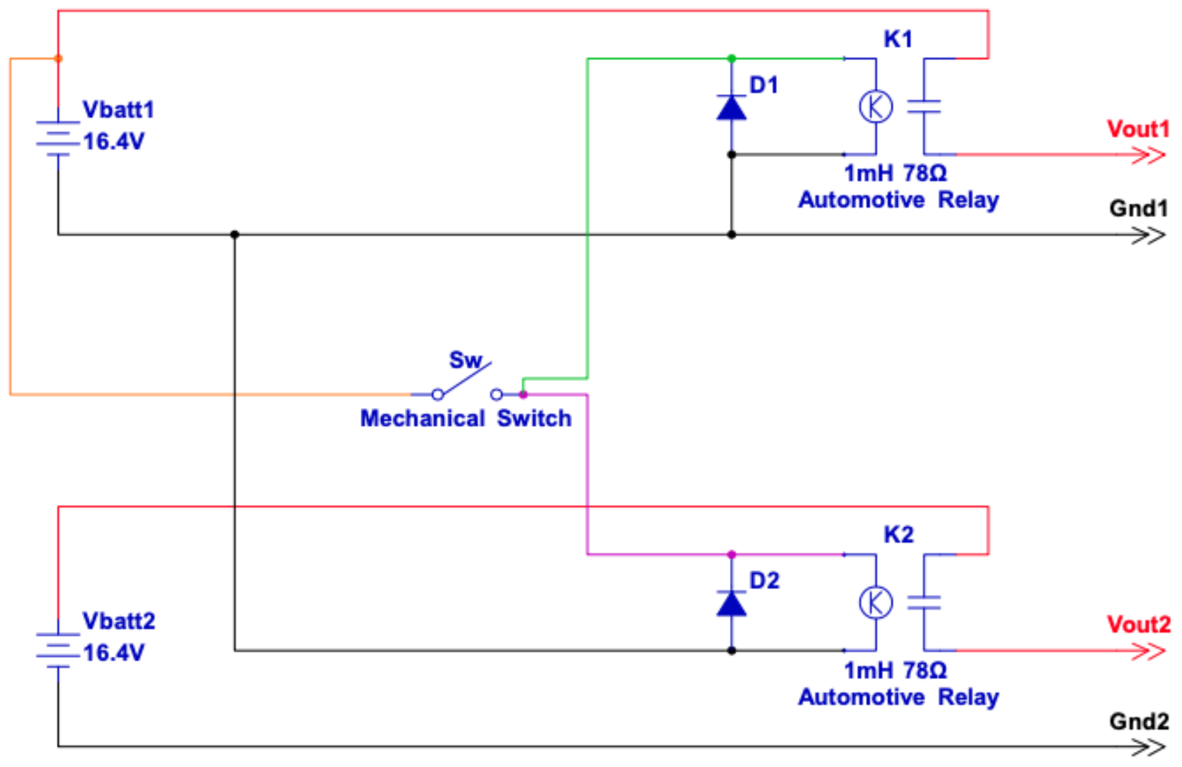


Figure 36: Switch circuit diagram for the on/off switch, clearly showing both relays and how they are connected

As seen in Figure 36, the new diagram more clearly illustrates how the combined system operates. It is worth noting that in the physical operation, either battery can be plugged into either of the sub-circuits without issue. However, it is preferable to power the switch from the bottom chamber's battery because the top chamber's battery has to power the thrusters and so is already discharging faster. This part of the redesign, too, was a success; the robot's new power switch operates reliably and should continue to do so indefinitely.

4.2 Spear Mechanism and Gearbox

4.2.1 Spear Mechanism

Before we began manufacturing the spear mechanism, there were a few small changes we wanted to make. The primary change we wanted to make was the addition of a force-dampening spring at the back of the mechanism to prevent the constant-force spring from destroying itself and the mechanism on impact. Secondly, we widened the back clamping mechanism to accommodate all the necessary hardware to assemble it without it interfering with the constant force spring.

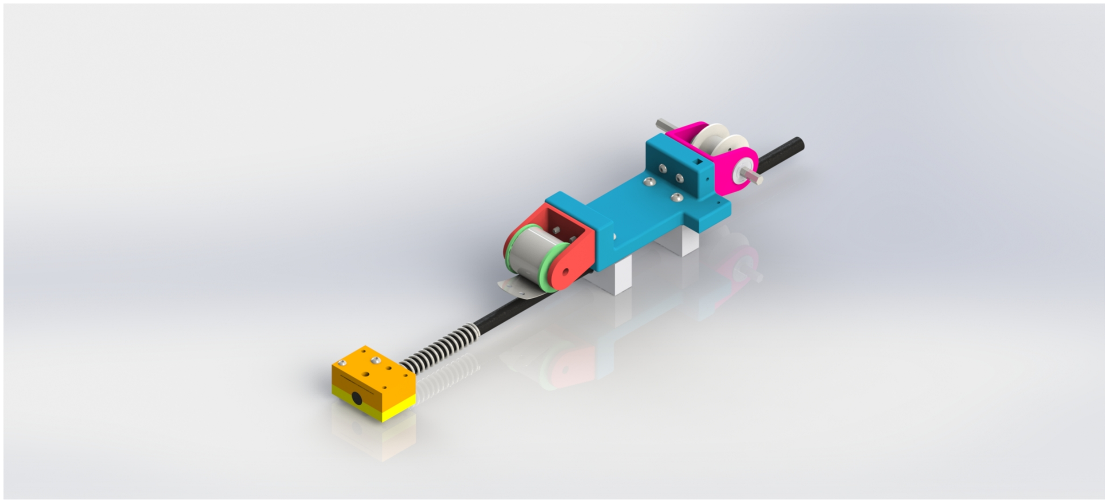


Figure 37.a: Spear Mechanism CAD Version 1

Once we had decided on the method for the spear mechanism, we finalized a version 1 prototype and manufactured it using 3D-printed components and parts from McMaster Carr.

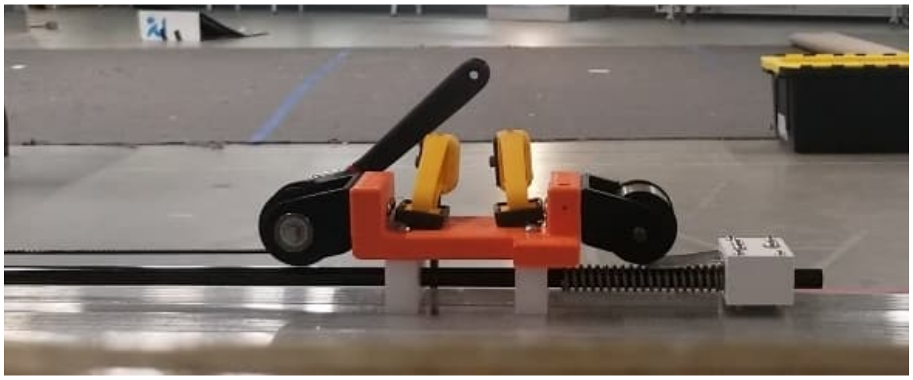


Figure 37.b: Spear Mechanism Prototype Version 1

The first iteration proved that the concept would work, but we quickly realized we would need to make some improvements. In the very first test, we looped the rope under the pulley and tried pulling it manually, but because the rope was pulling the top of the spear at an angle, it caused the linear bearings to cantilever and bind, making it impossible to pull back the spear. This had been a concern from the beginning, but by pulling the rope from an angle closer to the spear, we were able to pull back the spear correctly. Before changing the geometry of the pulley, we decided to fully attach the string to the pulley and wind it. As predicted, this increased the diameter from which the rope was pulling and decreased the angle between the rope and the spear, and consequently the moment being placed on the spear and bearings. With the fully wound pulley, we were able to successfully pull back the spear as we had hoped. However, this made us realize another problem. The damping spring that we had added was compressed by the constant force spring in its resting position, and was imparting another moment on one of the linear bearings. This risked binding the bearing again or possibly even damaging the bearing or the threads that attached it to the main frame of the spear mechanism. In addition, the impact force that would be imparted on the bearing when the spear was fired had the potential to severely damage the bearing. Because of this, it was determined that it would be necessary to design a wall into the main part of the spear mechanism with a hole large enough for the spear to go through but too small for the spring to go through to remove the force from the linear bearing. When this was brought up at an advisor meeting, the advisors suggested adding walls to both sides of both linear bearings to minimize any side-to-side wobble or cantilevering that may occur under ordinary use. This idea was added to the next iteration of the design, and improved performance substantially.

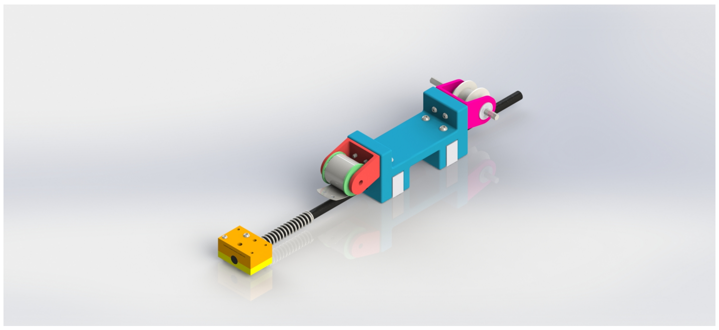


Figure 38: Spear Mechanism CAD Version 2

The new design was then manufactured and assembled for testing. Preliminary tests confirmed that it had the same functionality as the original design with the added protection for the linear bearings.

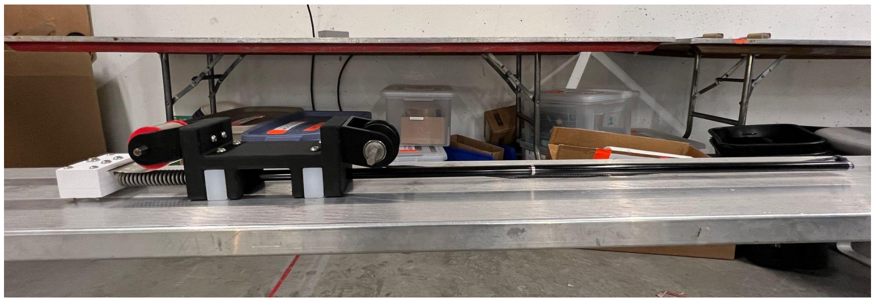


Figure 39: Spear Mechanism Prototype Version 2

However, when we attempted a full speed firing test, the force of the impact of the back carriage and the damping spring caused the 3D printed component that attached the constant force spring to the main frame of the spear mechanism to break.

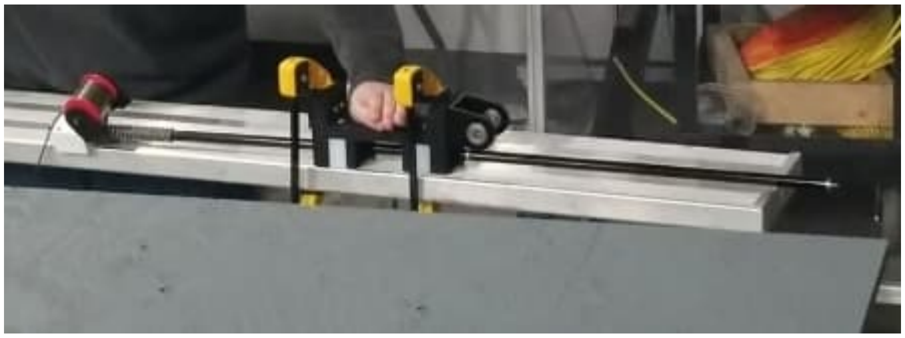


Figure 40: Spear Mechanism Prototype Version 2 Test Failure

This prompted a redesign of the constant force spring holder that we hoped would be strong enough to handle the impact that occurred when the spear mechanism was fired. We made the sides and back of the holder thicker and printed the new version, preparing to test the spear mechanism again.

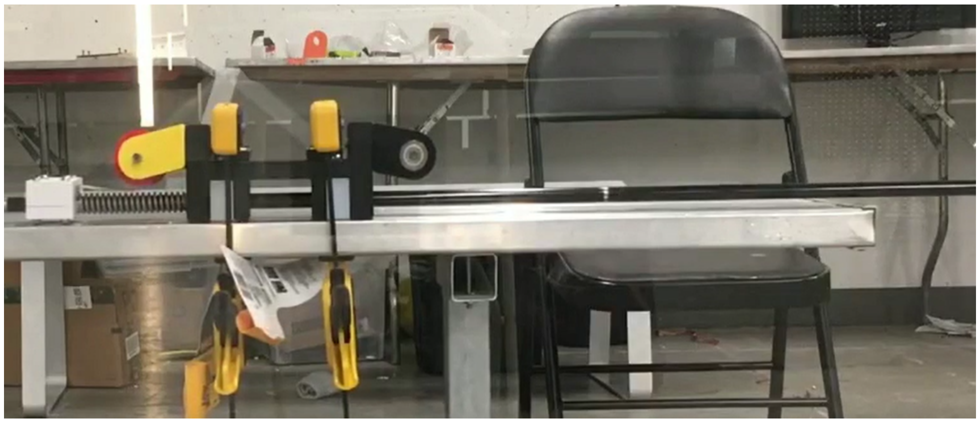


Figure 41: Newly Printed Constant Force Spring Holder

Unfortunately, this updated design caused yet another failure mode. This time, when the spear was fired, the 3D print that made up the main body of the assembly snapped off.

