MATE 2018 ROV COMPETITION

by

Oliver Law
Andrew Miltimore
Jonathan Murphy
Ethan Quisias

A report presented to the

British Columbia Institute of Technology

in partial fulfilment of the requirement for the degree of

Bachelor of Engineering (Mechanical)

Faculty Advisor: Taco Niet

Program Head: Mehrzad Tabatabaian

Burnaby, British Columbia, Canada, 2018

2018 MATE ROV COMPETITION

Deep Blue Marine Engineering

British Columbia Institute of Technology
Vancouver, Canada

INNOVATION FOR A COMPLEX WORLD

Jonathan Murphy – Project Manager
Andrew Miltimore – Lead Designer
Oliver Law – Task Lead
Ethan Quisias – Electronics Lead
Taco Niet – Faculty Advisor

Table of Contents

We hereby declare that we are the sole authors of this report.

Signature

Signature

Signature

Signature

We further authorize the British Columbia Institute of Technology to distribute digitally, or printed paper, copies of this report, in total or in part, at the request of other institutions or individuals for the purpose of scholarly activity.

Signature

Signature

Signature

Signature

ACKNOWLEDGEMENTS

We would like to acknowledge the companies who sponsored us:

- Pacific Fasteners Ltd.: provided us with stainless steel fasteners
- FiberTek: provided us with the foam used for the electrical housing
- Rocky Mountain Motion Control: provided us with aluminum extrusion used in the frame

The materials given to us by these companies were vital in completing our ROV.

We are especially grateful for the assistance, guidance, and support given to us from the following faculty. Our faculty sponsor, Taco Niet, for guiding us in all aspects of the project. Johan Fourie, for project management guidance and approving component purchases. Jason Brett of the Technology Teacher Education program for providing us with expert knowledge in electrical design and robotics. Greg King, Eugene, Dave Lewis, and Ernie Jazen from the CARI building for expert knowledge and advice in mechanical design and manufacturing.

Lastly, we would like to acknowledge the BCIT Mechanical Engineering department for providing us funding for the electrical and mechanical parts needed to complete our capstone project.

ABSTRACT

Deep Blue Marine Engineering (DBME) has developed an ROV that satisifies the design requirements outlined in the RFP submitted by the Applied Physics Laboratory (APL) at the University of Washington. DBME's *Marauder* was designed and built by a team of 4 mechanical engineering students from the British Columbia Institute of Technology (BCIT). *Marauder* was prototyped to perform tasks required for locating and recovering the engine of a vintage airplane, installing a seismometer, and installing a tidal turbine and instrumentation to monitor its marine environment. *Marauder* was intially prototyped at BCIT's Burnaby campus before it underwent thorough and rigirous testing at the BCIT Marine Campus to ensure functionality and reliability when performing the required tasks.

To complete the scope of work provided by the APL at the University of Washington, DBME was organized into mechanical and electrical design teams. A collaborative design approach between mechanical and electrical design teams was used to insure functionality and control of mission specific tooling during prototyping and testing.

Marauder's frame and tooling was precision manufactured using in-house equipment that includes a 3-axis CNC mill, a water jet cutter, and multiple 3D printers to insure component fitment and potential development of multiple prototypes. Furthermore, with the aid of precision manufacturing equipment, *Marauder's* frame and tooling was also designed to meet the minimal size and weight requirements for ease of portability.

The following technical document outlines the design process and results produced by BCIT Deep Blue Marine Engineering during the development and prototyping of *Marauder*.

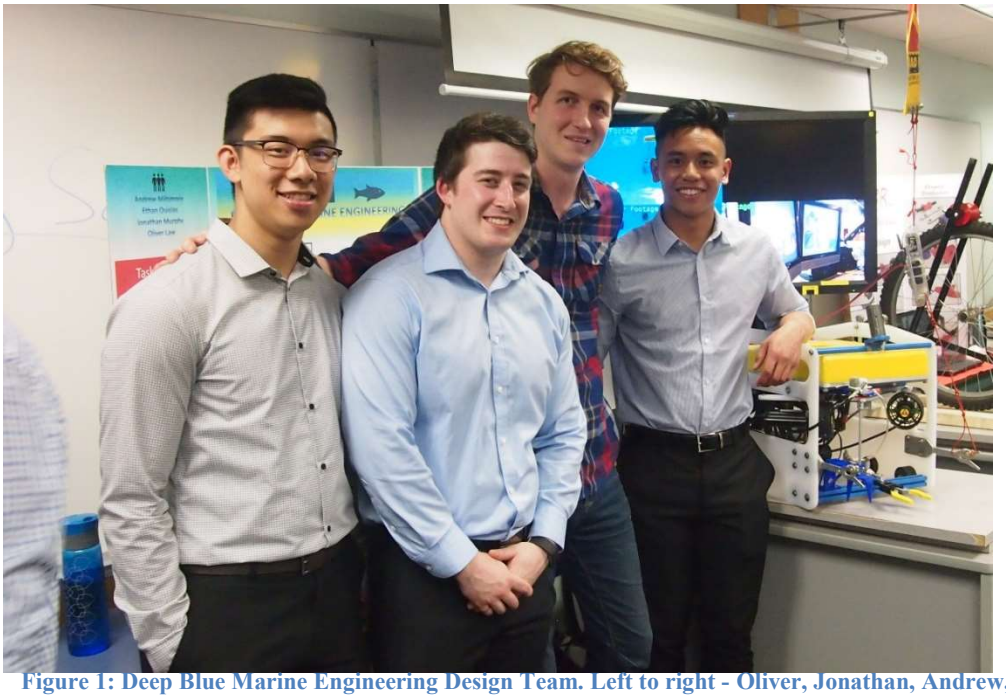


Figure 1: Deep Blue Marine Engineering Design Team. Left to right - Oliver, Jonathan, Andrew, and Ethan

PROJECT MANAGEMENT

Team Member Roles

When selecting the roles for each of our individual team members, it was important that their project roles favored towards their strengths. It was also crucial that each team member had the opportunity to expand and develop their skills and knowledge of electrical and mechanical engineering design. To outline each team member's individual roles and responsibilities, the team developed a responsibility assignment matrix (RAM). Each member's responsibilities were interlinked with other members of the team by assigning either responsibility (R), consult (C), and Inform (I) to team members per activity. Assignment of the activities for each team member was a critical for defining project deadlines in the developed Gantt chart for project time management.

Activity	Jonathan	Oliver	Ethan	Andrew	Taco
Vehicle Frame	R	I	I	C	I
Mission Objectives Design	C	C	C	R	C
Motor Control	C	C	R	C	C
Communication System	I	R	C	I	I
Data Acquisition System	C	C	R	C	C
Video Capture System	I	R	C	I	C
Prototyping	C	C	C	R	I
Operational Testing	R	C	C	I	C
Developmental Testing	R	C	C	C	C
Underwater Testing	C	C	C	R	C
Jobsite safety	R	C	I	I	C
Spare Components	I	C	C	R	I
Graphical User Interface	C	C	R	I	I

Figure 2 - Responsibility Assignment Matrix

Design phases and process

The ROV project consisted of two design phases. Phase one consisted of building the functional ROV which consisted of the mechanical design of the frame and electronics housing, and phase two consisted of designing the mission specific tools. Because this was the first ROV built by the design team, it was important that phase one be completed in a timely manner to confirm that the expected maneuverability and control was adequate. Insuring adequate maneuverability and control from phase 1 was critical to commence phase 2. This is because adequate maneuverability would allow more freedom of design for the mission specific tools. From examining previous competition results, it was apparent that the function and maneuverability developed in phase 1 was much more critical to success than attempting to develop mission specific tools in phase 2 that compensated an ROV with inadequate maneuverability and control. Therefore, a majority of the design team's time was spent designing and building the ROV frame and electronics housing in phase 1. After completion of phase 1, the design team began focusing on phase 2, which consisted of mission specific tool design. To complete each design phase, the design team approached each problem using a collaborative design process.

The design team's collaborative design process shown in Figure 3, consisted of each team member creating conceptual designs and prototypes that could demonstrate potential functionality for both phase 1 and phase 2. After comparing individual design concepts and numerical ranking using a decision matrix, the team would select a concept to continue forward with. A decision matrix was vital when completing phase 1 of the design process because of the numerous configurations of the thrusters and electronics housing design choices available. Utilizing a decision matrix proved useful when designing the mission specific tools because of the numerous ways to complete each task. The design team typically chose designs that could be manufactured rapidly and easily with an expected low risk of failure. By utilizing this collaborative design approach, the design team was skillfully fortunate to avoid many potential catastrophic failures.

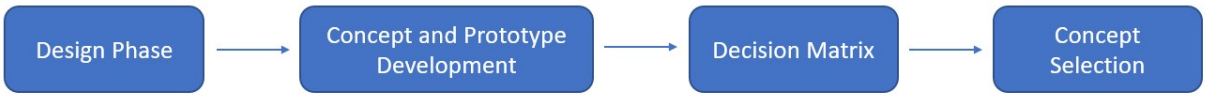


Figure 3 - Collaborative design process

Component implementation and organization

To implement and organize the various components through-out the ROV, the design team used block diagrams. Block diagrams allowed both design teams to identify and diagnose potential errors during implementation and to properly organize the required components for subsystems quickly and effectively. An example block diagram for the wi-fi sensor is pictured in Figure 4.

Figure 4 - Example Block Diagram (Wi-fi Sensor)

Time management

The design team managed time spent on each activity assigned in the responsibility assignment matrix of the project using a Gantt chart. The Gantt chart was useful to ensure the both adequate time was spent on each task and to ensure that no time was being wasted. By allocating time effectively, the design team anticipated completion by May 9th, 2018. However, the design team completed the defined scope early on May 4th, 2018. The design team also tracked progress and completion of critical objectives using a milestone schedule.