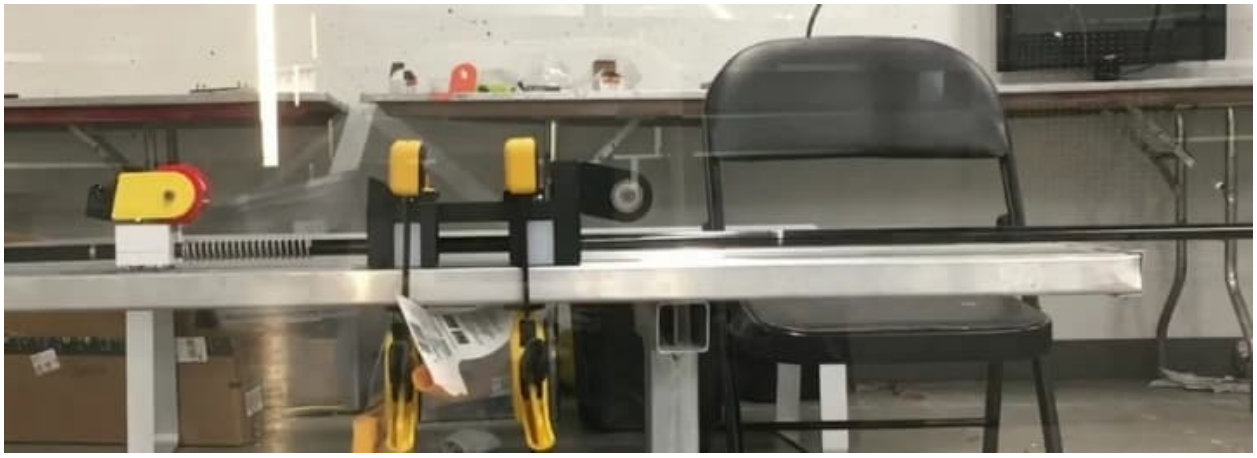


Figure 42: Spear Mechanism Prototype Version 3 Test Failure

After some discussion with the advisors, we determined that, since we had proven that the design concept and geometry would work, the best course of action would be to design and construct an aluminum version of the main body of the spear mechanism. To be used in a marine environment, the aluminum version would eventually have to be anodized or given some kind of protective finish, but for the purposes of this year's MQP we determined that this would be the most appropriate course of action. An aluminum version of the spear mechanism was designed with most of the same geometry, and the components were analyzed using Solidworks to determine that our assumptions that the materials would be more than strong enough to handle the impact were correct. Then, the components were machined on the manual mill in Washburn.

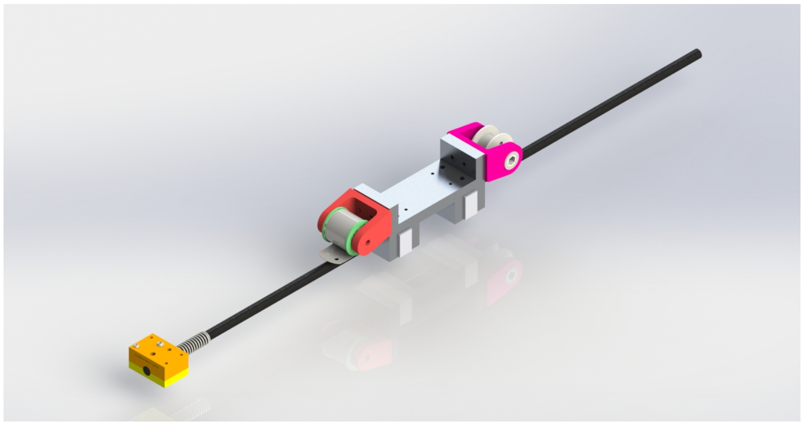


Figure 43: Spear Mechanism Prototype Version 4 CAD Render

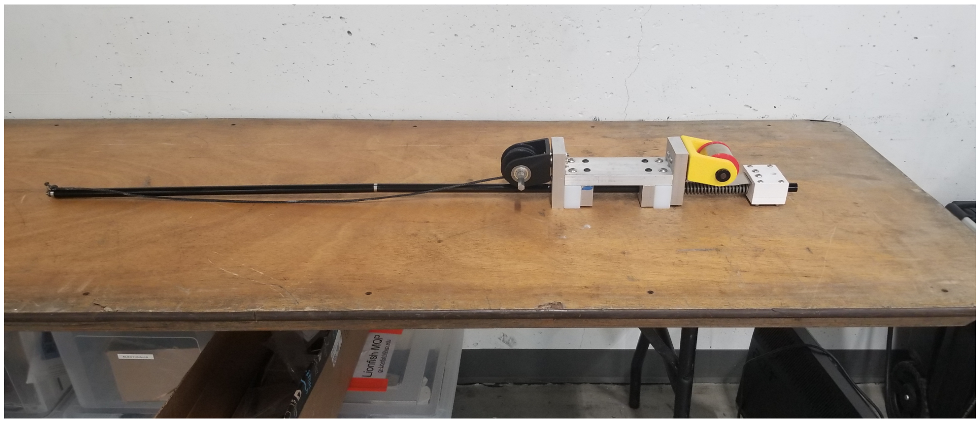


Figure 44: Spear Mechanism Prototype Version 4 Fully Assembled

Once the new aluminum version of the spear mechanism had been manufactured and assembled, we were eager to see if this design would be strong enough to handle the impact of the spring and back carriage when the mechanism fired. We attempted to do some static analysis on the constant force spring holder, but because of the anisotropic and extremely variable properties of 3D printed plastic we knew it would be extremely difficult to get an accurate result. Unfortunately, on the first test, the constant force spring holder broke yet again. At this point, we knew we would have to turn to more drastic measures.

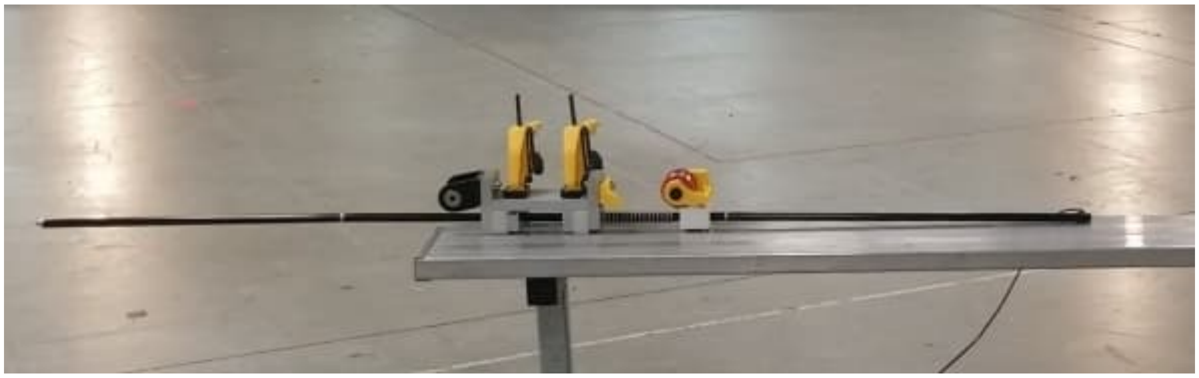


Figure 45: Spear Mechanism Prototype Version 4 Test Failure

While analysis for 3D printed parts is still unreliable at best, one of our group members had used parts printed in Onyx or Nylon X from Markforged printers on heavyweight BattleBots projects before, so we were confident that this material would hold up to the abuse our spear mechanism would supply. The next iteration of the design was essentially the same, but was printed on the Markforged in the RBE lab and had some slight geometry changes to account for bolt heads in the aluminum design that we quickly realized our old design had not accounted for, but was easily remedied with a few washers for the initial test. Once we obtained the Markforged part, we were finally able to conduct our final test. At long last, when the spear fired, the constant force spring holder did not break! The only minor damage any of the parts sustained were some chips in the plastic pulley on which the constant force spring was wound, but this did not pose any major structural issues. Figure 46 shows a picture of the final spear assembly with the Markforged part.

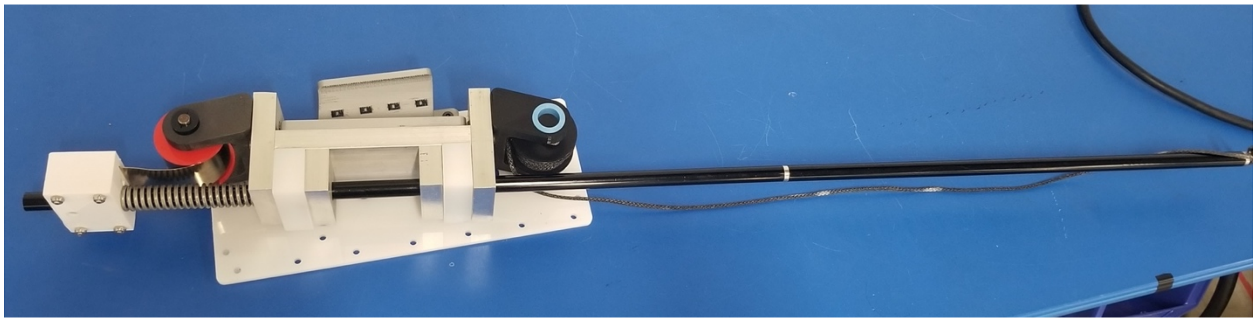


Figure 46: Final Spear Mechanism Prototype Fully Assembled

4.2.2 Spear Mechanism Gearbox

After we had determined that our spear design was viable, we began designing a gearbox to control the winch that cocked the spear mechanism. The first step was to determine the amount of torque that would be required to turn the output shaft.

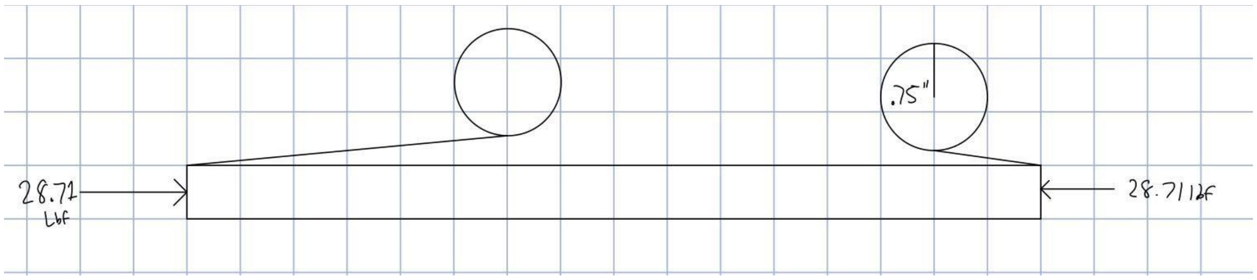


Figure 47: Basic Force Diagram of Spear Mechanism

We assumed the angle between the constant force spring, winch, and spear shaft to be negligible, and used the force of the constant force spring, 28.71 ft-lb, as the force that would be pushing on the spear shaft and therefore the force that would need to be pulled back by the winch. The winch drum had a diameter of 0.75 inches, or 0.0625 feet. Therefore, we could calculate that the output torque of the shaft would need to be at least 1.794 foot-pounds, or 2.433 Newton Meters.

$$\frac{.75}{12} = \frac{3}{48} = \frac{1}{16} = 0.0675$$

$$\begin{array}{r} 28.7100 \\ \cdot \quad .0625 \\ \hline T_o = 1.794375 \text{ ftlb} \end{array}$$

$$1.794375 \text{ ft lb}$$

$$T_o = 2.432845831 \text{ Nm}$$

Equation 1: Output Torque Calculation

However, we were concerned that a gearbox that would support this output torque would not be strong enough to support the output torque necessary to pull back a larger constant force spring if we determined it would be necessary to have a larger spearing force. Because of this, we decided to do the calculation for the output torque for the 40.9 ft-lb constant force spring that we had considered using initially.

$$\begin{array}{r} 40.900 \\ \cdot \quad .0625 \\ \hline T_o = 2.55625 \text{ ftlb} \end{array}$$

Equation 2: Ouput for Larger Constant Force Spring

Next, we had to calculate the output torque of the motor to determine the necessary gear ratio. We decided to use an M200 Thruster from Blue Robotics to run the winch, as the Phase 1 MQP had also done this, and the motor would already be waterproofed. There was extensive information on the motor specifications and thruster force in a spreadsheet from Blue Robotics' website, as the motor came from the thruster module, but unfortunately the torque output was not calculated in the spreadsheet. However, we knew we would be able to calculate the torque with the information that was provided. We were able to determine the output torque of the motor by using Equations 1 and 2:

$$Mechanical\ Power\ (W) = \frac{Thrust\ (g)}{Propeller\ Efficiency\ (g/W)}$$

Equation 3: Mechanical Power of Motor

$$\tau\ (N.m) = \frac{Mechanical\ Power(kW)}{\omega(RPM)\dfrac{0.10472\ rad/s}{1\ RPM}}$$

Equation 4: Output Torque of Motor

By creating two new columns in the spreadsheet with these equations in them, we were able to graph the motor efficiency curves and determine the most efficient output torque to use, as shown in Figure 48.

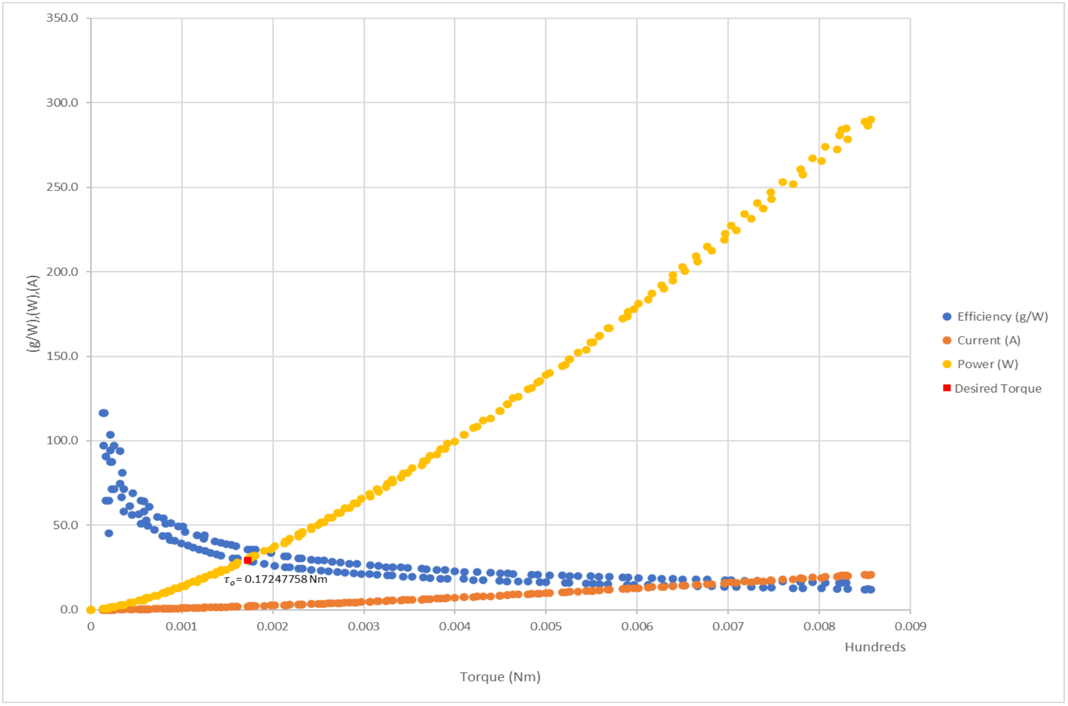


Figure 48: Efficiency Curve of M200 Motor and Desired Torque Marked

Once we had determined the motor torque we wanted to use, we could then calculate the necessary gear ratio for the spear mechanism gearbox. Using the maximum spear torque we calculated, we converted the value to Newton Meters and divided it by the desired motor output torque and got a gear ratio of 20.15:1. We decided to raise this to a gear ratio of 30:1 to allow for a factor of safety and make it a round number that would be easy to create a gearbox for.

$$\text{Max spring torque: } \frac{\overset{\tau_{omax} = 2.55625 \text{ ft lb}}{3.4658096303}}{0.172} = 20.15:1$$

$$\Rightarrow 30:1 \text{ gr for Safety}$$

Equation 5: Gearbox Ratio Calculation

We then began designing the first prototype of the gearbox. Our plan was originally to make a simple unsealed version of the final gearbox to ensure that it would work, then to use the same gears and components on the final sealed gearbox. We had planned on making a sealed gearbox with water-resistant or waterproof components in case of an accidental leak, as we were unsure if it would be reasonable to lubricate the gearbox with the surrounding saltwater as opposed to traditional grease. We knew this would be very challenging, however, as we needed a way for the output shaft to be able to exit the gearbox and rotate without allowing water into the gearbox. However, after investigating several gear sources, we determined that VexPro's hex bore and dog-shifting gears had a waterproof Teflon-infused ceramic coating that would be functional in seawater and would lend itself well to the shifting gearbox. Once we were certain that we could source hardware and electronics that would be able to operate in the ocean, we decided to make the gearbox unsealed and use the water as a lubricant instead of traditional grease.

The first thing we had to do was determine what gear train to create to make a 30:1 gear ratio using the gears that VexPro had. Generally 5:1 is the highest gear ratio that is recommended for spur gears, so we determined that the best combination of gears would be a 5:1, a 3:1, and a 2:1 to accomplish the 30:1 gear ratio, and then two more stages of 1:1 gears that would contain the locking mechanism and the free-spin mechanism, as both could only have one gear on the shaft in order to be functional.

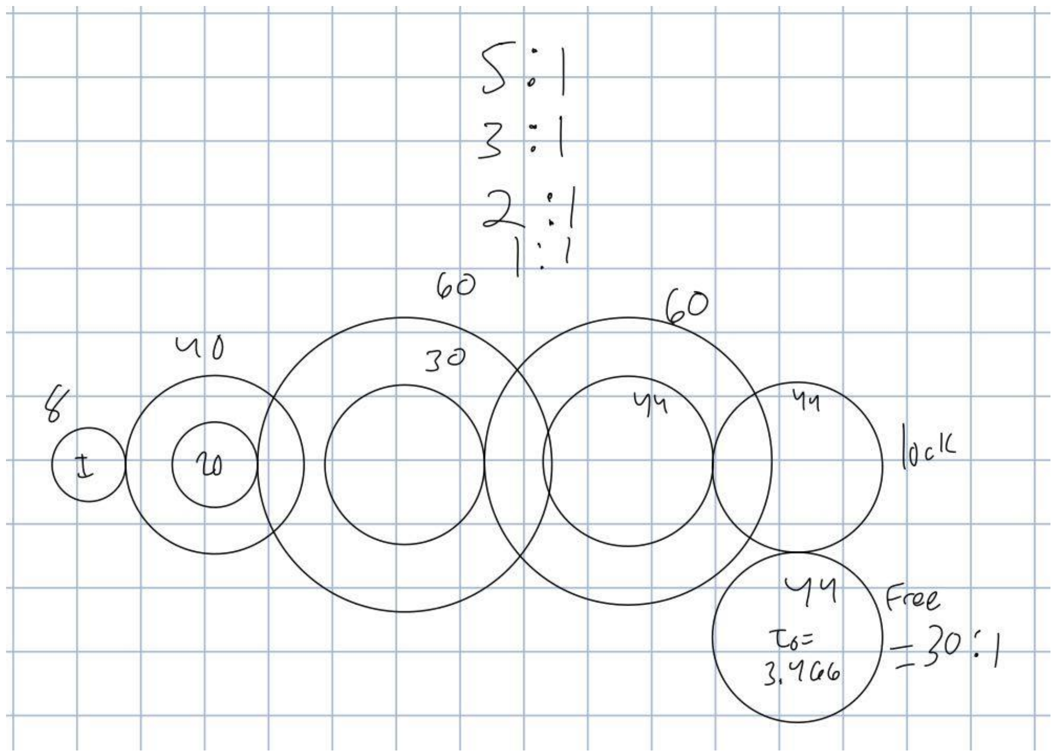


Figure 49: Layout of Spear Gearbox Gear Train

Originally, we were going to make our first stage a 6T:30T gear ratio, as 30T was the largest steel gear offered by VexPro and the motor pinion for the size of motor we needed was only offered in steel, but this caused interference between the shaft and the motor, so we decided to mesh a steel 8T gear with an aluminum 40T gear with the understanding that the 40T gear would likely need to be replaced sooner than some of the other gears. We chose a 20T:60T gear ratio for the 1:3 stage, and a 30T:60T ratio for the 1:2 stage. We then chose to use 44T gears for the remaining 1:1 stage.

Once our gear train and gears had been chosen, we began the CAD for the gearbox. We used VexPro's dog-shifting shaft and gears for the locking and free spinning stages. We were originally going to use VexPro's ball-shifter gearbox, as we thought it would be easier to get the individual components that we needed without buying a full gearbox, but we were concerned that this version would not hold up very well in saltwater. We soon realized that it would be possible to buy only the components of the VexPro dog shifting gearbox that we needed for our design. Having learned this, we switched to the dog shifting gearbox instead, because it used a dog gear instead of a ball-bearing based shifter and was made of more corrosion-resistant materials. We also used VexPro hex bore gears and ThunderHex shaft for the remaining stages. We chose to use the ThunderHex because VexPro also sells acetal bushings that ThunderHex can be used with instead of using traditional ball bearings. This was perfect for our application, because acetal plastic is resistant to saltwater, unlike most traditional ball bearings. We did, however, must buy a few acetal bearings from McMaster-Carr to adapt the shifting stages to use bushings instead of bearings.

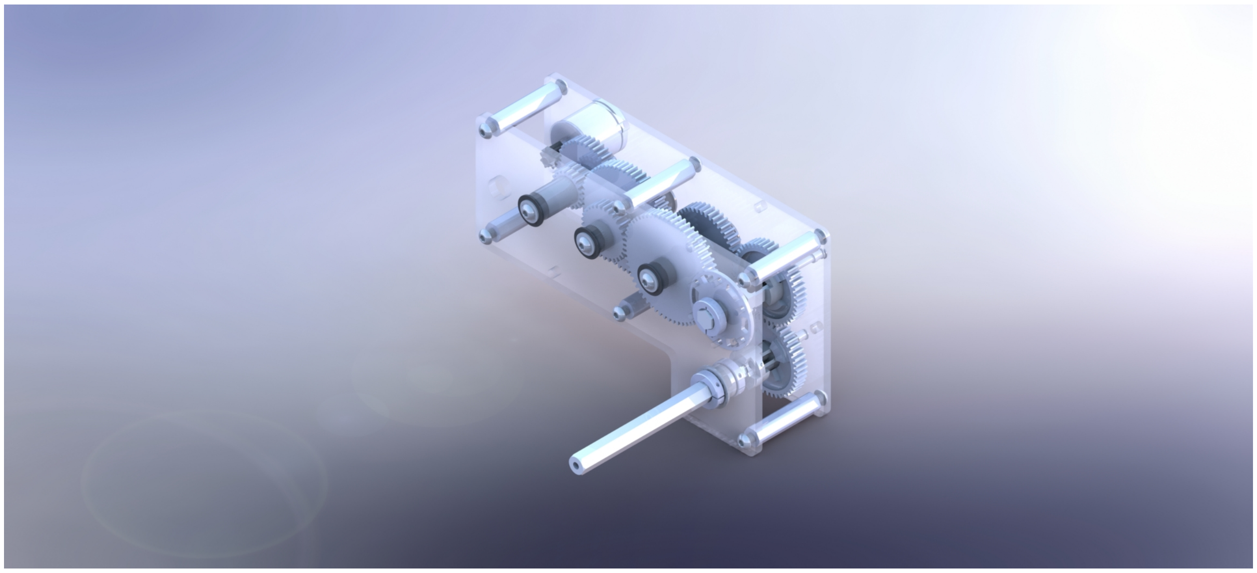


Figure 50: Render of First Gearbox Design

Normally a dog shifting gearbox such as this one would shift between 2 different gears, engaging with one or the other to change the torque and speed at which the output shaft runs. However, our application uses the dog shifter a little differently. The locking stage is normally not engaged with the gear on its shaft, instead allowing it to be a free-spinning idler gear. This is because the shifter shaft is locked into a hub to prevent it from spinning. However, once the motor reaches the desired setpoint, the dog shifter shifts into contact with the gear, engaging it with the locked shaft and preventing the gearbox and output shaft from back driving. The free-spinning stage on the output shaft, on the other hand, normally is in contact with its shaft so that the output shaft transmits the power from the gear train. However, when the robot has identified its target and is lined up for the shot, the output shaft shifter disengages with the gear, allowing the output shaft to be spun freely by the force of the constant force spring and allowing the spear to be thrust forward. Once it has been fired, the shifter will re-engage with the output gear, and the locking stage will unlock, re-cocking the speargun and re-locking it into place in the desired position.

This design also allows the motor to drive backwards, so in the event the spear is armed and needs to be disarmed before it is brought ashore, the gearbox can simply unlock and be driven backwards without being shifted into free-spinning mode.

Once this concept was finalized, little about the overall design was changed. However, once we began integrating the gearbox with the spear mechanism and determining how to attach it to the robot, we decided to make some minor changes. First, we decided to make all the gears in line with each other horizontally instead of having the output shaft on the bottom, as we wanted the mechanism to stick below the robot as little as possible. Secondly, we realized we would need to make the gearbox slightly thinner to allow it to be centered and still fit with a potential side-mounted containment system. This was done easily by replacing some of the spacers in the gearbox with thinner ones. We also finalized the motor mount design and integrated it with the gearbox.

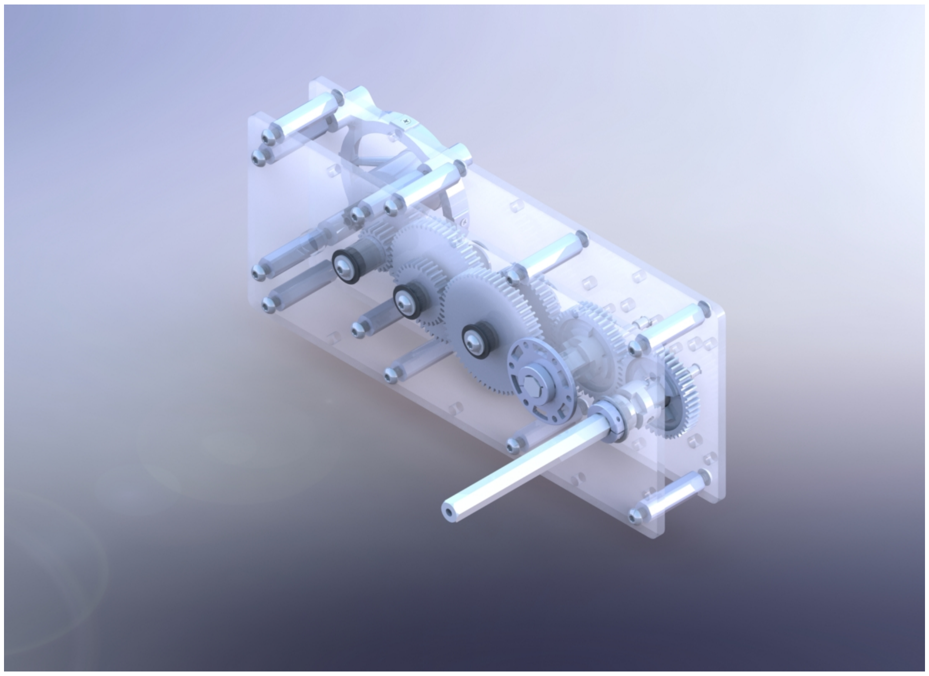


Figure 51: Render of Final Spear Gearbox Design

Finally, we realized we wanted to tilt the spear down slightly so the robot could line up to the lionfish more easily. This is because lionfish tend to float fairly close to the ocean floor, and we did not want the robot to have to be dragging the spear mechanism so close to the bottom of the ocean in order to line up to the fish. This was accomplished by designing a 3D printed mount to attach to both the base of the spear mechanism and the side plate of the gearbox that would attach the two assemblies together at the desired angle.

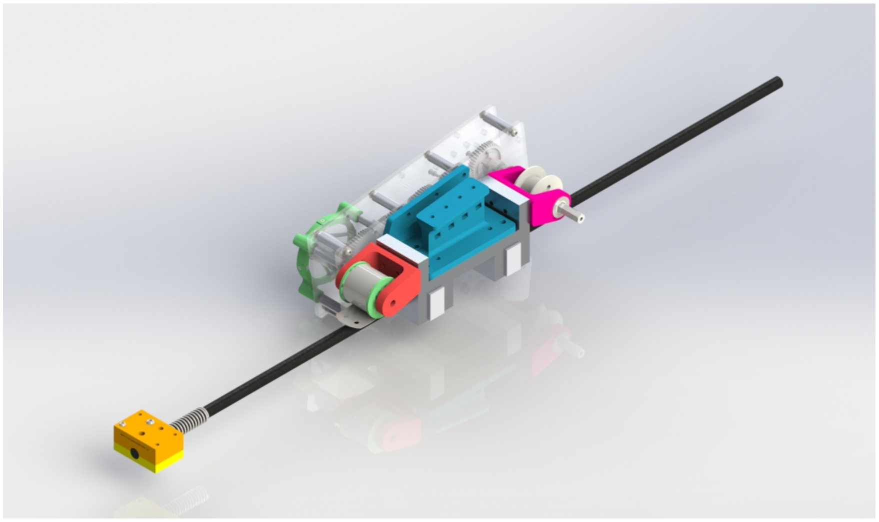


Figure 52: Final Spear Mechanism and Gearbox Integrated Design Render

Unfortunately, due to delays in shipping, the gearbox components were not able to arrive before the end of the year. However, detailed instructions for constructing the gearbox and attaching it to the spear mechanism are included in the "Future Work" section 5.2

4.3 Containment System

While our focus throughout the year remained on the spear mechanism, progress was made on a potential containment system that may be investigated and improved on by future MQPs. We partially based our design on the containment unit created by the Phase IV Lionfish MQP, and utilized their containment units that had already been proven to minimize drag in on the robot compared to many other designs of previous iterations. However, we modified the cages to be attached sideways to each side of the robot and to have a closed top and bottom with a small opening in the front through which lionfish could be inserted. The CAD is still in very preliminary stages, but we modified Phase IV's existing LCU to attach sideways to the robot, as shown in Figure 53. As previously mentioned, the final design would have a one-way valve that the spear would shoot through attached to a tube that the lionfish would be pushed through to one of the LCUs mounted on the side of the robot.