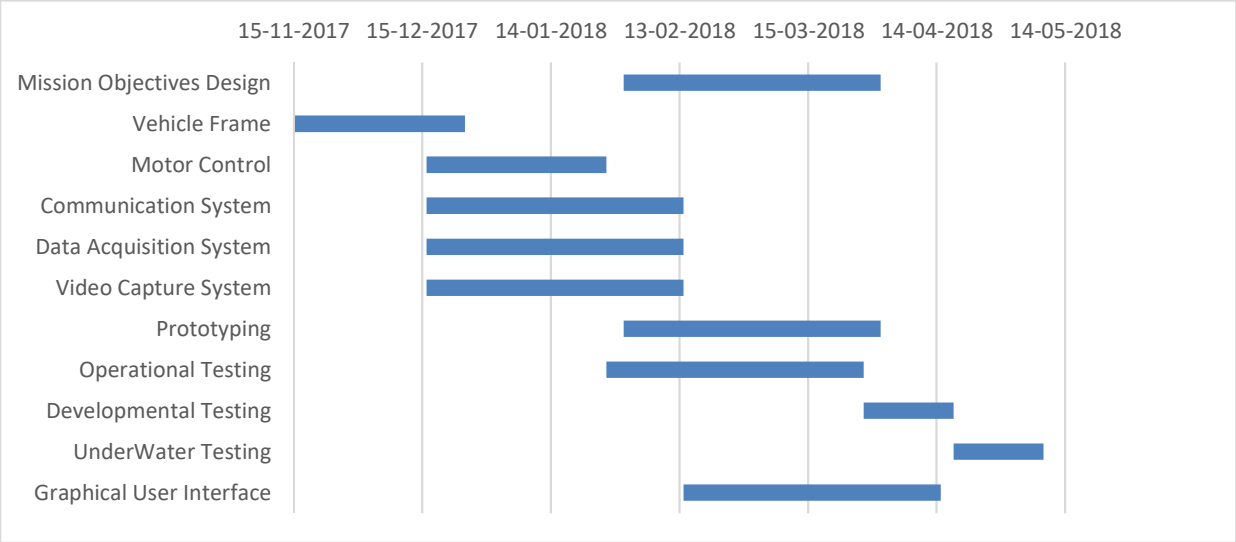


Figure 5 - Gantt Chart for time management

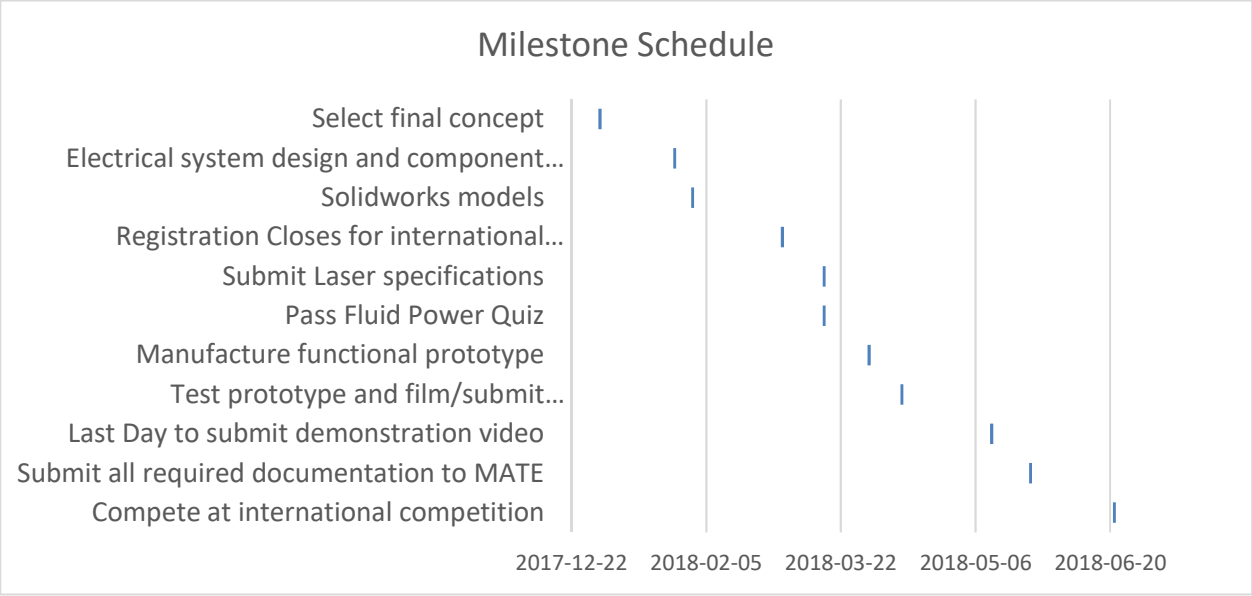


Figure 6 - Milestone schedule for objective management

ROV Mechanical Design Rationale

Frame

The frame as seen in Figure 7, is designed in such a way to provide both adaptability and ease of manufacturing. To ensure the frame is adaptable to many different setups of tooling the team uses t-slots which allow positioning of components anywhere along them on all four sides. The t-slots in combination with the plethora of mounting holes on the side panels as shown in Figure 7 provide endless mounting possibilities for tooling. The frame is also designed in such a way that any 2-dimensional cutting machines such as a CNC router or waterjet would be able to easily machine the profiles. The frame also includes mounting holes on the side panels for additional ballast weights to accommodate potential inaccuracy in determining the total buoyancy force and the buoyancy center of the ROV. In addition to correcting extbuoyancy, the added ballast weights lower the ROV's center of mass further below its buoyancy center, thus improving stability.

Figure 7 - ROV Frame

Electronics Housing

The electrical housing as seen in Figure 8 consists of three main parts: a casted polyurethane foam cover (shown below), a set of ¼" aluminum rings and a 6mm Lexan bottom piece. The cast polyurethane foam prior to machining to size is pictured in Figure 9. There will also be an assortment of intelligently spaced bolts to compress an O-ring that lives between the Lexan and aluminum to seal against water and to provide a consistent and easily replaceable seal. To seal the polyurethane foam to the aluminum the team used a 2-part marine epoxy. Both the Lexan cover and the aluminum rings are cut on a waterjet cutter with the aluminum rings receiving a CNC milled O-ring groove. The O-ring groove has been slightly undersized for a 3mm O-ring to ensure good contact with the Lexan. This foam housing also doubles as the main source of buoyancy for the ROV.

Figure 9 - Foam casting before machining

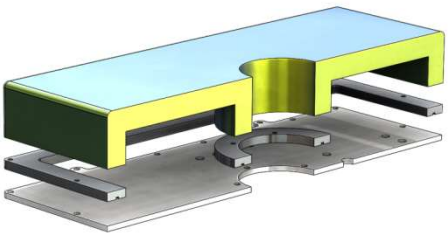


Figure 8 - Electronics Housing Section View

Mission Specific Tooling Design Rationale

Task 1- Tool Integration

To accomplish task 1 the ROV must lift debris weighing about 40 N in water off of a platform and then release it via Bluetooth. To accomplish this, the ROV utilizes the 3 separate tools. These 3 components work in conjunction to grab, lift and release debris wirelessly. Combined, they are what make up DBME's lift bag release mechanism.

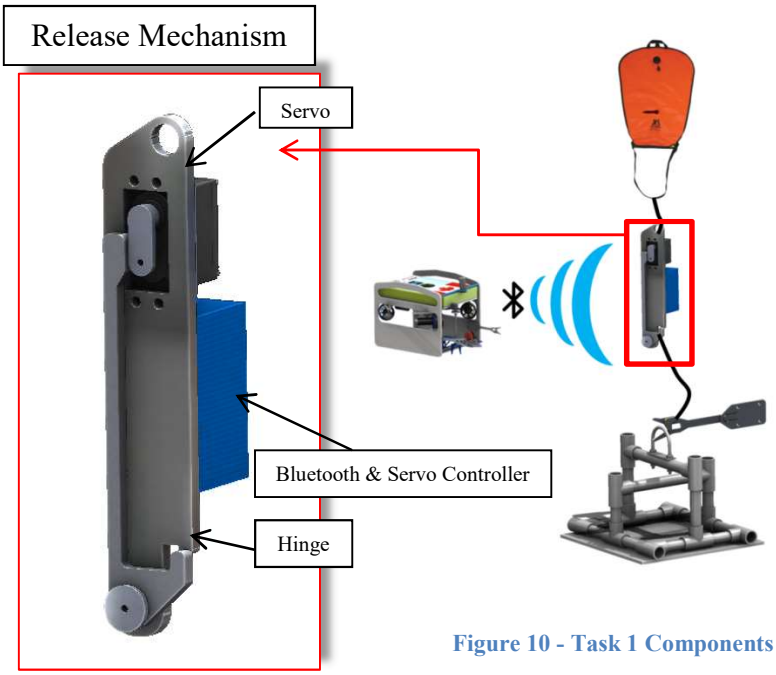


Figure 10 - Task 1 Components

Task 1 – Lift Bag

This lift bag is a simple diver's lift bag that inflates and allows it to lift very heavy objects underwater. The design team found an old lift bag and then modified it to displace the correct amount of water to give us the target lifting force of just under 40 newton's. The lift bag is simply inflated via the pneumatic system and a 2/2 valve plumbed into the bag.

Figure 11 – Lift Bag

Task 1 – Release Mechanism

The release mechanism as seen in Figure 12 uses a lever arm to allow a small servo to release an object weighing well of 40 newton's. The servo is controlled by an onboard Arduino which has a Bluetooth board connected to it. The Bluetooth board is constantly looking to connect to the ROV which has a similar Bluetooth board. Once connected the ROV can trigger the servo to release the weight. This design is used because it allowed easy integration with the onboard electronics is reliable and simply ties inline between the grabber and lift bag, so only one design is need for the grabber to also accomplish the second half of Task 1.

Figure 12 - Release Mechanism

Task 1 - Grabber

The grabber as seen in Figure 13 is a one-way latch that can be driven into a U-bolt to connect it. Then a tapered sleeve cut into the aluminum holder will release once the ROV reverses. This is an extremely simple design that has proven to be very effective at semi-permanently attaching itself to a U-bolt.

Figure 13 - Grabber

Task 2 - Ocean Bottom Seismometer Inductive Coupling

The inductive coupling contained three vital components for successful operation. First, an adapter for the OBS was designed and 3D printed using PLA as seen in Figure 14. The OBS adapter contains the inductive power transmitter which was potted in marine epoxy for extra weight and waterproofing. The housing for the batter housing was then made using 2-inch PVC pipe with a screw on end-cap. The screw on end-cap was chosen to allow for easy replacement of the 9V and AA batteries.