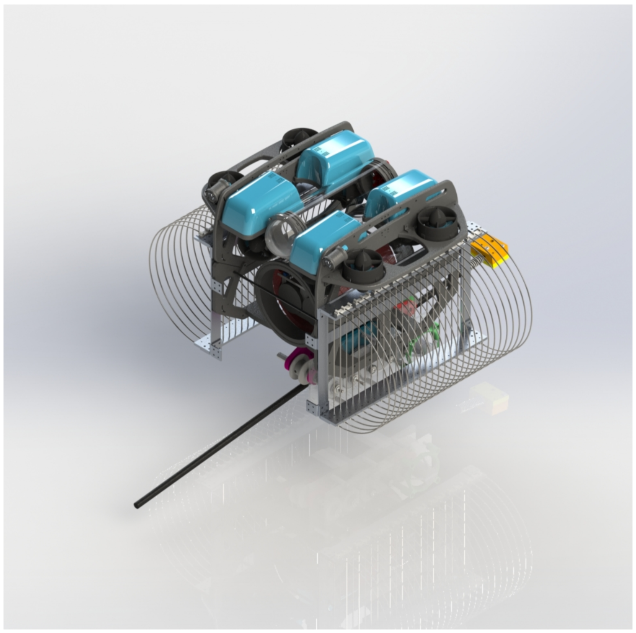


Figure 53: Preliminary CAD of Containment Unit

4.4 Weight System

While originally identifying issues with the robot, our team found the weights system in use to be inadequate, as it was just a few lead weights in the main electronics compartment of the robot. The weights themselves were wrapped in duct tape and loose in the compartment which allowed them to move freely in the electronics compartment. This resulted in a scenario where if the robot rolls, the weights could hit the electronics and batteries and damage the internal electronics of the robot, causing major damage to the system and making it unsafe and possibly destroying the ability to control. In addition, the free movement of the weights, resulting in them shifting back and forward, could also significantly affect the trim of the robot, causing instability. Other desired outcomes for the weight system were to be able to correct the listing and trim of the robot. Therefore the team determined our design must have the weights fixed externally, to ensure the listing and trim of the robot will no longer be negatively affected by the weights system. Instead, the weights remain in one place, but the amount of the weight can be easily changed to ensure stability while the robot is underwater. The robot is designed to function vertically in the water column. By the continued use of the fixed weights on each of the four sides of the robot's frame, in conjunction with the fixed weights now on the central canister, listing is no longer an issue.

Having identified the free movement of the weights to be an issue as far as the trim and protection of the electronics, the team decided an external fixed weight system was required. The team developed several prototypes of the fixed weight system. This led to designing the weights system by modifying and utilizing the current holders for the central electronics canister. Some design ideas had components where a snap in for the weights or latch system was needed. These all were decided to have too much of a possibility of parts snapping or breaking so instead the design that the team chose was the design that made modified the central canister holder to have a closed end to which the weights could slide in and out of for ease of access, maintenance, and preexisting hardware. This saves on cost and prevents a total redesign.

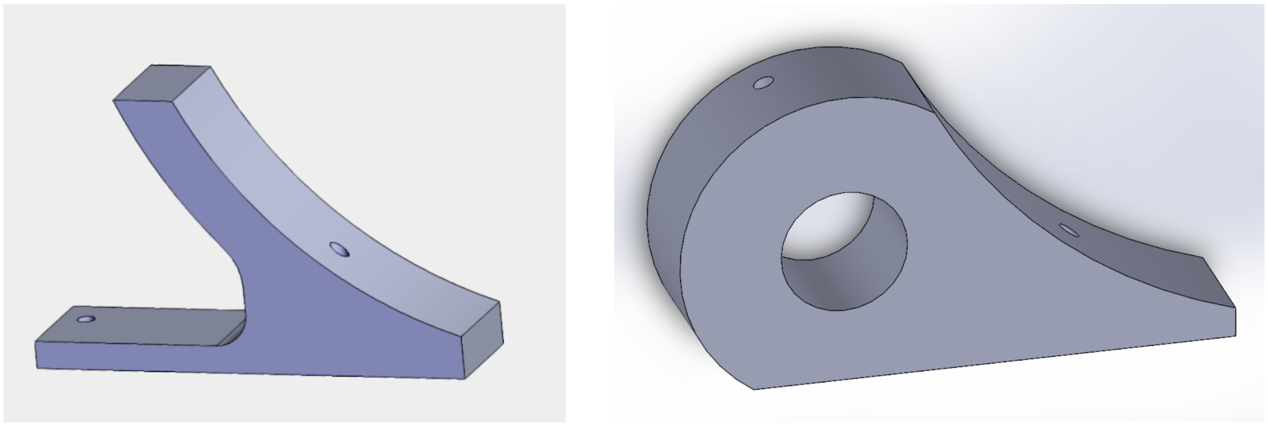


Figure 54: Previous design (left) and our redesign (right) of the central electronics canister holder

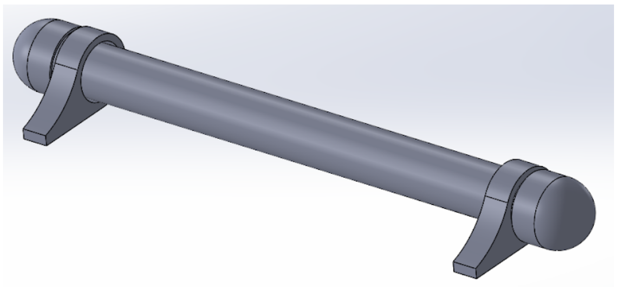


Figure 55: Final design of the improved weight system

The weight tubes are made from PVC tubing with PVC end caps and are filled with lead shot such as is used in SCUBA diving weight belts. There is one tube on each side which is secured by the modified canister holders. Each PVC tube has one end cap epoxied on. To remove the tube, you simply remove the end cap that is not epoxied on, and slide the weighted tube out. To install a different weight tube the user can slide it through the holders and then replace the end cap to secure it in place. This method also ensures that the weight system cannot fall off the robot and damage the reefs and corals that it is designed to protect. The tubes themselves have varying weight, consisting of three separate weights in each tube of different amounts. The distribution of the weight in the tubes for the preliminary prototype is a gradient from the front of the robot being the heaviest and the back being the lightest ranging from one to five pounds. This is done by using lead shot of different weights and sizes which are five, three and two pounds respectively from front to back. The lead shot repurposed from scuba weights is tightly packed into the tube to prevent the weights from shifting out of place.

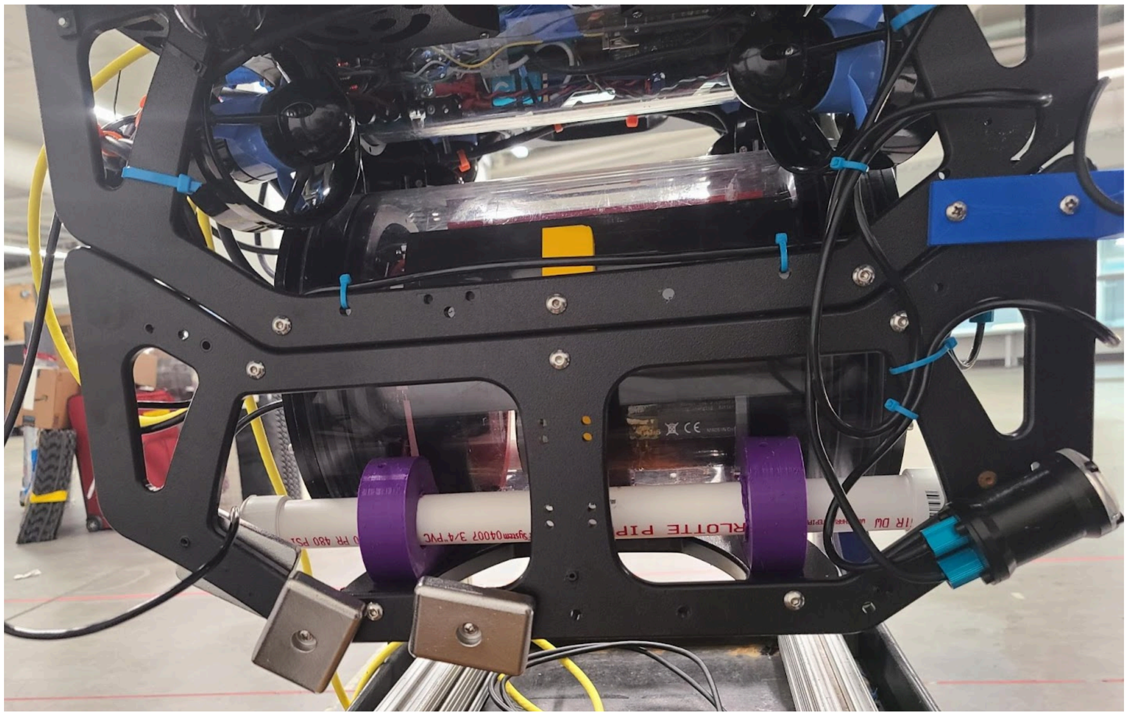


Figure 56: Installed weight system from the right

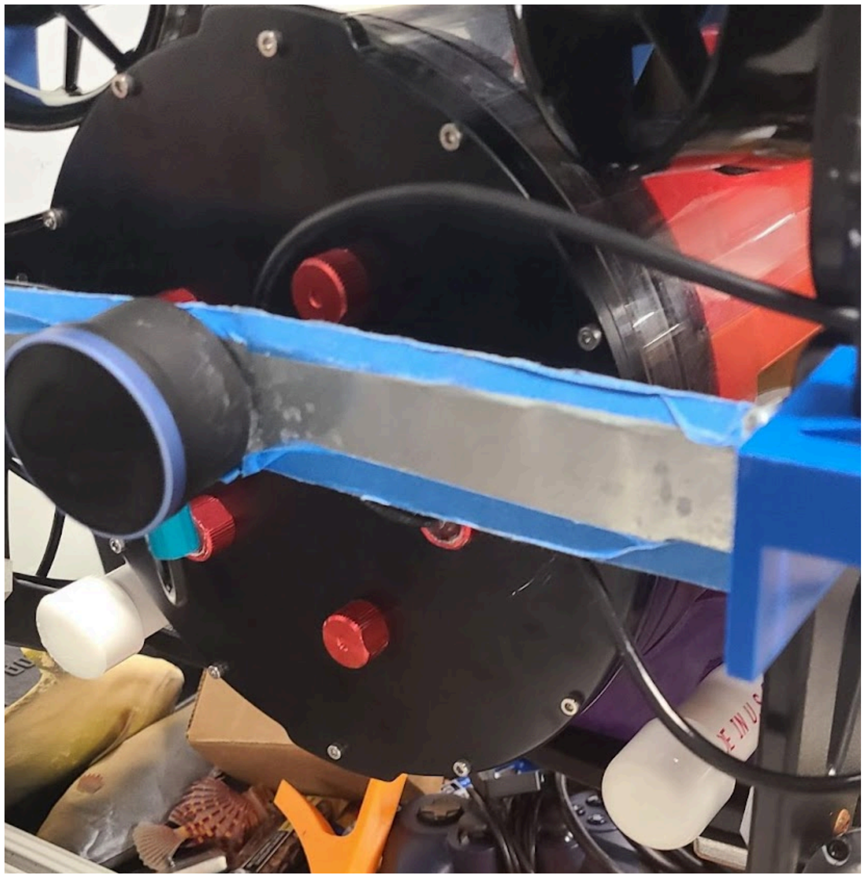


Figure 57: Front view of the installed weights system

Thus, the tubes themselves have varying weight, to offset the weight from the spear and gear mechanism which moves the center of gravity towards the back and bottom of the robot. This offsets the imbalance created by the weight from the spear and gear mechanism, which had moved the center of gravity towards the back and bottom of the robot, and helps the robot achieve neutral buoyancy. The result of the weight distribution in the tubes with heavier weights towards the front of the robot with the lighter side being towards the back assists with buoyancy and helps to move the center of gravity back towards the middle of the robot. The tubes are designed with small holes in them which are smaller than the weights themselves to allow the flow of water to purge any air in the system, negating any buoyancy created by the weight system itself without the lead shot being able to exit the tube.

Different locations and other local factors like salinity will change the amount of weight the robot needs. For example, the team's pool testing requires a different amount of weight in the system than a higher salinity location like the dead sea would need to keep the robot neutrally buoyant. The design of the weight system allows for ease of access to the weights and gives the user the ability to easily switch out the weights, making the robot more versatile for different locations.

4.5 Computer Vision and Object Detection

We initially trained our model to detect “face masks” as due to COVID-19, most people would be wearing facemasks and thus would help us identify most of the edge-cases while developing and testing the program. We used a probabilistic detection model that outputs the probability of how confident the ML model was at classifying the identified object into a pre-labeled category (such as “face masks” or “fish”). The model then returns the maximum probability amongst all the categories and the label related to that probability as the “confidence” of the program (Figure 58). While testing the “face-masks” detection program on the webcam, we observed that the program returned a confidence level of more than 95% (Figure 58). However, we observed that the confidence level of the program would drop when it successfully identified two masks in the same frame (Figure 59). Nonetheless, the program was very accurate at identifying facemask and tracking it as one would move their face within the camera frame.