Figure 14 - Inductive coupling

Task 2 - Ocean Bottom Seismometer Tee Manipulator

The tee manipulator, as seen in Figure 15, consisted of three vital components. The waterproof servo, the mounting bracket, and the end-effector. Before water-proofing the servo, the feedback potentiometer was removed and replaced with 2 equal resistors to allow for continuous rotation in both directions. The hobby servo was waterproofed using marine epoxy and placing an o-ring at the output shaft. After submerging the servo overnight and then performing a megaohm test, the servo was deemed water proof after reading greater than 10 Mega ohms. The mounting bracket was 3D printed using PLA and designed to provide adequate support as a cantilever beam and provide torsional rigidity. The end-effector was designed with four prongs to allow easy positioning and was mounted on an extended shaft to allow for better viewing from the bird's eye and forward-facing cameras. Because of their complex shape, the four-pronged end effector and mounting bracket were 3D printed using PLA.

Figure 15 - Tee Manipulator

Task 2 – Ocean Bottom Seismometer Wi-fi Receiver

The wi-fi receiver to recover seismometer data from the OBS was developed using the ESP8266-01. This wi-fi board was used because of it's functionality as a wi-fi client as well as it's compact size and weight. The ESP8266-01 was configured as a wi-fi client and connected to the Arduino Mega on board the ROV. The wi-fi sensor was mounted to the gripper to have closer proximity to the OBS because of the limited range of wi-fi underwater. The wi-fi sensor was waterproofed using epoxy that cured transparent so that the ESP8266-01's built in LEDs could still be viewed for trouble shooting (Figure 16). After testing, this device was found to be relatively unreliable. On the first attempt during testing, the firmware on board the wi-fi sensor ceased to function, making the device unusable. There was no apparent reason as to why this occurred because the device had not been touched since it had last been functioning. Another wi-fi sensor was remade to rectify this issue.

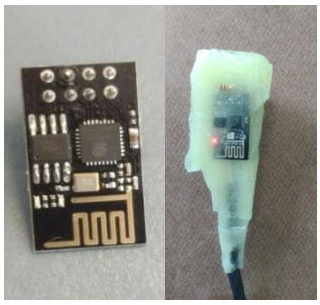


Figure 16 - Wi-fi Sensor before and after waterproofing

Task 3 - Pneumatic Gripper

Task 3 mainly consists of gripping components and moving them to designated areas or inserting them into other apparatuses. This includes installing a tidal turbine in the optimum location, an Intelligent Adaptable Monitoring Package (I-AMP), placing a mooring a given distance from the tidal turbine, and suspending an Acoustic Doppler Velocimeter at a given height on the mooring. With regards to Task 1, a pneumatic system was already in place on the ROV and adding an additional line to manipulate the gripper was the easier option rather than going through the process of waterproofing a motor. The gripper (Figure 17) was designed such that it can grip the maximum diameter of PVC pipe which is 2 inches.

Figure 17: Pneumatic Gripper

Task 3 - Acoustic Doppler Velocimeter

The last part of Task 3 consists of suspending an Acoustic Doppler Velocimeter (ADV) at a given height on the mooring. DBME was tasked with designing its own ADV to simulate the equipment. From the specifications given by MATE, the ADV was created as shown in Figure 18. Sticking with a simple design to minimize manufacturing work, the ADV was created with a two-inch PVC pipe, compatible with the maximum diameter the pneumatic gripper can handle, and a metal hook on the top to hook onto the #310 U-bolt. To counter-act the buoyancy of PVC, holes were drilled into the ADV to allow water to flood inside and caused it to become negatively buoyant. To ensure that the ADV does not slip through the gripper's claws, it was wrapped in duct tape.

Figure 18: Acoustic Doppler Velocimeter

Task 3 - Measuring Tape

To measure the horizontal distance from the tidal turbine, DBME fashioned a measuring tape on the front of the ROV. This apparatus is pictured in Figure 19. Through cooperation with the camera angles, the pilot will be able to hook onto the PVC pipe with a ring installed at the end of the tape, and drive backwards to measure distance. This simplistic design does not further complicate the electrical system as it only relies on the cameras to read horizontal distance and the pilot to be able to retract the measuring tape.

Figure 19: Measuring Tape

Electrical System Design rationale

Control System

The ROV's control system consists of the communication, motion control and feedback, peripheral control subsystems. Communication will be handled via a full duplex RS485 serial communication protocol circuit between two Arduino MEGA 2560 microcontrollers and two MAX488 integrated circuit chips. Motion control and feedback consisted of five thrusters, an inertial measurement unit, and an absolute pressure sensor to measure depth. Lastly, the peripheral control subsystem will control the pneumatic solenoids for the gripper and airbags.

Communication Subsystem

The communication subsystem has two Arduino MEGA 2560 microcontrollers communicating via full duplex RS485 serial communication through a CAT-5 cable through the tether. The RS485 serial communication protocol was chosen for its noise cancelling capabilities, its high data transfer rate, and its ease of implementation. This protocol's noise cancelling capabilities stems from the use of a differential signal in which two signals are created from the original communication signal: a non-inverting and an inverting signal. And through the twisted pair of the CAT-5 cable, the induced noise current from the magnetic fields in the environment are in opposite directions, resulting in each of the amplitudes of the noise to be cancelled.

This microcontroller was selected for its ease of implementation and the amount of digital and analog I/O pins. Furthermore, the multiple hardware serial ports allow for communication between the Arduinos as well as with the graphical user interface that will be displayed on a laptop. Although the Arduino IDE has software serial capability, using this feature would increase the chances of miscommunication due to the complexity of code needed compared to hardware serial ports.

Inside the top-side control box, we will be using two two-axis joysticks, one rotational potentiometer, two two-position switches, two-three-position rocker switches, and seven indicator LEDs. The joysticks and potentiometer will be connected to the analog inputs of the microcontroller and will undergo A/D conversion and thrust vectoring to obtain the desired values for motion control. Furthermore, the switches will be connected to digital inputs and will send boolean logic to the ROV for task specific and peripheral controls.

Motion Control and Feedback Subsystem

This subsystem is in charge of linear movement in the x-, y-, and z-axes, rotation around the z-axis, and obtaining information regarding position and orientation for the use of the pilot. To achieve this, the ROV is equipped with five thrusters (four at each corner angled at 45° and one vertical central thruster), an IMU module, and a pressure sensor for determining the depth of the ROV.

The thrusters are mounted at 45° instead of parallel to the x- and y-axes because it allows for more balanced translational and rotational movement. However, the drawback to this method of mounting the thrusters is that the implementation of precise control is more complex. The motors must be thrust vectored to achieve the desired motion, which is the process of vector addition of all the thrusts to achieve an overall thrust in a certain direction.

To implement thrust vectoring, four matrices must be created: $\vec{\tau}$ which contains the duty cycle values from the joysticks in the x-, and y-axes and rotation about the z-axis, $\vec{f}$ which contains the final thrust vectored values, $\vec{Q}$ which contains the coefficients necessary for adjusting each motor's duty cycle for a given angle and radius, and the thrust allocation matrix $\overrightarrow{W}$. When thrust vectoring is completed, the following equations were obtained and then implemented directly into the code:

$$\vec{f} = \frac{1}{4}\overrightarrow{W}\begin{bmatrix} 0 \\ Q^{-1}\vec{\tau} \end{bmatrix} = \frac{1}{4}\begin{bmatrix} \frac{\tau_x}{cos\phi} + \frac{\tau_y}{\sin\phi} + \frac{\tau_{rot}}{r} \\ \frac{\tau_x}{cos\phi} - \frac{\tau_y}{\sin\phi} - \frac{\tau_{rot}}{r} \\ -\frac{\tau_x}{cos\phi} + \frac{\tau_y}{\sin\phi} - \frac{\tau_{rot}}{r} \\ -\frac{\tau_x}{cos\phi} - \frac{\tau_y}{\sin\phi} + \frac{\tau_{rot}}{r} \end{bmatrix}$$

In terms of the feedback portion of this subsystem, the IMU and the pressure sensor give the pilot information about positioning and orientation, which is very useful when the ROV pilot is constrained to only the on-board camera views. The IMU is a 6-axis module that contains a 3-axis gyroscope, and 3-axis accelerometer. The gyroscope will be used to detect any rotational movement and the accelerometer will detect translational movement as well as yaw, pitch, roll movements. Furthermore, the pressure sensor will be calibrated to output data regarding the depth of the ROV. The pressure sensor senses absolute pressure and by the use of careful calibration and simple hydrostatic pressure calculations.

Peripheral Control Subsystem

The *Peripheral Control* subsystem is the control system for task specific functions and will be controlled be the *Communications* subsystem, similar to the *Motion Control and Feedback* subsystem. This subsystem will control the pneumatic gripper, the inflation of the two lift bags, the Bluetooth module used for Task 1, and the servo that will be used to level the ocean-bottom seismometer.

Boolean logic values from the switches on the top-side control box will be used to control the task specific peripherals on the ROV. Two 12V electric solenoid valves will be controlled by the microcontroller and two TIP-122 NPN transistor circuits for the inflation of the lift bags. And similarly, a 24V electric solenoid will be used to control the gripper. Furthermore, the Bluetooth module for Task 1 will be controlled by an Arduino Pro Micro that will turn the servo that releases a latch when a HIGH signal from the main Arduino MEGA microcontroller is transmitted. Lastly, the same Arduino Pro Micro previously mentioned will control a continuous rotation servo motor used to level the ocean-bottom seismometer. The servo will follow the same operational protocol as the Bluetooth module in which a HIGH signal from the Arduino MEGA to the Arduino Pro Micro will determine the direction of rotation of the servo motor.

Camera System

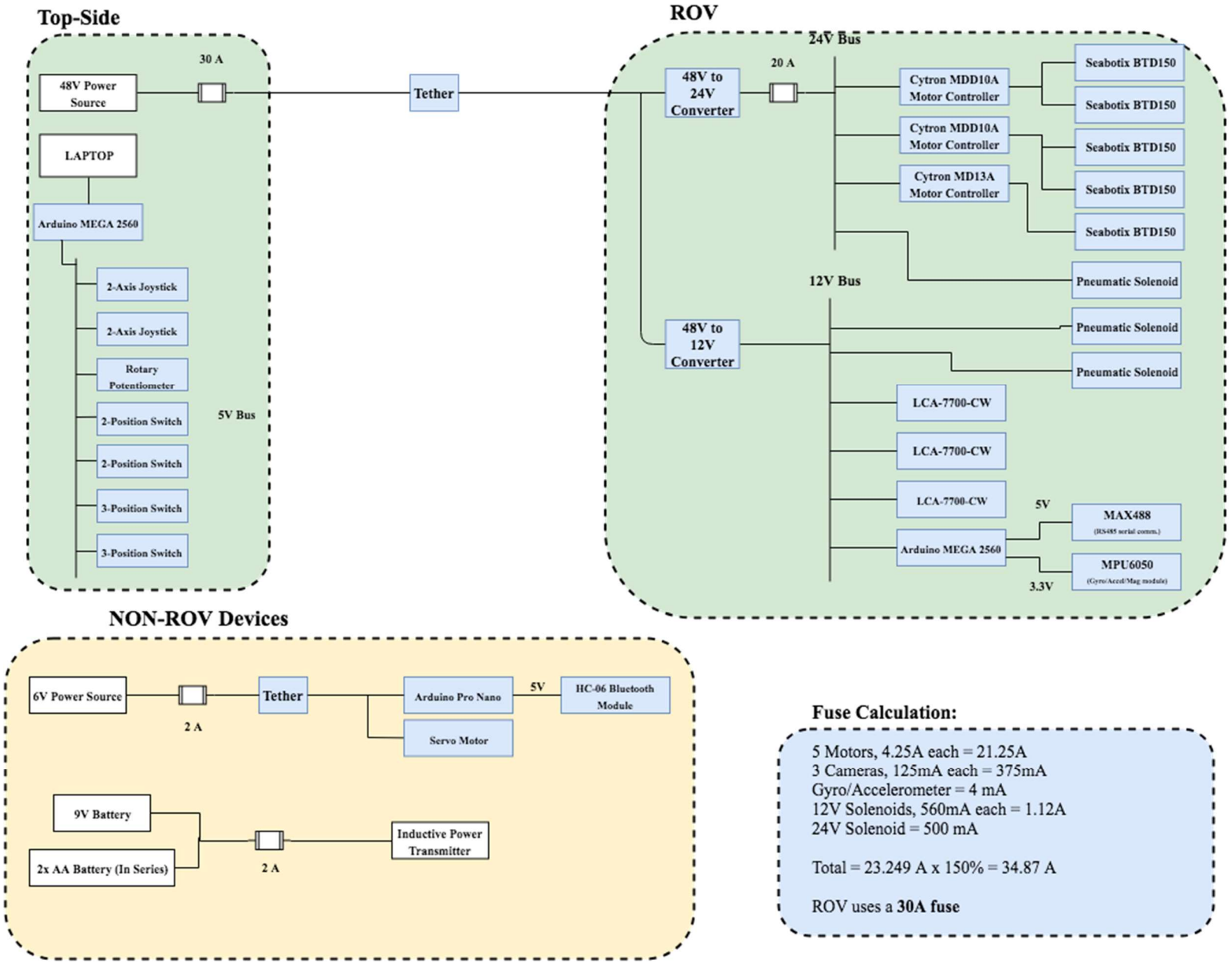
The *Camera System* was kept as an independent system to minimize troubleshooting and complications. Two cameras providing the front view for the ROV were used to incorporate depth perception by crossing views and utilizing the resulting parallax. The third camera provides a top-down view of the task-specific components on the front of the ROV to further aid the driver in accomplishing tasks. To minimize the amount of wires in the tether, a 4-Channel BNC to RJ45 video balun was used to convert up to four video signals to one ethernet cable running up the tether. This design allows for an additional camera to be implemented for future work incase another view was needed for driving or competition purposes. From the ethernet cable running up the tether to the shore, the signal is converted back into separate BNC video signals through another video balun and plugged into their respective monitors with an RCA adapter.

Power System

From the shore, 48VDC at 30A is sent through an in-line 30A fuse, then through the tether and into the ROV which is then distributed into two step-down buck converters, a 24V and 12V converter. These two converters were chosen for their efficiency and added safety precautions that are built in. They use an electrical circuit comprised of inductors and capacitors to step the voltage down whereas the counterpart – linear regulators – step down the voltage by dissipating power as heat, making our converters much more efficient. Furthermore, these converters include safety features such as over-load, current, temperature, and voltage protection, and a water- and moisture proof-design. Following the 24V converter, an in-line 20A fuse is placed for added protection for our five Seabotix BTD150 thrusters. The 12V converter does not have an in-line fuse because it supplies power to easily-replaceable, and inexpensive components; although the initial fuse on the top should be sufficient for any unforeseen problems. The two voltage levels are then distributed into two separate power rails, powering the other subsystems.

SYSTEM INTERGRATION DIAGRAMS

Electrical SID



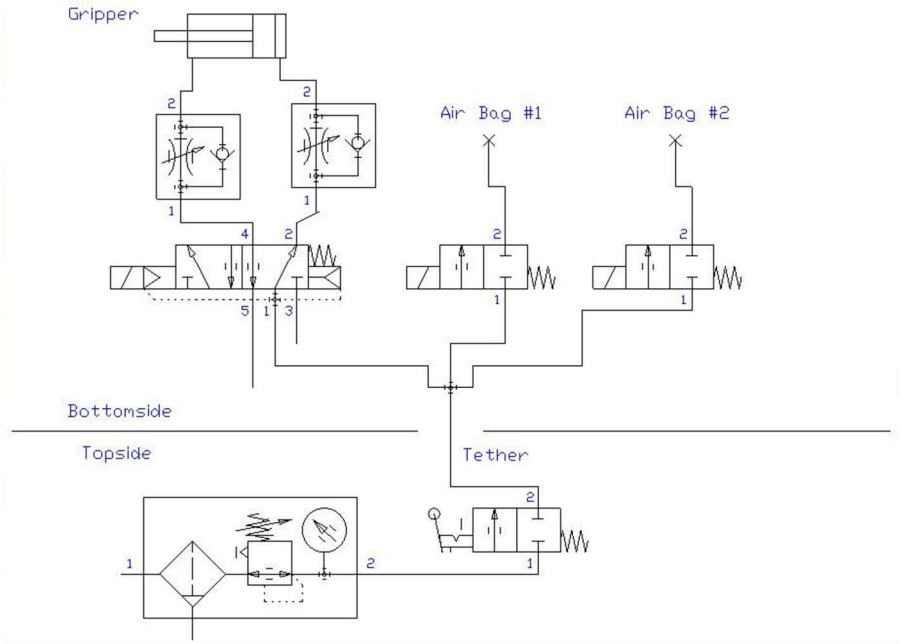
Fuse Calculation:

5 Motors, 4.25A each = 21.25A
3 Cameras, 125mA each = 375mA
Gyro/Accelerometer = 4 mA
12V Solenoids, 560mA each = 1.12A
24V Solenoid = 500 mA

Total = 23.249 A x 150% = 34.87 A

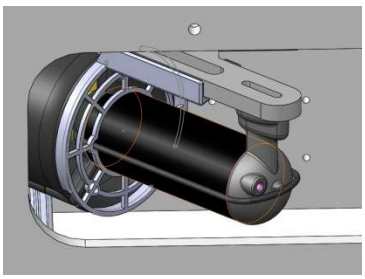
ROV uses a **30A fuse**

Pneumatic SID



SAFTEY

Throughout the build of the ROV the company kept a keen eye for where implementing a safety feature may be necessary. Our philosophy is that an untrained person could safely operate the vehicle without training. This started with making the controls intuitive so that any pilot could safely navigate through the water. Another key safety feature was to ensure a pilot had a sense of depth perception so they know how close the ROV is to objects.



A key feature which is a requirement for passing tech are thruster guards to ensure fingers do not get sucked into the thruster. The company made custom 3D printed guards which attached semi permanently to the thruster. These thruster guards are pictured in Figure 20

Figure 20 - Custom 3D printed thruster guards

The most important safety feature on the ROV is the shutdown sequence when communication is lost but power is not. Our ROV checks that it is getting updated commands from the surface constantly and should that communication stop the ROV shuts down all motors to zero thrust. Due to the positive buoyancy of the ROV, it will then slowly float to the surface of the water.

CHALLENGES

Technical Challenges

Since this was the first underwater ROV designed by the team, there were numerous technical challenges the team had to overcome. The main technical challenges the team encountered originated from the ROV frame and electrical housing design, the ROV's communication programming, and the mission specific tooling design. The technical challenges associated with the frame and electrical housing design primarily involved the calculations involved in estimating the ROV's total buoyancy force and drag center. The estimation of the ROV's total buoyancy force was difficult because of the complex nature of the volume displaced by the ROV and the unknown parameters of the mission specific tooling that had not been developed before building the ROV's frame and electrical housing. In order to overcome the technical challenge involved in estimation/calculation of the ROV's buoyancy force, the team added additional weights and fabricated the electrical housing out of Urethane foam in order to be easily machined down to tune the total buoyancy force after the ROV was fully assembled. To overcome technical challenges in estimation/calculation of the center of drag of the ROV, the team simplified the frame to rectangular shapes and drag factors. Estimating the center of drag was important to analytically validate the potential stability of the ROV while in motion.

In terms of programming, the task that had the highest priority, was the serial communication between the top side and the ROV. The procedure in solving this task began with brainstorming ideas on how this can be programmed, and testing these methods using online Arduino simulation software. Initially, a simple algorithm was created in which raw data would be sent and read; however, assigning these data values to a variable on the receiving end would be nearly impossible because the order of the data being sent is unreliable because the timing between sending and receiving data can be easily off, it also would only work for a simplex communication circuit. At this point, the algorithm was changed slightly to add a specific and constant character at the beginning of each iteration of the loop function and a wait function to create a quasi-duplex circuit. Although this is a viable solution, it does not fully utilize the RS485 protocol's capability since wait function will slow down the overall data transfer rate. In this case, we implemented an Arduino library created by a member of the Arduino community that eases the implementation of serial communication and fully utilizes its capabilities. The library saves the values into the memory of the Arduino and sends the chunk of memory through serial. Overall, the communication challenges in programming stemmed from the timing between the two Arduinos.