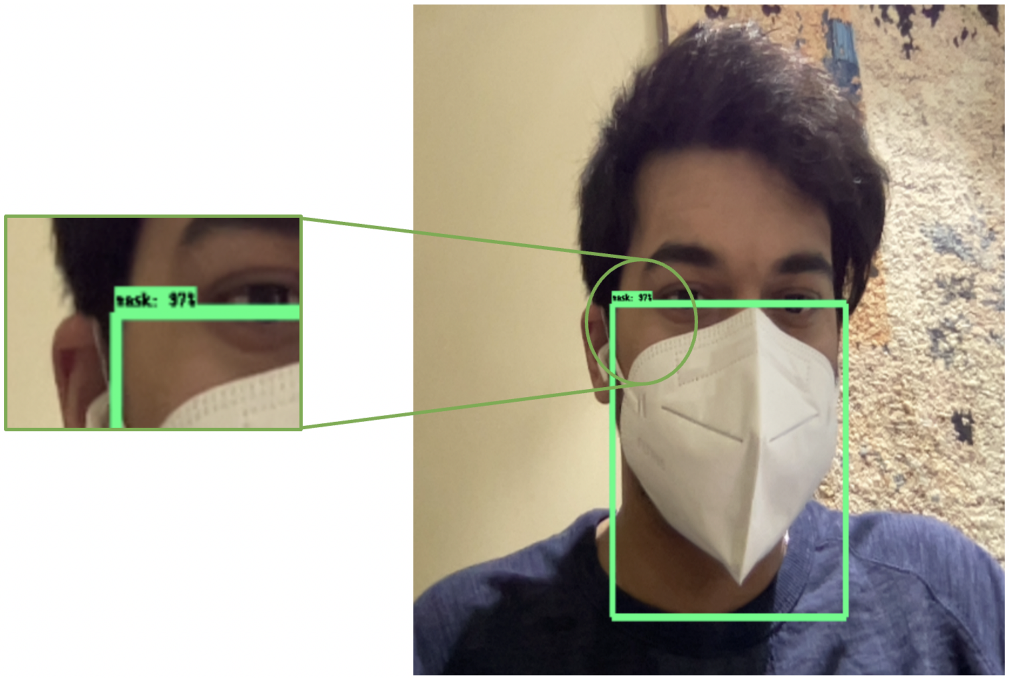


Figure 58: The program identifying facemask with 97% confidence

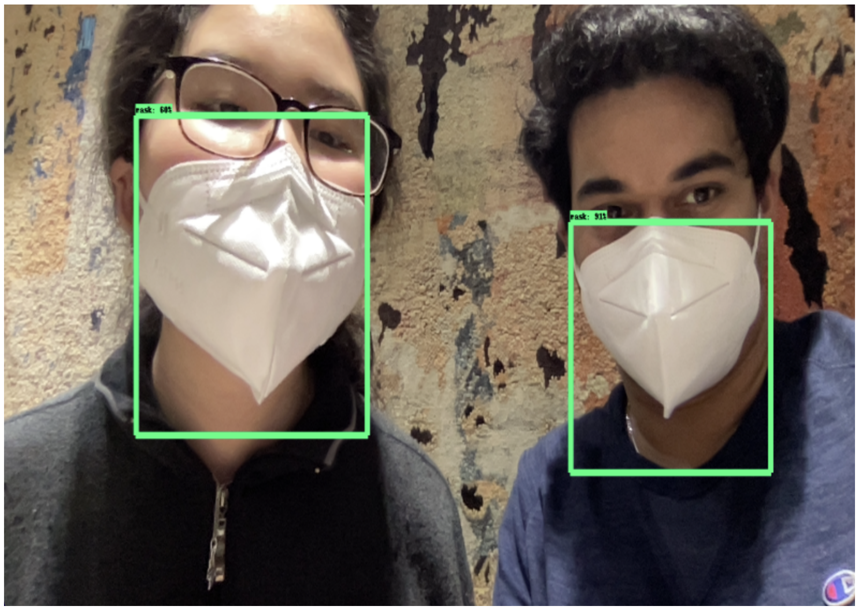


Figure 59: The program identifying facemasks with 69% (left) and 91% (right) confidence respectively

When we were satisfied with the program's performance at identifying facemasks, we then trained the model to identify "fish". As there were more varieties of fish than facemasks, we had to expand our training dataset to include different species of fish. Since we would later converge to accurately identify Lionfish amongst other fish, we thought it would be better to first train the model to correctly identify fish and later distinguish between Lionfish and other fish. This would further prevent the model from accidentally identifying a person as a Lionfish. The program would first check if the unknown object is a person or a fish, and then check if it is a Lionfish. While training the model, we got the minimum loss as 0.78. As the ideal loss is supposed to be between 0.1 and 0.2, we decided to train the model further in hopes that the loss would reduce further. However, after a while, the loss started increasing, indicating that the model is at the limit of overfitting. We kept the batch-size of our training model configuration to be a small value resulting in a huge discrepancy between each learning iteration. Increasing the batch-size would result in less discrepancies per iteration, and further decrease the loss (resulting in a better fit learning curve), but would also result in less accurate object-detection. As we wanted to maximize the model accuracy, we decided to keep the batch-size a small value and stopped the learning program at the loss of 0.75.

We tested our "fish" detection program on toy fish and observed that the program would detect fish with more than 97% confidence in real-time (on a webcam) and with no significant framerate drops. The program was also able to track the fish if it was within the camera frame. The program was able to identify fish even when we rotated the fish around its axis. We observed that when the fish is held sideways in front of the camera, the program relies on the streamline shape and its fins to identify the fish (Figure 60). As we turned the fish to face the camera, the program relies more on eyes and the shape of the head to identify the fish (Figure 61). As the fish's movement is dynamic underwater, we tried emulating it by attaching a string to the fish toy and suspending it freely so that it moves like a pendulum. We observed that the program was consistent at detecting the fish and the bounding-box around the fish would move with the fish, keeping it within the bounding-box (Figure 62).

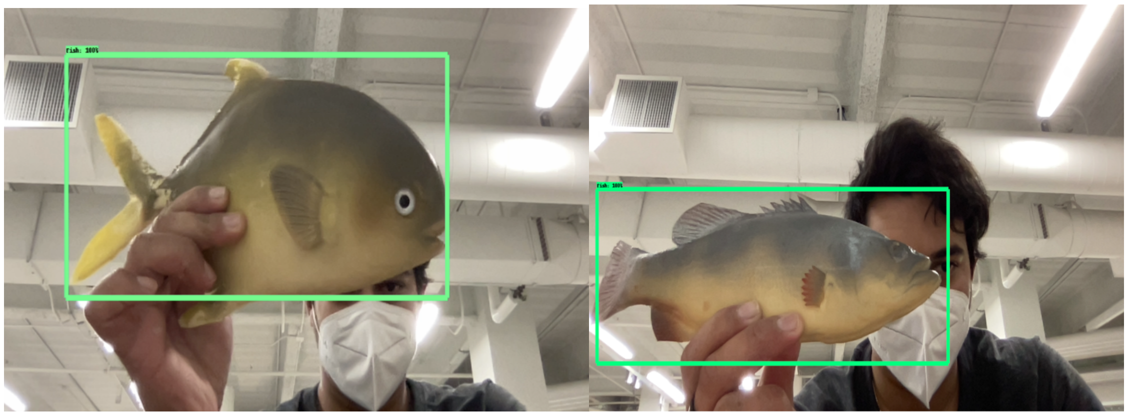


Figure 60: The program identifying fish with 100% confidence (when the fish is sideways)

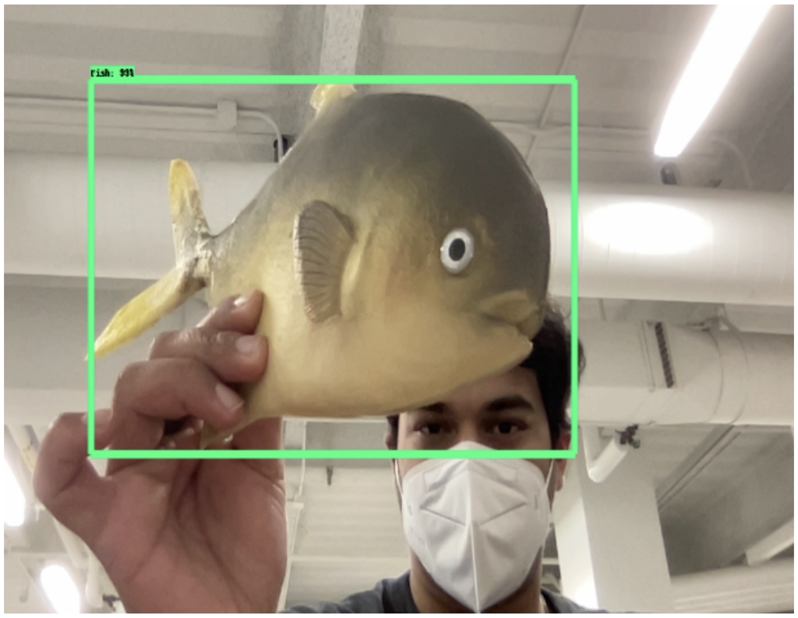


Figure 61: The program identifying fish with 99% confidence (when the fish is facing the camera)

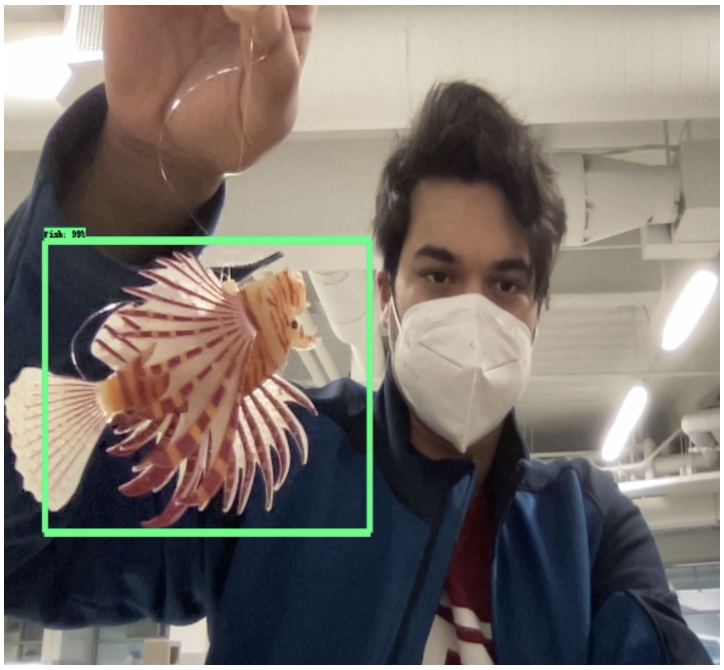


Figure 62: The program identifying fish with 99% confidence (When suspended by a string)

When we were satisfied with the output of the model in detecting fish, we tried to test the model on the top-module camera of the robot. However, due to some connection issues, we were not successful in getting the live feed from the camera and running it on the object detection program. Thus, to confirm the compatibility of the object detection program with the robot's camera feed and the ability to ensure accurate object detection underwater, we ran the robot in the pool with the Lionfish toy submerged underwater and suspended with a string. We recorded the camera feed from the robot's top module camera and passed the video recording to the object detection program. As the size of the toy is smaller than the size of the actual lionfish, the robot had to get close to the toy for the object detection program to register the toy as a fish.

Nevertheless, from Figures 63 and 64, we observed that the program was able to detect Lionfish underwater and thus, no further modifications were required to the object detection program to make it run underwater. Furthermore, since we trained our model to identify fish only as the first iteration, we can observe in Figure 64 that our program was able to successfully identify Lionfish even when a diver was present in the frame. In the future, when we train the model to identify both, the diver, and the Lionfish, we would superimpose the two identified object bounding boxes in the same frame. As soon as the program identifies the object as diver, it would trigger the spear mechanism to lock in place so that even if a Lionfish is detected in the same frame as the diver, the robot would not shoot the Lionfish. We plan to keep the locking mechanism turned on for a few seconds after the diver has exited the robot's frame to further reduce any chances of robot misfires.