Technical challenges involved with the mission specific tooling design were primarily found during system implementation. Because each of the three missions were proven to be substantially different, they required specialized tooling that created technical challenges in programming and electrical system design. The sub systems that required servo motors, Bluetooth sensors, and wi-fi sensors were the primary source of technical challenges in programming implementation. Technical challenges in the electrical system were also found because of the need to accommodate several different voltages ranging from 3.3V to 24V. Although this was not a huge challenge, it created unwanted complexity in the electronics housing.

Teamwork Challenges

Challenges in teamwork originated primarily in coordination of project time for each team member. Because each team member was enrolled in a Bachelors of Engineering and had to complete additional course assignments, it was difficult to coordinate group project work and manufacturing sessions. Because it was difficult coordinating group project work sessions, team members had to perform work outside of school time without proper ability to involve other team members during their design. This lack of consulting other team members created problems later on in manufacturing in the form of mechanical interference between individual components, and electrical errors in PCB design that required the PCB to be re-designed and remade. These avoidable mechanical interferences and electrical design errors caused the team time and created additional costs. The nature of the small design team and large scope provided by MATE also increased the presence of errors created during design that were not found until manufacturing because projective objectives had to be completed rapidly by team members to meet deadlines.

LESSONS LEARNED

Learning how to efficiently waterproof our components and electronics housing was a very essential aspect of this project. To seal the electronics housing, a custom O-ring was used and in order to utilize it properly the groove in which the O-ring sits must be precision machined to get a smooth finish. This requirement for using O-rings deterred us from using them with 3D printed parts since it is difficult to achieve the required surface finish. Furthermore, using O-rings in this project showed us the potential of O-rings, in that they can be used to seal almost any shaped housing as long as there are no sharp corners and that the groove has a smooth surface finish. Apart from O-rings, to waterproof our components, 3M 5200 (Marine Adhesive Sealant) and epoxy were used to coat the Bluetooth module, Wi-Fi module, and the servo motor. The epoxy was proved to be very effective and efficient in that it set quickly especially compared to the 3M 5200. The marine adhesive sealant we used was very effective when used to seal cables and connections exposed to water; however, the drawback that came with this product is that its setting time was in excess of four days. This created setbacks in testing, and fully implementing certain task specific components.

Valuable lessons learned when working with electronics were cable management and the importance of correct and complete wiring diagrams. Since there are many subsystems that are interconnected, these aspects were intrinsic in building a functional ROV. For example, the initial wiring of the control box was not optimal and the untidiness caused wires to disconnect and produce unreliable data values. A simple fix of shortening the wires to a more reasonable length and adding heat shrink to organize the wires helped mitigate this issue. Furthermore, many iterations of the top and bottom side Arduino shields were made due to incomplete or incorrect wiring diagrams. This issue caused the timeline to slow down since these shields were necessary for implementing most of the electronic subsystems.

Furthermore, working within competition rules enabled the team to be more creative work around certain guidelines to achieve a working and competition-ready ROV. For example, to control the thrusters, only electronic speed controllers for brushless DC motors or H-bridge based motor controllers for brushed DC motors can be used. This guideline was a setback since the initial motor controller was not H-bridge based instead it used on small on-board microcontroller to determine the amount of voltage being sent to the motors. However, a simple replacement to an H-bridge motor controller helped us remain within our competition constraints.

TEAM REFLECTIONS

Jonathan Murphy, Project Manager

Having designed and manufactured a 3 degrees of freedom robot with a team of 4 in my second year at BCIT, I welcomed the challenge of building an ROV. Having been my first experience acting as a project manager, I was able to learn and apply the fundamentals of project management I have learned in previous courses. Applying project management instruments such as Gantt and milestone charts provided me with excellent experience. Although these time management tools were implemented, a simple improvement would be starting the design of the ROV earlier to decrease stress around deadlines. Overall, designing an ROV for the MATE international ROV competition was a worthwhile experience and learning opportunity.

Andrew Miltimore, Lead Designer

I couldn't think of a better project to demonstrate the skills learned in post-secondary that will need to be applied in the real world. This project exposed me to new engineering practices that required me to learn and understand prior to implementing them myself. A great example of this is in Task 1 where I designed the Bluetooth module to release the lift bag. The concept of UART data transfer has been taught to me however the implementation of this in a Bluetooth module was a new challenge that required learning new concepts and protocols. The large scope of this project had me soldering one day and CNC machining the next, I hope future students will take the leap and take on this project as their capstone project.

Oliver Law, Task Lead

The ROV project was an excellent learning experience as mechanical engineers can be exposed to electrical system design and constraints along with mechanical design. With previous experience in completing a three degrees of freedom robotic arm, this project allowed me to apply the knowledge of robotic systems and collaborating mechanical and electrical designs. The best lessons learned from completing this project were the ability to meet strict deadlines and design components within specified rules and constraints. The ROV competition gave enough leeway such that designing components for individual tasks included some freedom although designing around safety codes and competition rules gave me a sense of direction. While brainstorming how to complete Task 3, I learned that simple designs work best and trying to create a complicated design will just create problems for the team down the line. Working within time constraints and alongside school assignments and labs, the most important thing is that the job gets done at the end of the day so aim for a simple design that works and requires minimal troubleshooting. Overall, creating an ROV for the MATE competition helped me grow as a future mechanical engineer and provided me with the experience and knowledge to take into the workforce.

Ethan Quisias, Electronics Lead

With robotics and electro-mechanical systems as personal interests, I was really intrigued by the designing and building of an ROV as my capstone project. Similar to my second year project, I was the electronics lead for the ROV. My experience from second year gave me the fundamental knowledge of programming, control systems, wiring, and power; but the ROV enabled me to improve my skills in the electrical and control systems aspect. Applying my knowledge in wiring diagrams I was able to create custom PCBs to tame what would have been a very messy electronics housing. Keeping the wiring as clean and organized as possible allowed for very easy debugging whenever there is a problem. I found cable management a key consideration since our electronics housing already contains many components, so the space for wires is minimal. In terms of programming and communication, I gained valuable experience in different communication protocols when researching our best option, and in different programming techniques to minimize floating-point operations which improves the operating and communication speed. Overall, creating an ROV for the international MATE ROV competition was a great learning opportunity that diversified and improved my set of skills.

FUTURE IMPROVEMENTS

As with any design, there are always room for improvements. Upon a group reflection, The ROV *Marauder* created by DBME this year could be vastly improved. The largest improvement for the ROV involves developing a smaller frame and more compact electronics housing. This would allow the ROV to displace less water and in turn, require less additional weight to achieve neutral buoyancy. Although this is effective, adding additional weight to compensate for over-buoyancy is extremely inefficient. Another improvement would be to utilize smaller cameras that operated on an IP network on board the ROV, instead of CCTV video signals. An IP networked camera system would create a more simplistic viewing system because the cameras would be easier to interface with surface level computers. Although Seabotix BTD150 thrusters offer professional grade capabilities, it was difficult to minimize the overall size of ROV *Marauder* because of their overall size. As a solution, further improvements to this years ROV would be to use the BlueRobotics T200 thrusters. Blue Robotics T200 thrusters are anticipated by the design team to greatly reduce the overall size and cost of future ROV's developed by the school.

COST ACCOUNTING

Without incorporating travel and lodging, *Marauder* was built on an allocated budget from BCIT's Mechanical Engineering department of roughly $1000. Deep Blue Marine Engineering acquired donations from outside vendors and through BCIT to bring our total costs down. By re-using previous components, *Marauder* was manufactured under budget with an estimated value of $865.55 as opposed to $9912.30 without re-using components and acquiring donations. Cost accounting showed that the students are expected to cover the cost of gas, approximately $150, with their personal contributions to the team. The cost of work provided by students was not included because it is difficult to put a price on knowledge and experience gained.

BCIT Deep Blue Marine Engineering Cost accounting for 2018					
Budget Category	Item and Description	Type	Estimated Cost	Actual Incurred Cost	Allocated Budget
Electrical: Components	Mini Analog Joystick - 10k POT	Purchased	$ 19.99	$ 19.99	
	Accelerometer (LSM303)	Purchased	$ 20.81	$ 20.81	
	Fuse Holder (60A)	Purchased	$ 9.92	$ 9.92	
	Fuse (30A)	Purchased	$ 5.82	$ 5.82	
	Fuse (25A)	Purchased	$ 5.82	$ 5.82	
	HS645MG servo motor	Purchased	$ 37.20	$ 37.20	
	Logic level converted	Purchased	$ 5.90	$ 5.90	
	RS485 IC (MAX488ESA+CT-ND)	Purchased	$ 19.80	$ 19.80	
	Pressure/Depth Sensor	Purchased	$ 14.58	$ 14.58	
	MPU9250 (Gyro/Accel/Mag)	Purchased	$ 19.17	$ 19.17	
	Cytron 10A Dual Motor Driver	Purchased	$ 60.24	$ 60.24	
	Kuman ESP8266-01	Purchased	$ 15.00	$ 15.00	$ 300.00
Electrical: Voltage Regulators	Voltage Regulator 48V-24V	Purchased	$ 50.94	$ 50.94	
	Voltage Regulator 48-12V	Purchased	$ 28.93	$ 28.93	
	LM2596 DC to DC (adjustable)	Purchased	$ 16.99	$ 16.99	$ 100.00
Electrical: Tether	CAT5 Cabling (Stranded)	Purchased	$ 80.00	$ 80.00	
	Power Cable	Purchased	$ 50.00	$ 50.00	
	Anderson PowerPole connectors	Purchased	$ 17.99	$ 17.99	$ 150.00
Electrical: Microcontrollers	Arduino Mega	Purchased	$ 43.72	$ 43.72	
	Arduino Nano	Purchased	$ 26.99	$ 26.99	$ 80.00
Electrical: Video System	4 Channel BNC to RJ45 Video Balun	Purchased	$ 31.14	$ 31.14	
	Cameras (LCA 7700-CW)	Re-used	$ 2,700.00	$ -	$ 40.00
Electrical: Electrical Connectors	Waterproof Subconn Connectors	Re-used	$ 1,000.00	$ -	
	Miscelaneous connectors	Purchased	$ 30.00	$ 30.00	$ 30.00
Mechanical: Manufacturing	3D printing materials (PLA, PETG)	Purchased	$ 60.00	$ 60.00	
	Epoxy Glue	Donated (BCIT)	$ 50.00	$ -	$ 60.00
Mechanical: Electrical Housing	3/8 inch 6061 Aluminum Sheet	Purchased	$ 133.66	$ 133.66	
	Urethane Foam	Donated (FiberTek)	$ 150.00	$ -	
	6mm lexan sheet (24"x24")	Donated (Bcit Racing)	$ 50.00	$ -	
	1/4-20 x 1/2 inch bolts (SS)	Donated (Pacific Fastener)	$ 34.00	$ -	$ 170.00
Mechanical: ROV Frame	24"x36" sheet (UHMW PE)	Purchased	$ 60.94	$ 60.94	
	Aluminum t-slots (1"x1")	Donated (RMMC)	$ 42.75	$ -	
	Bolts and Washers	Donated (Pacific Fastener)	$ 20.00	$ -	$ 70.00
Mechanical: Thrusters	Seabotix (BTD-150)	Re-used	$ 5,000.00	$ -	0
Total Expenses for ROV construction			$ 9,912.30	$ 865.55	$ 1,000.00
Lodging and Travel	Hotel	Purchased	$ 750.00	$ 750.00	
	Gas	Purchased	$ 150.00	$ 150.00	
	Competition Registration	Purchased	$ 350.00	$ 350.00	
Total Expenses for team travel			$ 1,250.00	$ 1,250.00	$ 1,500.00
School Donations	BCIT Mechanical Engineering Department	Donation	$ 350.00	$ -	
	BCIT Student Association	Donation	$ 750.00	$ -	
	Captone Project Funding	Donation	$ 865.55	$ -	
Total School Donations for 2018 year			$ 1,965.55	$ -	
Total Expenses			$ 11,162.30	$ 2,115.55	
Total School Donations			$ -	$ 1,965.55	
Funds for next year investment			$ -	$ -	
Costs to cover by student members (gas money)			$ -	$ 150.00	$ -

APPENDIX

Arduino Code for Surface control box (Master)

```
#include <EasyTransfer.h>

EasyTransfer dataIN, dataOUT;

struct RECEIVE {
  double ax, ay, az, gx, gy, gz, yaw, pitch, roll;
  byte gripperStatus;
  byte task1Status;
  byte statusCW;
  byte statusCCW;
  byte statusMPU9250;
};

struct SEND {
  int pwm_1;
  int pwm_2;
  int pwm_3;
  int pwm_4;
  int pwm_5;
  byte airBag_1;
  byte airBag_2;
  byte servoCW;
  byte servoCCW;
  byte gripper;
  byte proNano;
};

RECEIVE rxdata;
SEND txdata;

double ax;

/****** Initialize Pins for I/O ******/
int pin_x = 0;
int pin_y = 1;
int pin_yaw = 2;
int pin_heave = 3;
int pin_zOffset = 4;
int airBag1 = 31;
int airBag2 = 33;
int Servo_CW = 35;
int Servo_CCW = 37;
int gripper_pin = 41;
int pro_nano = 39;
/************************************/

/****** Initialize Variables ******/
int joystick_x, joystick_y, yaw, heave, zOffset;
int x[5], y[5], yw[5], h[5], z[5];
int m_1[5], m_2[5], m_3[5], m_4[5], m_5;
```

```
int pwm_1, pwm_2, pwm_3, pwm_4;
float pi = 3.14159;
float phi = pi / 4, theta = pi / 4, d_COG = 2.3348666 / 3; // 2.3348666
float t_x = cos(phi);
float t_y = sin(phi);
float t_yaw = d_COG;
/*********************************/

int servoCW_pin = 2; // Gripper closed/open
int servoCCW_pin = 3; // GUI communication established
int LB1_pin = 4; // Pro Nano on/off
int commROV_pin = 5; // Serial communication established between Master and Slave
Arduinos
int gripperStatus_pin = 6; // Servo CW
int commGUI_pin = 7; // Servo CCW
int statusTask1_pin = 8; // MPU9250 communication established

/************* GUI Setup ***************
String inString;
int inByte = 0;
boolean firstContact = false;
int i;
int sensorNumber = 0;
/*************************************/

void setup() {
  Serial.begin(115200);
  Serial1.begin(115200);

  dataIN.begin(details(rxdata), &Serial1);
  dataOUT.begin(details(txdata), &Serial1);

  pinMode(servoCW_pin, OUTPUT);
  pinMode(servoCCW_pin, OUTPUT);
  pinMode(LB1_pin, OUTPUT);
  pinMode(commROV_pin, OUTPUT);
  pinMode(gripperStatus_pin, OUTPUT);
  pinMode(commGUI_pin, OUTPUT);
  pinMode(statusTask1_pin, OUTPUT);

  digitalWrite(servoCW_pin, LOW);
  digitalWrite(servoCCW_pin, LOW);
  digitalWrite(LB1_pin, LOW);
  digitalWrite(commROV_pin, LOW);
  digitalWrite(gripperStatus_pin, LOW);
  digitalWrite(commGUI_pin, LOW);
  digitalWrite(statusTask1_pin, LOW);

  pinMode(airBag1, INPUT);
  pinMode(airBag2, INPUT);
  pinMode(Servo_CW, INPUT);
  pinMode(Servo_CCW, INPUT);
  pinMode(gripper_pin, INPUT);
  pinMode(pro_nano, INPUT);
```

```
  /*while (Serial.available() <= 0) {
    // pwm1, pwm2, pwm3, pwm4, pwm5, ax, ay, az, yaw, pitch, roll
    // Serial.println("0,0,0,0,0,0,0,0,0,0,0");
    Serial.println("0,0,0,0,0");
    Serial.flush();
    delay(500);
  }*/
}

/*****************************************************************************/

void loop() {

  //if (Serial.available() > 0); {

    //digitalWrite(commGUI_pin, HIGH);

    if (Serial1.available()) {
      digitalWrite(commROV_pin, HIGH);
    }

    /*if (txdata.gripper) {
      digitalWrite(gripperStatus_pin, HIGH);
    }
    if (!(txdata.gripper)) {
      digitalWrite(gripperStatus_pin, LOW);
    }*/



    /********* Thrust Vectoring Control Values *********/
    for (int i = 0; i < 5; i++) {
      x[i] = (analogRead(pin_x) >> 2);  // Read potentiometer value --> convert from
10-bit to 8-bit number
      y[i] = (analogRead(pin_y) >> 2);  // Read potentiometer value --> convert from
10-bit to 8-bit number
      yw[i] = (analogRead(pin_yaw) >> 2); // Read potentiometer value --> convert
from 10-bit to 8-bit number
      h[i] = (analogRead(pin_heave) >> 2); // Read potentiometer value --> convert
from 10-bit to 8-bit number
      z[i] = (analogRead(pin_zOffset) >> 2);
    }
    joystick_x = (x[0] + x[1] + x[2] + x[3] + x[4]) / 5;
    joystick_y = (y[0] + y[1] + y[2] + y[3] + y[4]) / 5;
    yaw = (yw[0] + yw[1] + yw[2] + yw[3] + yw[4]) / 5;
    heave = (h[0] + h[1] + h[2] + h[3] + h[4]) / 5;
    zOffset = (z[0] + z[1] + z[2] + z[3] + z[4]) / 5;

    /***************** Deadzone *****************/
    if (joystick_x > 120 && joystick_x < 136) {
      joystick_x = 128;
    }
    if (joystick_y > 120 && joystick_y < 136) {
      joystick_y = 128;
    }
```

```
if (yaw > 120 && yaw < 136) {
  yaw = 128;
}
if (heave > 120 && heave < 136) {
  heave = 128;
}
/*****************************************/

for (int i = 0; i < 5; i++) {
  m_1[i] = (joystick_y / t_y + joystick_x / t_x + yaw / t_yaw);
  m_2[i] = (joystick_y / t_y - joystick_x / t_x - yaw / t_yaw);
  m_3[i] = (-joystick_y / t_y + joystick_x / t_x - yaw / t_yaw);
  m_4[i] = (-joystick_y / t_y - joystick_x / t_x + yaw / t_yaw);
}
pwm_1 = (m_1[0] + m_1[1] + m_1[2] + m_1[3] + m_1[4]) / 5;
pwm_2 = (m_2[0] + m_2[1] + m_2[2] + m_2[3] + m_2[4]) / 5;
pwm_3 = (m_3[0] + m_3[1] + m_3[2] + m_3[3] + m_3[4]) / 5;
pwm_4 = (m_4[0] + m_4[1] + m_4[2] + m_4[3] + m_4[4]) / 5;

zOffset = map(zOffset, 0, 255, -127, 128);
pwm_1 = map(pwm_1, 0, 1048, 0, 255);
pwm_2 = map(pwm_2, -688, 360, 0, 255);
pwm_3 = map(pwm_3, -688, 360, 0, 255);
pwm_4 = map(pwm_4, -721, 327, 0, 255);
m_5 = heave + zOffset;

if (m_5 > 255) {
  m_5 = 255;
}
if (m_5 < 0) {
  m_5 = 0;
}

/*************************************************/

digitalWrite(gripperStatus_pin, txdata.gripper);
digitalWrite(statusTask1_pin, txdata.proNano);
digitalWrite(servoCW_pin, txdata.servoCW);
digitalWrite(servoCCW_pin, txdata.servoCCW);
digitalWrite(LB1_pin, txdata.airBag_1);

/********* Peripheral Controls **********/
txdata.airBag_1 = digitalRead(airBag1);
txdata.airBag_2 = digitalRead(airBag2);
txdata.servoCW = digitalRead(Servo_CW);
txdata.servoCCW = digitalRead(Servo_CCW);
txdata.gripper = digitalRead(gripper_pin);
txdata.proNano = digitalRead(pro_nano);
/*************************************/

/********* Send Data to Slave Arduino *********/
txdata.pwm_1 = pwm_1;
txdata.pwm_2 = pwm_2;
txdata.pwm_3 = pwm_3;
txdata.pwm_4 = pwm_4;
```

```
    txdata.pwm_5 = m_5;
    dataOUT.sendData();
    /*********************************************/

    /******** Receive Data from Slave Arduino ********/
    for (int i = 0; i < 5; i++) {
      dataIN.receiveData();
    }
    /*************************************************/

    /********** Status LEDs **********/

    /*********************************/

    /********** GUI Communication **********
    // Read serial data
    inByte = Serial.read();
    if (inByte > 0 && isDigit(inByte)) // Expecting a "1" from Processing.
    {
      inString = (char)inByte;
      sensorNumber = inString.toInt();
      // Send back the requested results
      Serial.print(String(pwm_1) + ",");
      Serial.print(String(pwm_2) + ",");
      Serial.print(String(pwm_3) + ",");
      Serial.print(String(pwm_4) + ",");
      Serial.print(String(m_5) + ",");
      /*Serial.print(String(1000 * rxdata.ax) + ",");
      Serial.print(String(1000 * rxdata.ay) + ",");
      Serial.print(String(1000 * rxdata.az) + ",");
      Serial.print(String(1000 * rxdata.yaw) + ",");
      Serial.print(String(1000 * rxdata.pitch) + ",");
      Serial.println(String(1000 * rxdata.roll));
    }
  }
  /************************************/

  Serial.println("m_1: " + String(pwm_1) + "  " + "m_2: " + String(pwm_2) + "  " +
"m_3: " + String(pwm_3) + "  " + "m_4: " + String(pwm_4) + "  " + "m_5: " +
String(m_5) + "  " + "z Offset: " + String(zOffset) + "  " + "airbag 1: " +
String(txdata.airBag_1) + "  " + "airbag 2: " + String(txdata.airBag_2) + "  " +
"Servo CW: " + String(txdata.servoCW) + "  " + "servo CCW: " +
String(txdata.servoCCW) + "  " + "Pro Nano: " + String(txdata.proNano) + "  " +
"Gripper: " + String(txdata.gripper));
}
```