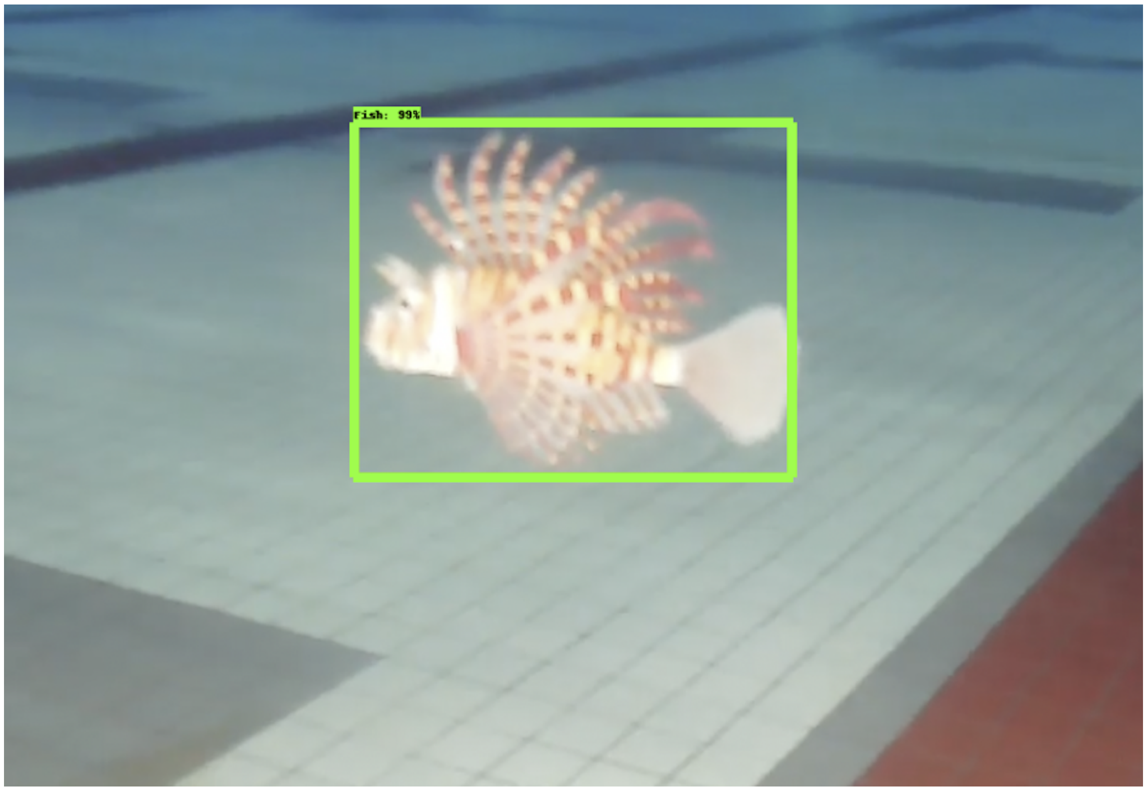


Figure 63: The program detecting Lionfish underwater with 99% confidence

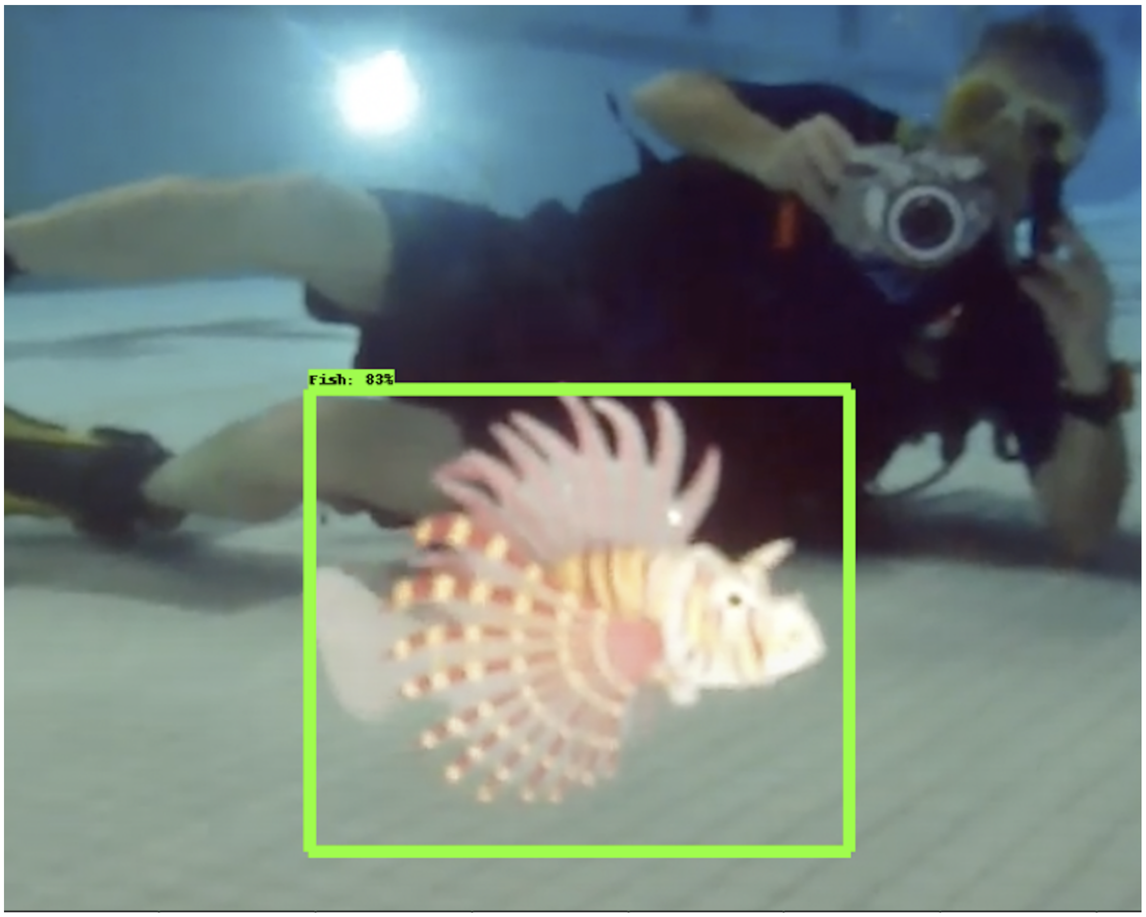


Figure 64: The program accurately identifying Lionfish with diver (Prof. Craig Putnam) in the frame

4.6 Movement Control

Movement control can be done manually or autonomously. Manual control can be done using an Xbox or similar joystick controller, or by typing in commands via the command line. A computer running QGroundControl free drone control software is used to facilitate manual control with either a joystick controller or by using the keyboard. Autonomous control can be achieved using Joystick emulation or Vector control. Joystick emulation is based on RC channel commands imitating manual control using joysticks. Vector control works by giving the ROV an axis (x, y, z, r) to travel upon which bypasses the RC channel control structure eliminating the need to imitate a controller.

The Joystick emulating control structure was initially difficult to work with, but became more reliable at the end of first phase of dry testing. The Joystick emulation manual control did not always display the same behavior as it had shown during dry testing. The time control failed during the first set of tests but was fixed after a quick debugging session. At one point, QGroundControl reset the axes, causing some dramatic undesired operation where the robot would flip itself over repeatedly. This was simple to reset, but the possibility of this error occurring again once again highlighted the inadequacies of this method. Its sensitivity to changes in orientation and control scheme via QGroundControl make it undesirable for long-term usage. The vector control method was developed as a response, though it proved very difficult to control because it activated the dive motors as well as the appropriate motor group being called, with all commands resulting in the robot diving to the bottom of the pool. This was eventually determined to be a compatibility issue due to vector control using a drone code library with the up/down directions inverted. While many of the extra features this offers is desirable, not all of them translate so well from air to water. In a drone, fighting the constant force of gravity requires an extra -9.8 m/s of acceleration to be factored into any other movement. In our robot, the force of buoyancy which it needed to counteract to keep the robot below-water is not nearly as strong. The other significant issue is that vector control mode is set up to use up to six thruster maxes, while the current ROV being used is of the heavy lift configuration which has eight thrusters. Further research on the company form has revealed a promising lead on free code to modify the current version of the pymavlink folder which adds a mavactive.py file and a rc_override_example.py file to the pymavlink folder. These changes are implemented in a separate branch, but their effectiveness is untested.

5 Future Work

5.1 Spear Mechanism

Future iterations of the project may want to determine whether the robot's mobile aiming abilities will be sufficient for spearing the lionfish without assistance from a secondary automatic aiming mechanism for the spear. We attempted to make it compact enough that such an addition would be possible, but it may be necessary to make some changes to the design to make this work in a future iteration. One design that we considered when questioning if such an aiming device might be necessary was a modified Stuart Platform, similar to the one used by Youtuber Stuff Made Here in his video "Automatic pool stick vs. strangers."

Figure 65: Automatic Pool Stick (Wighton, 2021)

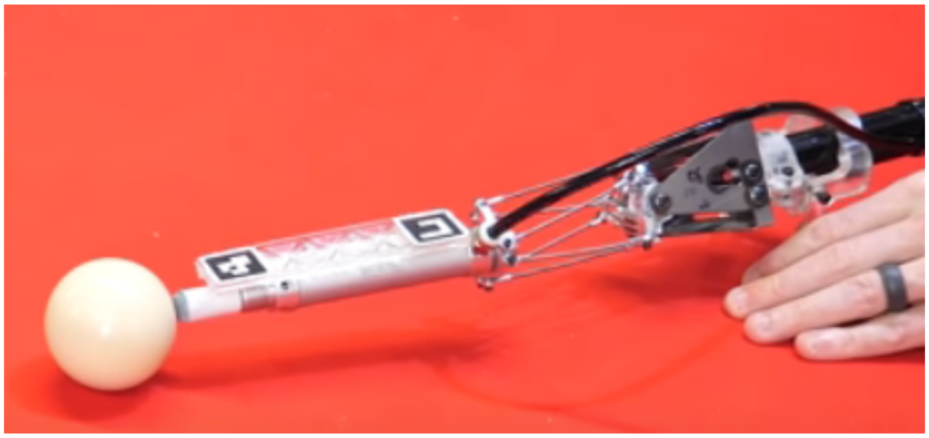


Figure 66: Automatic Pool Stick Utilizing Stuart Platform (Wighton, 2021)

Wighton used the Stuart Platform to aim a pool cue and hit billiard balls extremely precisely. Our version would not need to be as compact, but would have to be more robust to survive the harsh underwater environment. It would also have to allow the spear to slide through it while still being able to hold it for aiming purposes. This would likely be possible if it was mounted to the top of the main part of the spear mechanism so it could aim the spear from the top rather than from the back as Wighton does in his pool stick design.

In addition, it may be prudent in the long run to create a sealed version of the spear mechanism gearbox to prolong the life of the gearbox and add extra protection to the gears and electronics that drive it. This would likely be made possible by rotary seal bearings, which may be prohibitively expensive for a project such as ours. Another option, if issues arise with the corrosion resistance of the current gearbox components, would be to find gears and shafts of different materials, such as acetal plastic, and replace the current system with those. This may be more difficult, as creating a custom shifting gearbox out of plastic is not a common application and it may be difficult to source COTS parts for such a mechanism. However, it may be possible to have such parts custom made or machined if the situation calls for it.

Finally, we think it would be beneficial to add either a proximity sensor and a steel washer to the spear to prevent the gearbox from being drawn too far back, or an absolute encoder to keep track of the spear position. We think this would be more reliable than using time-based controls to autonomously fire the spear. This year, we had simply hoped to control the spear manually, but due to administrative issues unfortunately were not able to finish gearbox assembly.

5.2 Spear Gearbox

While the spear gearbox is fully designed, we were unfortunately unable to obtain the parts for assembly before the end of the year. We placed the order well in advance, but unfortunately administrative issues prevented us from obtaining the parts on time. However, the following includes detailed instruction on the assembly and small amount of manufacturing necessary to assemble this gearbox. First the spacers and shafts must be cut and tapped out of ThunderHex, a specialty hex shaft from VexPro.

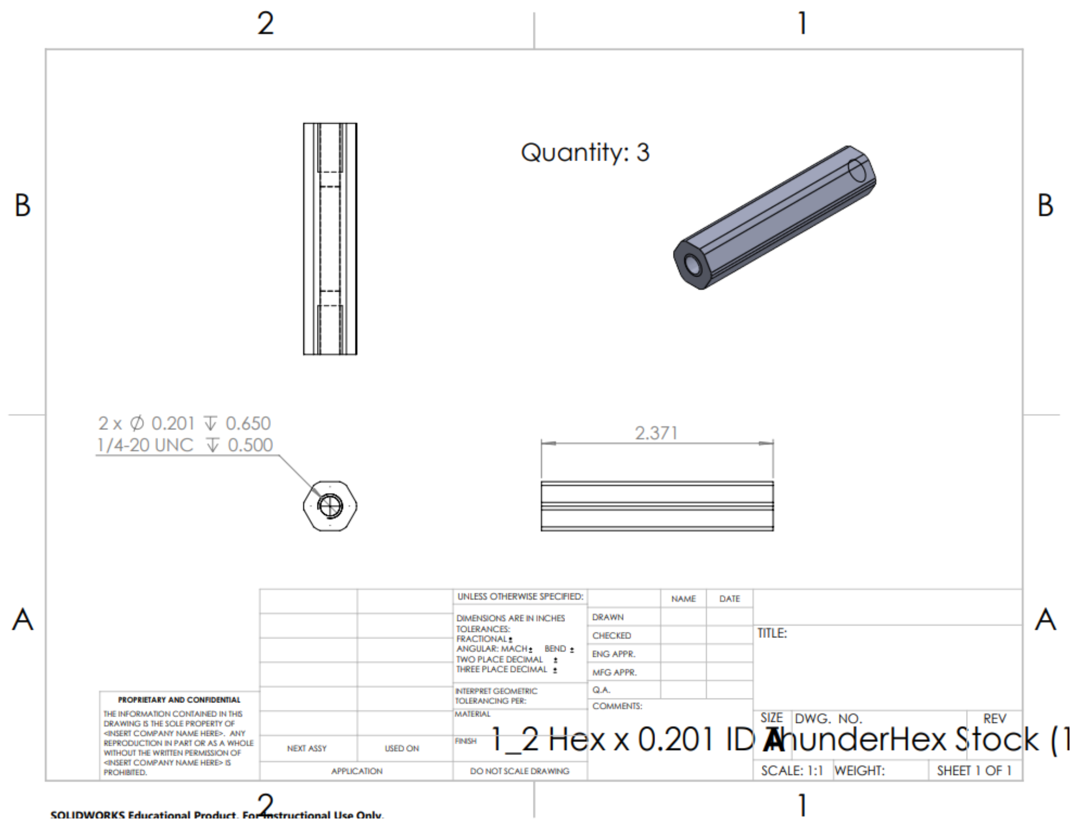

Quantity: 3
2 x ∅ 0.201 ↧ 0.650
1/4-20 UNC ↧ 0.500
2.371
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
DWG. NO.
REV
A
1_2 Hex x 0.201 ID ThunderHex Stock (1
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
SOLIDWORKS Educational Product. For Instructional Use Only.