Arduino Code for ROV electronics housing (Slave)

```
#include <PWM.h>
#include <EasyTransfer.h>
//#include "quaternionFilters.h"
//#include "MPU9250.h"
#include <Servo.h>
#include <I2Cdev.h>
#include <helper_3dmath.h>
#include <MPU6050_6Axis_MotionApps20.h>

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

/********* Initialize Serial Communication with Master Arduino *********/
EasyTransfer dataIN, dataOUT;

struct RECEIVE {
  int pwm_1;
  int pwm_2;
  int pwm_3;
  int pwm_4;
  int pwm_5;
  byte airBag_1;
  byte airBag_2;
  byte servoCW;
  byte servoCCW;
  byte gripper;
  byte proNano;
};

struct SEND {
  double ax, ay, az, gx, gy, gz, yaw, pitch, roll;
  byte gripperStatus;
  byte task1Status;
  byte statusCW;
  byte statusCCW;
  byte statusMPU9250;
};

RECEIVE rxdata;
SEND txdata;
/**********************************************************************/

/********* Initialzie Pins for I/O's *********/
int pwm_pin1 = 30, pwm_pin2 = 32; // PWM Pins for Motor Controller 1
int pwm_pin3 = 34, pwm_pin4 = 36; // PWM Pins for Motor Controller 2
int pwm_pin5 = 38; // PWM Pin for Motor Controller 3
int dir_1 = 3, dir_2 = 5, dir_3 = 6, dir_4 = 7, dir_5 = 8; // DIR Pins for Motor
Controllers
int pressure_pin = 0;
/*****************************************/
```

```
/********* Intitialize Variables *********/
int m_1, m_2, m_3, m_4, m_5;
int32_t frequency = 8000; // Set PWM Frequency to 20 kHz
uint16_t res;
uint16_t dzTop, dzBottom;
uint16_t maxPWM, minPWM;
uint16_t dead;
int pressure_analog;
double pressure_voltage;
int communication[5];
double commAvg;
/*************************************/

/********* Initialize MPU9250 *********
#define AHRS false
int intPin = 12;
//MPU9250 myIMU;
double fXg = 0;
double fYg = 0;
double fZg = 0;
const float alpha = 0.5;

double a_x[4], a_y[4], a_z[4];
double delta = 1 / 1000;
/*************************************/

/*************** MPU6050 Setup ***************/
MPU6050 mpu;
// Connect INT pin to DIO PIN 2

#define OUTPUT_READABLE_QUATERNION
#define OUTPUT_READABLE_YAWPITCHROLL

bool dmpReady = false;
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0
= error)
uint16_t packetSize;
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

Quaternion q;
VectorInt16 aa;
VectorInt16 aaReal;
VectorInt16 aaWorld;
VectorFloat gravity;

volatile bool mpuInterrupt = false;
void dmpDataReady() {
  mpuInterrupt = true;
}
/*****************************************/

/************** Peripherals Initialization ***************/
int airBag1 = 41;
```

```
int airBag2 = 43;
int servo = 2;
int gripper_pin = 45;
int pro_nano = 47;
int servoCW_pin = 10;
int servoCCW_pin = 9;

Servo dank_servo;
/*********************************************************/

void setup() {
  /******** Initialize Serial Comm. *********/
  Serial.begin(115200);
  Serial1.begin(115200);
  dataIN.begin(details(rxdata), &Serial1);
  dataOUT.begin(details(txdata), &Serial1);
  /***************************************/

  /*********** Peripheral Setup *************/
  pinMode(airBag1, OUTPUT);
  pinMode(airBag2, OUTPUT);
  pinMode(gripper_pin, OUTPUT);
  pinMode(pro_nano, OUTPUT);
  pinMode(servoCW_pin, OUTPUT);
  pinMode(servoCCW_pin, OUTPUT);

  dank_servo.attach(servo);
  /***************************************/

  /********** Initialize Motor Control and PWM Frequency ***********/
  InitTimersSafe(); // Initialize all timers except Timer 0
  SetPinFrequencySafe(dir_1, frequency);
  SetPinFrequencySafe(dir_2, frequency);
  SetPinFrequencySafe(dir_3, frequency);
  SetPinFrequencySafe(dir_4, frequency);
  SetPinFrequencySafe(dir_5, frequency);
  // SetPinFrequencySafe(servo, freqServo);

  res = Timer3_GetTop();
  dzTop = res / 2 + 12;
  dzBottom = res / 2 - 12;
  maxPWM = res * 0.75;
  minPWM = res * 0.25;
  dead = res / 2;

  Serial.println(String(dead) + "  " + String(res) + "  " + String(maxPWM) + "  " +
String(minPWM));

  pinMode(pwm_pin1, OUTPUT);
  pinMode(pwm_pin2, OUTPUT);
  pinMode(pwm_pin3, OUTPUT);
  pinMode(pwm_pin4, OUTPUT);
  pinMode(pwm_pin5, OUTPUT);
  /*************************************************************/
```

```
  txdata.statusMPU9250 = LOW;
  txdata.gripperStatus = LOW;
  txdata.statusCW = LOW;
  txdata.statusCCW = LOW;
  txdata.task1Status = LOW;
  dataOUT.sendData();

  /***************** Setup MPU9250 ********************
  Wire.begin();
  pinMode(intPin, INPUT);
  digitalWrite(intPin, LOW);

  byte c = myIMU.readByte(MPU9250_ADDRESS, WHO_AM_I_MPU9250);

  if (c == 0x71) // WHO_AM_I should always be 0x71
  {
    myIMU.MPU9250SelfTest(myIMU.selfTest); // start by performing self test
    myIMU.calibrateMPU9250(myIMU.gyroBias, myIMU.accelBias);

    myIMU.initMPU9250();
    myIMU.getAres();
    myIMU.getGres();
  }
  else
  {
    txdata.statusMPU9250 = HIGH;
    dataOUT.sendData();
    abort();
  }
  /*****************************************************/

  /******************** MPU6050 Setup/Initialization **********************/
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  TWBR = 24;
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif

  mpu.initialize();     // initialize device
  devStatus = mpu.dmpInitialize();

  mpu.setXGyroOffset(95);
  mpu.setYGyroOffset(42);
  mpu.setZGyroOffset(35);
  mpu.setXAccelOffset(-2519);
  mpu.setYAccelOffset(1559);
  mpu.setZAccelOffset(3039);

  if (devStatus == 0) {
    mpu.setDMPEnabled(true);
    attachInterrupt(0, dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();

    dmpReady = true;
```

```
    packetSize = mpu.dmpGetFIFOPacketSize();
  }
  /*************************************************************************/

}

void loop() {


  /********* INPUT Data **********/
  for (int i = 0; i < 5; i++) {
    dataIN.receiveData();
  }
  /*****************************/


  /************************** GYRO/ACCEL SYSTEM ***************************/
  if (!dmpReady) return;
  while (!mpuInterrupt && fifoCount < packetSize) {}

  mpuInterrupt = false;
  mpuIntStatus = mpu.getIntStatus();

  if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
    mpu.resetFIFO();
  }
  else if (mpuIntStatus & 0x02) {
    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount(); // wait for
correct available data length

    //read a packet from FIFO
    mpu.getFIFOBytes(fifoBuffer, packetSize);
    fifoCount -= packetSize;
  }

  mpu.dmpGetQuaternion(&q, fifoBuffer);
  mpu.dmpGetAccel(&aa, fifoBuffer);
  mpu.dmpGetGravity(&gravity, &q);
  mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
  mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);

  txdata.ax = aaWorld.x;
  txdata.ay = aaWorld.y;
  txdata.az = aaWorld.z;
  txdata.gx = 0;
  txdata.gy = 0;
  txdata.gz = 0;
  txdata.roll = ypr[2];
  txdata.pitch = ypr[1];
  txdata.yaw = ypr[0];
  /*************************************************************************/

  /************** MOTOR CONTROL SYSTEM **************/
  digitalWrite(pwm_pin1, HIGH);
  digitalWrite(pwm_pin2, HIGH);
```

```
  digitalWrite(pwm_pin3, HIGH);
  digitalWrite(pwm_pin4, HIGH);
  digitalWrite(pwm_pin5, HIGH);

  m_1 = map(rxdata.pwm_1, 0, 255, 0, res - 1);
  m_2 = map(rxdata.pwm_2, 0, 255, 0, res - 1);
  m_3 = map(rxdata.pwm_3, 0, 255, 0, res - 1);
  m_4 = map(rxdata.pwm_4, 0, 255, 0, res - 1);
  m_5 = map(rxdata.pwm_5, 0, 255, 0, res - 1);

  /************ Create Deadzone *************/
  if (m_1 > dzBottom && m_1 < dzTop) {
    m_1 = dead;
  }
  if (m_2 > dzBottom && m_2 < dzTop) {
    m_2 = dead;
  }
  if (m_3 > dzBottom && m_3 < dzTop) {
    m_3 = dead;
  }
  if (m_4 > dzBottom && m_4 < dzTop) {
    m_4 = dead;
  }
  if (m_5 > dzBottom && m_5 < dzTop) {
    m_5 = dead;
  }
  /*****************************************/

  // PWM'ing 24V down to 19V ---> (19/24)*100% = 79.17% ---> Limit the duty cycle
  // +79% = 256*0.79 = 202;
  // Should we have a slight safety factor? Instead of 101 we use 98
  // Max = 256 + 200 = 426;
  // Min = 256 - 200 = 56;
  // The method of motor control we are using is called Locked Anti-Phase Mode --> no
rotation at 50% duty cycle, and getting closer to 0% or 100% duty cycle increases
  // speed in one direction

  if (m_1 >= maxPWM) {
    m_1 = maxPWM;
  }
  if (m_1 <= minPWM) {
    m_1 = minPWM;
  }
  if (m_2 >= maxPWM) {
    m_2 = maxPWM;
  }
  if (m_2 <= minPWM) {
    m_2 = minPWM;
  }
  if (m_3 >= maxPWM) {
    m_3 = maxPWM;
  }
  if (m_3 <= minPWM) {
    m_3 = minPWM;
  }
```

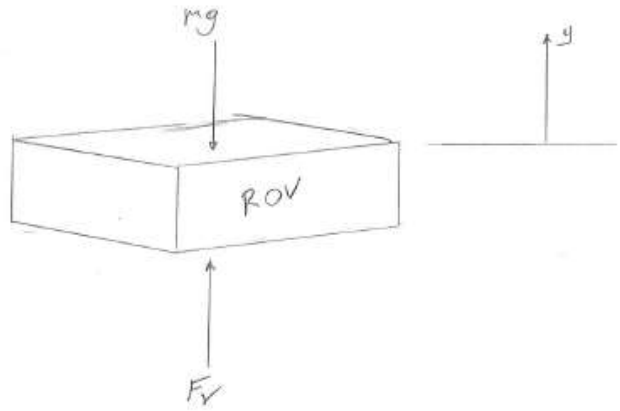
```
  if (m_4 >= maxPWM) {
    m_4 = maxPWM;
  }
  if (m_4 <= minPWM) {
    m_4 = minPWM;
  }
  if (m_5 >= maxPWM) {
    m_5 = maxPWM;
  }
  if (m_5 <= minPWM) {
    m_5 = minPWM;
  }

  /************* KILL MOTORS WHEN DISCONNECTED FROM THE MASTER **************
  for (int i = 0; i < 5; i++){
    communication[i] = 1000*Serial1.available();
  }
  commAvg = (communication[0] + communication[1] + communication [2] +
communication[3] + communication[4])/5;
  if (commAvg >= 5500) {
    m_1 = dead;
    m_2 = dead;
    m_3 = dead;
    m_4 = dead;
    m_5 = dead;
  }

/*****************************************************************************
/

  analogWrite(dir_1, m_1);
  analogWrite(dir_2, m_2);
  analogWrite(dir_3, m_3);
  analogWrite(dir_4, m_4);
  analogWrite(dir_5, m_5);
  /*********************************************************/

  /**************** Peripheral Controls *****************/
  digitalWrite(airBag1, rxdata.airBag_1);
  digitalWrite(airBag2, rxdata.airBag_2);
  if (rxdata.gripper == HIGH) {
    txdata.gripperStatus = HIGH;
  }
  digitalWrite(gripper_pin, rxdata.gripper);
  if (rxdata.proNano == HIGH) {
    txdata.task1Status = HIGH;
  }
  digitalWrite(pro_nano, rxdata.proNano);

  if (rxdata.servoCW == LOW && rxdata.servoCCW == LOW) {
    digitalWrite(servoCW_pin, LOW);
    digitalWrite(servoCCW_pin, LOW);
  }
  if (rxdata.servoCW == HIGH) {
    digitalWrite(servoCW_pin, HIGH);
```

```
    txdata.statusCW = HIGH;
  }
  if (rxdata.servoCCW == HIGH) {
    digitalWrite(servoCCW_pin, HIGH);
    txdata.statusCCW = HIGH;
  }
  /*****************************************************/

  /************* DEPTH SENSOR *************/
  pressure_analog = analogRead(pressure_pin);


  /*************************************/

  /********* OUTPUT Data *********/
  dataOUT.sendData();
  /*******************************/

  Serial.print(m_1);
  Serial.print("\t");
  Serial.print(rxdata.pwm_1);
  Serial.print("\t");
  Serial.print(m_2);
  Serial.print("\t");
  Serial.print(rxdata.pwm_2);
  Serial.print("\t");
  Serial.println(commAvg);
}
```

February 11, 2018 | Mech 8290 - ROV | Jonathan Murphy

Bouyancy of ROV:

↳ Excluding mission specific tools



Electrical housing (urethane foam)

$V_{displaced} = 939.60\ in^3$

$V_{foam} = 628.99\ in^3$

$\rho_{urethane}$

$m_{foam} = \left(\frac{628.99\ in^3}{12^3}\right)\left(8\ lb/ft^3\right) \Rightarrow m_{foam} = 2.912\ lb$

UHMW-PE side panels

$V_{displaced} = 39.14\ in^3$

$\rho_{UHMW} = 0.034\ lb/in^3$

$m_{panel} = (V_{displaced})(\rho_{UHMW}) \Rightarrow m_{panel} = 1.33\ lb$

Poly Carbonate part of Electrical housing

$V_{Displaced} = 58.54\ in^3$

$\rho_{polycarbonate} = 1.2\ g/cm^3 = 0.0434\ lb/in^3$

$m_{poly} = (58.54\ in^3)(0.0434\ lb/in^3) \Rightarrow m_{poly} = 2.54\ lb$

Aluminum part of electrical housing

$V_{Al} = 27.50\ in^3$ $\quad V_0 = \left(\frac{3}{8}\right)(13)(20) = 97.5\ in^3$

$\rho_{Al} = 0.0975\ lb/in^3$

$m_{al} = (27.50\ in^3)(0.0975\ lb/in^3) \Rightarrow m_{al} = 2.68125\ lb$

BTD-150 thruster

$V_d = 17.22\ in^3$

$m_{thruster} = 1.21\ lb$

Thruster mount (UHMW)

$V_d = 2.6\ in^3$

$m = 0.09\ lb$

Aluminum T-slots

$V_d = 6.15\ in^3$

$\rho = 0.0975\ lb/in^3$

$m_{T\text{-}slot} = 0.60\ lb$

LCA 9700 camera

$V_d = 6.03\ in^3$

$m = 0.88\ lbs$

$F_{NET} = F_B - mg$, $F_B = \rho g V$

$mg = m_{foam} + 2(m_{panel}) + m_{body} + m_{alt} + 5(m_{thruster}) + 5(m_{round}) + 4(m_{T\text{-}slot}) + 3(m_{corner})$

$mg = 2.912\,lb + 2(1.331\,lb) + 2.541\,lb + 2.6813 + 5(1.21\,lb) + 4(0.091\,lb) + 4(0.601\,lb) + 3(0.881\,lb)$

$mg = 22.24\,lb$ → 22.55 lb from solidworks but not all densities correct

$V_{total} = \left(939.60\,in^3 + \frac{3}{8}(15)(20)\right) + 2(39.14\,in^3) + 58.54\,in^3 + 17.22\,in^3 + 4(2.61\,in^3) + 5(6.15\,in^3) + 3(6.03\,in^3)$

$V_{total} = 1232.29\,in^3$ → 1249.78 in^3 from solidworks which includes everything

$\rho_{water} = 0.036\,lb/in^3$

$F_B = (1232.29\,in^3)(0.036\,lb/in^3) \Rightarrow F_B = 44.362\,lb$

$F_{NET} = F_B - mg = 44.362\,lb - 22.241\,lb \Rightarrow F_{NET} = 22.122\,lb\uparrow$

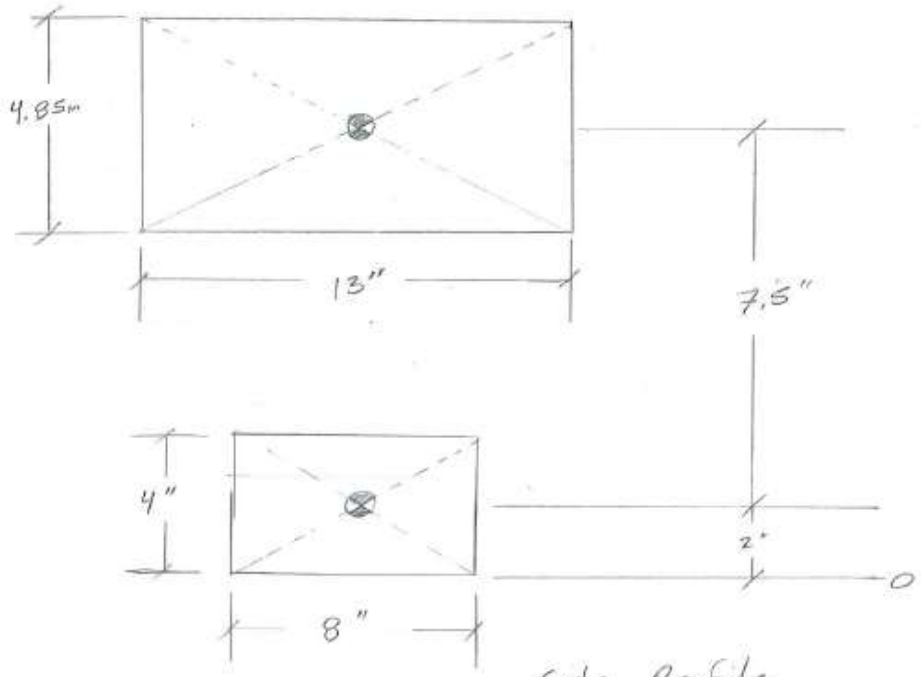
Drag Center Location Calculations/Estimations

February 11, 2018 | Mech 8290 - ROV | Jonathan Melarchy