Figure 67: ThunderHex Axle Drawing

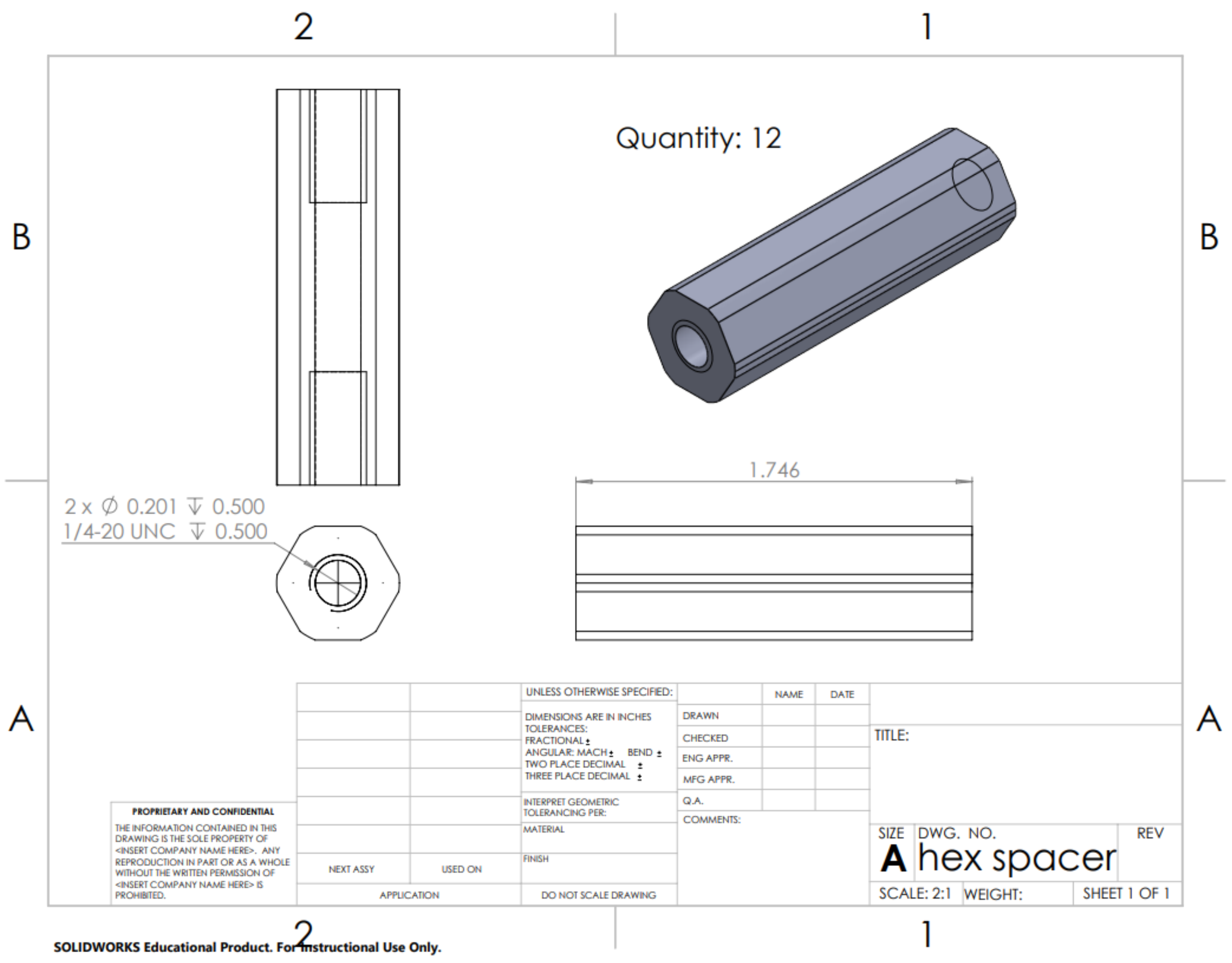

Quantity: 12
1.746
2 x ∅ 0.201 ↧ 0.500
1/4-20 UNC ↧ 0.500
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE
DWG. NO.
REV
A
hex spacer
SCALE: 2:1
WEIGHT:
SHEET 1 OF 1
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
SOLIDWORKS Educational Product. For Instructional Use Only.

Figure 68: Gearbox Hex Spacers Drawing

Additionally, **ONE** of the two shifting shafts must be shortened to fit in the gearbox.

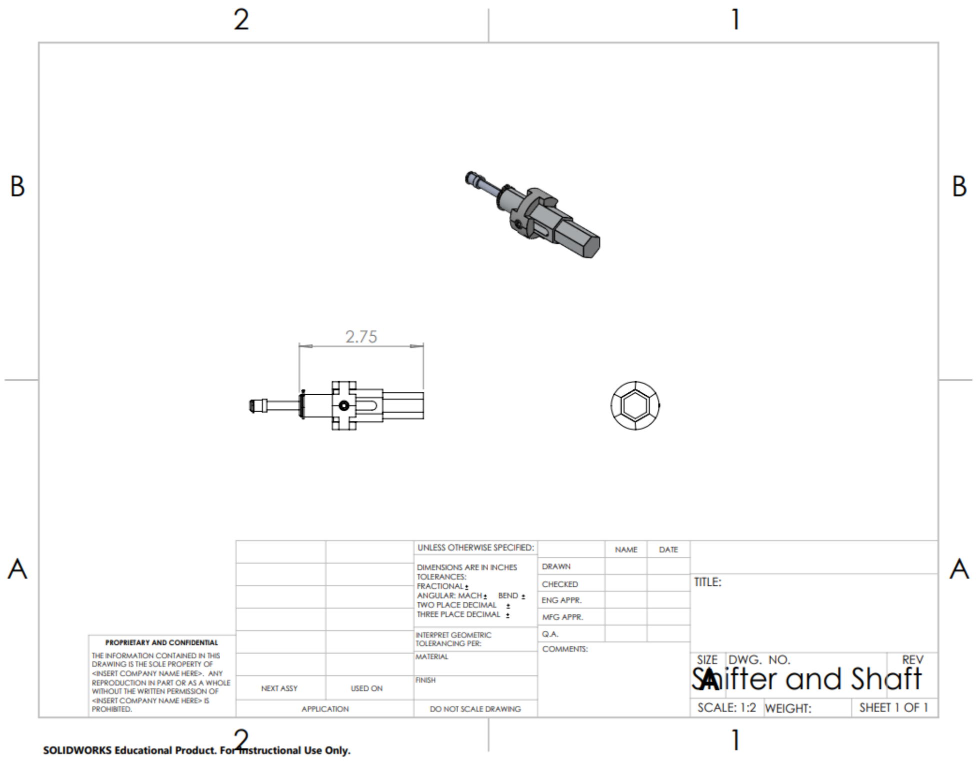

2.75
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
PROPRIETARY AND CONFIDENTIAL
TITLE:
Shifter and Shaft
SIZE A
SCALE: 1:2
SHEET 1 OF 1
SOLIDWORKS Educational Product. For Instructional Use Only.

Figure 69: Drawing of Shifting Shaft

Once these pieces are manufactured, the gearbox assembly is relatively straightforward. First, the motor must be attached to the outer attachment. These materials have already been obtained, so this step has been done already.

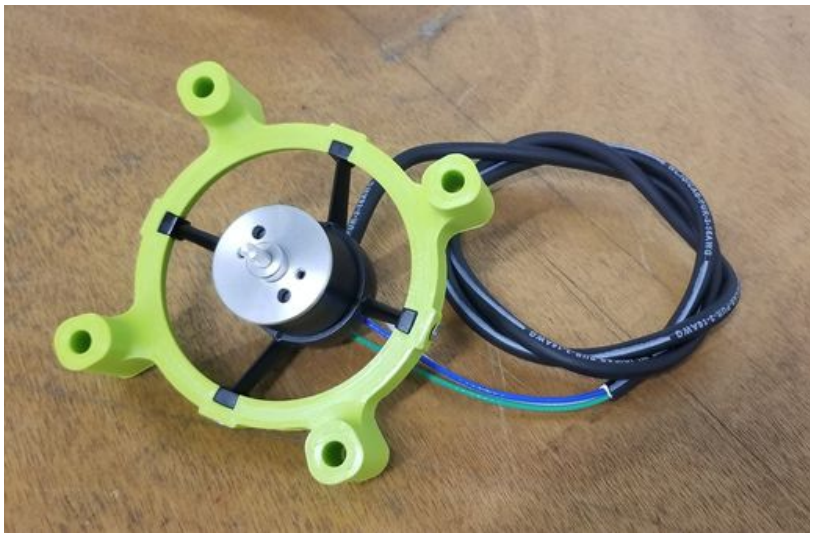
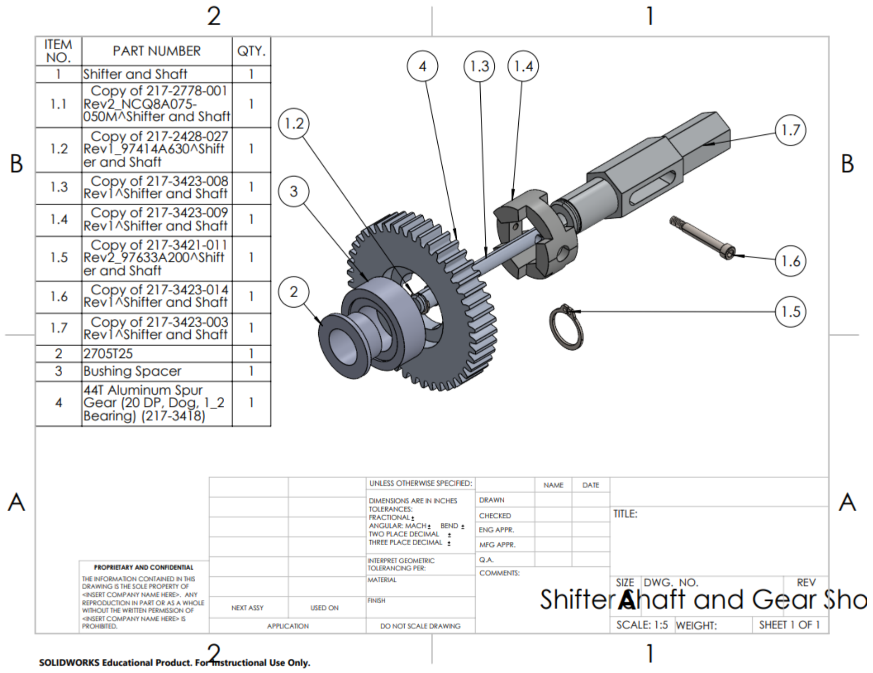


Figure 70: Motor and Motor Mount Assembly

Next, insert the bushings for the axles into their respective holes on the gearbox plate as shown in the drawing. The ThunderHex bushings are matched with the ThunderHex axles, and the bearings that match the hole size for the shifting axles should be inserted into the remaining bearing holes. Then, attach the hex hub to the gearbox plate using 8-32 nuts and bolts, or tap the hex hub holes using a 10-32 tap and attach it using 10-32 screws. Next, insert the three small thrust bearings onto the shaft collar of the motor and press the 8T Steel Spur Gear onto the shaft of the motor. This can be done using an arbor press. Once this subassembly is complete, it can be attached to the gearbox plate using four ⅝ in long ¼-20 bolts and 4 of the Hex Shaft Spacers. Next, insert a ¼-20 bolt and washer into one end of each of the three ThunderHex Axles. Insert the axles through the ThunderHex bushings. Then, on the axle closest to the motor, insert a 1/16-inch spacer. On the middle axle, insert a ½ inch spacer and an ⅛ inch spacer. On the final remaining axle, insert a ⅛ inch spacer and a 1/32-inch spacer. Next, assemble both shifting axle assemblies. Insert the shifting axle into the main axle, slide the dog shifter onto the axle and insert the screw through it such that it slides through the hole at the bottom of the shifting axle. Then, press the bushing and bushing spacer for the shifting axles into the dog shifting gears and slide them onto the shaft. Now, the shafts can be inserted into the bushings in the gearbox plate and the e-clip can be inserted into the slot on the end of the axle. This video clip shows the assembly process (though the e-clip should be added after it is attached to the gearbox plate): https://youtu.be/gqlKYGbRE2M. This video clip shows the same process for the longer axle: https://youtu.be/O7_QNIvSLk4.



ITEM NO.	PART NUMBER	QTY.
1	Shifter and Shaft	1
1.1	Copy of 217-2778-001 Rev2_NCQ8A075-050M^Shifter and Shaft	1
1.2	Copy of 217-2428-027 Rev1_97414A630^Shifter and Shaft	1
1.3	Copy of 217-3423-008 Rev1^Shifter and Shaft	1
1.4	Copy of 217-3423-009 Rev1^Shifter and Shaft	1
1.5	Copy of 217-3421-011 Rev2_97633A200^Shifter and Shaft	1
1.6	Copy of 217-3423-014 Rev1^Shifter and Shaft	1
1.7	Copy of 217-3423-003 Rev1^Shifter and Shaft	1
2	2705T25	1
3	Bushing Spacer	1
4	44T Aluminum Spur Gear (20 DP, Dog, 1_2 Bearing) (217-3418)	1

2
1
B
A
4
1.3
1.4
1.2
1.7
3
1.6
2
1.5
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE A
DWG. NO.
REV
Shifter Shaft and Gear Sho
SCALE: 1:5
WEIGHT:
SHEET 1 OF 1
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
SOLIDWORKS Educational Product. For Instructional Use Only.

Figure 71: Exploded View of Shifting Axle Assembly

Next, put the 40T aluminum gear on the third shaft and the 40T steel gear on the first shaft. Then, put the 60T aluminum gear on the center shaft. The next step is to put a 1/16-inch spacer on the first and second shafts, and a ½ inch and 1/32-inch spacer on the third shaft. Now, put the 20T gear on shaft one, the 30T gear on shaft two, and the 60T gear on shaft three. Then, put an ⅛ inch spacer and a ½ inch spacer on shaft one, and a 1/16-inch spacer on shaft 2 and 3. Next, attach all the remaining Hex Spacers to the assembled half of the gearbox as shown in Figure 72 using ⅝ inch ¼-20 screws.

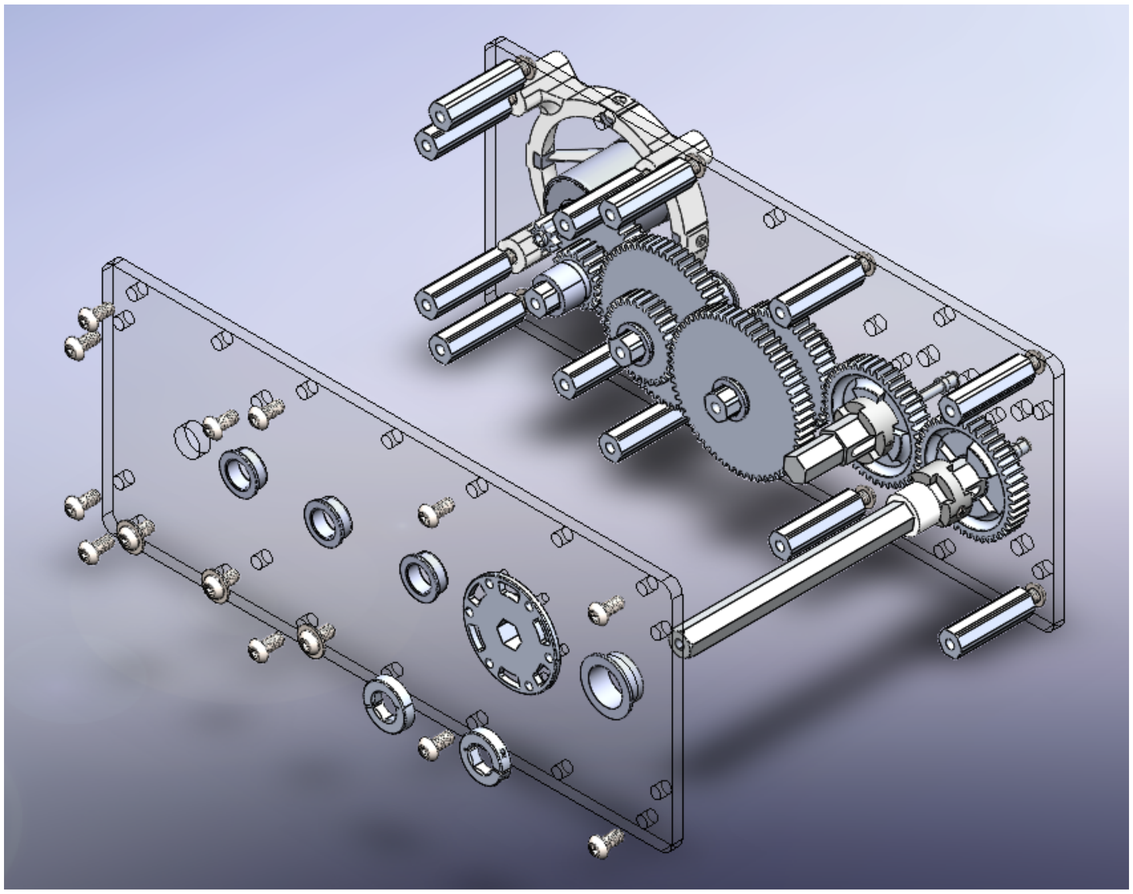


Figure 72: Partially Assembled Gearbox

Next, slide the Special Half Inch Spacer onto the long shifter shaft. Then, line up the bushings and axles on the second gearbox plate with the installed axles and slide the plate onto the axles, securing them with ⅝ inch ¼-20 bolts, making sure to install washers on the bolts inserted into the axles. Finally, attach the shaft collars onto the remaining shifter shafts and tighten them such that the axles will not move forwards and backwards in the gearbox. The following video shows the full gearbox assembly process: https://youtu.be/xq8wbnR9Udc.

All that's left to do now is install the spear mechanism on the output shaft and attach the base and gearbox connector to the gearbox plate by replacing the ⅝ inch ¼-20 screws that line up to the holes in the top of the attachment with longer screws to accommodate the attachment. A third hex spacer can be manufactured and added if necessary for added structural support for the attachment. The holes on the top of the attachment line up with holes on the bottom of the robot that can be widened to have 10-32 bolts installed in them to secure the spear mechanism. The following video shows this process: https://youtu.be/7vYIgu3wqNQ.

Once the spear mechanism and gearbox are fully assembled, preliminary testing can be done. The motor can be hooked up to a Blue Robotics ESC and a power source to be controlled and tested. The next step is to attempt to control the spear manually under load and use force sensors to determine how much force is needed to shift the gearbox when it is under load. These types of gearboxes are known to struggle to shift when there is a high load on them, but it is unclear how much load they can handle with little to no issue. For this reason, we wanted to test the gearbox's capabilities before designing the shifting linkage that will be replacing what would normally be a pancake cylinder. If the force needed to shift the gearbox under a static load is unreasonably high, further tests should be run in which the gear position is rotated slightly, and the gearbox is shifted as the adjustment takes place. This is a common method used in high load robotics applications. While it is not quite ideal, it is generally effective and a simple software solution. Once the amount of force necessary to shift the gearbox is determined, a waterproof servo or stepper motor capable of providing the correct amount of force can be identified and purchased, and the linkage can be created. The linkage style we suggest taking inspiration from is one demonstrated in the video "Do-it-yourself linear servo conversion" by RCSubGuy. He created a linear linkage on a servo using a linear slide and a free spinning linear bearing on top of a servo horn to create a linear linkage that was controlled by a small servo motor. The version for the gearbox would likely need to be a bit more robust, but this design concept is ideal for our purposes.

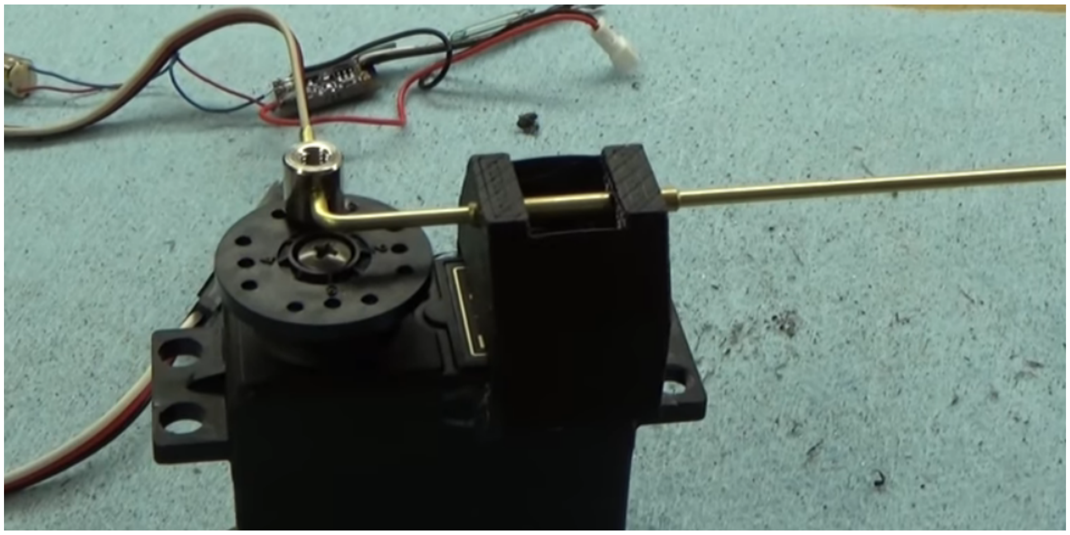


Figure 73: Linear Servo Conversion Linkage

5.3 Containment Unit

We highly recommend that future teams improve upon or investigate alternatives to the containment unit we began designing this year. If a future team agrees that it would be worthwhile to prototype, we suggest doing so to test the dynamics of such a system in the real world and determine the best way to move the lionfish from the entry point to the LCUs on either side. As mentioned previously, two of the ideas we had come up with were the linear actuators and the thrusters. It may be beneficial for a future team to investigate the benefits and risks of using thrusters or linear actuators, and to investigate potential problems either solution might run into that may have been unforeseeable in CAD. We also recommend testing different types of one-way valves, as normally the spears are pushed into the one-way valves rather than being pulled into them backwards. This could risk removing the lionfish from the spear, but this was not something we were able to test this year. However, there may be similar, more effective solutions to this problem that a future team may be able to identify.

5.4 Computer Vision and Object Detection

Since we were not able to test our object detection program with the live camera feed from the robot's top-module camera, we strongly recommend future teams to run the live camera feed from the robot to the object detection program. As the top-module camera is accessed via IP address, OpenCV has an in-built feature where we can pass in the IP address, the username and the password as the input argument to connect with the camera. More information on how to access the camera is provided in the article “Access IP Camera in Python OpenCV” by Stack Overflow.

Although we were successful in detecting Lionfish, the trained ML model is still far from perfect. There are many ways of further improving the current ML model to increase the detection accuracy. One way to improve the ML model is using Supervised pre-training. We recommend using COCO Pets dataset for Supervised pre-training as the model has already been trained on much larger training data and we can use the pre-trained neural network to further increase the accuracy of the model. By using the pre-trained neural network, we just need to train the final layer of the model to accurately detect Lionfish and person. This would reduce the overall train time.

Finally, we recommend the future teams to construct a centroid of the detected object. This would aid the spear mechanism as it would provide the target coordinates and significantly reduce any misfires. Given the coordinates of the bounding box, we can construct the diagonal lines and find the point where the lines intersect.

Citations

Access IP camera in Python opencv. Stack Overflow. (2018, January 1). Retrieved April 26, 2022, from https://stackoverflow.com/questions/49978705/access-ip-camera-in-python-opencv

Wurzbacher, J. *"The Lionfish Invasion"* Sailors for the Sea, Sep-2011. [Online]. Available: https://www.sailorsforthesea.org/programs/ocean-watch/lionfish-invasion. Accessed [10/6/2021].

Bottentus, Tyler *"Eradicating Lionfish"* Sailors for the Sea, Feb-2017. [Online]. Available: https://sailorsforthesea.org/programs/ocean-watch/eradicating-lionfish. Accessed [10/8/2021].

Blue Robotics Inc, "BLUEROV2," *Blue Robotics*, 11-Oct-2021. [Online]. Available: https://bluerobotics.com/store/rov/bluerov2/. [Accessed: 11-Oct-2021].

Busiello, Antonio. "Inside Look: Training Sharks to Eat Lionfish." *Underwater Photography Guide*, https://www.uwphotographyguide.com/sharks-eat-invasive-lionfish#:~:text=On%20the%20Island%20of%20Roatan,to%20protect%20the%20beautiful%20predators.

Seier, K. (2021, May 5). *The value of Computer Vision: More than meets the eye*. Insight. Retrieved October 9, 2021, from https://www.insight.com/en_US/content-and-resources/tech-journal/spring-2021/the-value-of-computer-vision--more-than-meets-the-eye.html.

Brownlee, J. (2019, July 5). *A gentle introduction to Computer Vision*. A Gentle Introduction to Computer Vision. Retrieved October 9, 2021, from https://machinelearningmastery.com/what-is-computer-vision/.

Côté, I. M., Green, S. J., & Hixon, M. A. (2013, June 6). *Predatory fish invaders: Insights from Indo-Pacific lionfish in the western Atlantic and Caribbean*. Biological Conservation. Retrieved September 20, 2021, from https://www.sciencedirect.com/science/article/pii/S0006320713001171.

"The Science." *Texas Lionfish Control Unit*, 26 July 2019, https://texaslionfish.org/the-science/#:~:text=Natural%20Predators,their%20native%20ranges%20as%20well.

SEACSUB S.p.A., "SEAC Polpone Sling Speargun, 50cm, 50 cm," Amazon.com. https://www.amazon.com/SEAC-Polpone-Sling-Speargun-50cm/dp/B00C2SHSLW/ref=sr_1_1?dchild=1&keywords=seac+polpone+sling+speargun&qid=1634238001&sr=8-1.

A-jiou, "A-jiou Fishing Pole Spear Trigger Glass Fiber 5.5' Travel 3 Pieces Hawaiian Sling with 3 Prong Tip and Bag," Amazon.com. https://www.amazon.com/jiou-Fishing-Trigger-Travel-Hawaiian/dp/B08DRH3H74/ref=pd_ybh_a_210?_encoding=UTF8&psc=1&refRID=ZNDPXZGYA69SB9QXAPH9.

Divers Direct, “Hammerhead Hawaiian Sling Spearfishing Combo Kit,” Diversdirect.com. https://www.diversdirect.com/p/hammerhead-hawaiian-sling-combo-kit.

Lionfishhunting.com, “ 30’ fixed barbed paralyzer tip fiberglass polespear,” Lionfishhunting.com. https://www.lionfishhunting.com/best-lionfish-polespears-c-6/30-fixed-barbed-paralyzer-tip-fiberglass-polespear-p-42.html#.YWjq39rMJPY.

Cressi, “Cressi Apache Aluminum Speargun Lion-Fish Edition W/Stainless Steel Shaft/Sling,” Europeanoutdoors.com. https://www.europeanoutdoors.com/product/P-ZFE3510-PAR/Cressi-Apache-Aluminum-Speargun-Lion-Fish-Edition-W~2FStainless-Steel-Shaft~2FSling.

Stuff Made Here, Automatic Pool Stick vs. Strangers, 2021. Accessed on: Sept. 7, 2021. [Video]. Available: YouTube. https://www.youtube.com/watch?v=vsTTXYxydOE.

RCSubGuy, Do-it-yourself linear servo conversion, 2018. Accessed on: March 14, 2022. [Video]. Available: YouTube. https://www.youtube.com/watch?v=pClmobqMa-A.

Dixie Divers Products, Seac Polpone Speargun, 2014. Accessed on: Sept. 7, 2021. [Video]. Available: YouTube. https://www.youtube.com/watch?v=vZONPS1dXcc.

MATLAB, What Is Autonomous Navigation? The MathWorks, Inc., 2020. Accessed on: October 15, 2021. [Video]. YouTube. https://www.youtube.com/watch?v=Fw8JQ5Q-ZwU.

Dr. D. Lu, “Eight degrees of difficulty for autonomous navigation,” PickNik, 04-Dec-2020. [Online]. Available: https://picknik.ai/ros/navigation/2020/12/04/navigation.html. [Accessed: 15-Oct-2021].

M. Abadjiev, Q. Chen, C. Ewen, N. Johnson, N. Olgado, H. Saperstein, O. Strickland, and C. Whimpenny, WPI, Worcester, MA, rep., 2020.

J. Lombardi, A. Yuzvik, N. Uvarov, B. Kelly, and W. Godsey, WPI, Worcester, MA, rep., 2018.

L. Gangaramaney, M. Ferriera, C. Johnson, and L. Go, WPI, Worcester, MA, rep., 2021

“Lionfish hunting, gear, equipment and Techniques,” Lionfish Hunting Lodge. [Online]. Available: https://lionfish.co/lionfish-hunting-2/. [Accessed: 15-Oct-2021].

Lionfish pole spear with paralyzer tip. (n.d.). [Photograph]. https://cdn11.bigcommerce.com/s-hwgrldcncv/images/stencil/original/products/71988/128511/lionfish_pole_spear_with_paralyzer_tip__75410.1629843676.jpg

Lombardi, J., Godsey, W., Kelly, B., Uvarov, N., & Yuzvik, A. (2018, April). *Autonomous Lionfish Harvester*. Worcester Polytechnic Institute. https://digital.wpi.edu/pdfviewer/dv13zw03p

Abadjiev, M., Chen, Q., Ewen, C., Johnson, N., Olgado, N., Saperstein, H., Strickland, O., & Whimpenny, C. (2020, May). *An Autonomous Underwater Lionfish Harvester*. Worcester Polytechnic Institute. https://digital.wpi.edu/pdfviewer/0p096947d

Gangaramane, L., Ferriera, M., Johnson, C., & Go, L. (2021, April). *Lionfish - Phase IV*. Worcester Polytechnic Institute. https://digital.wpi.edu/pdfviewer/cr56n391k

Alexander Vasenin, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons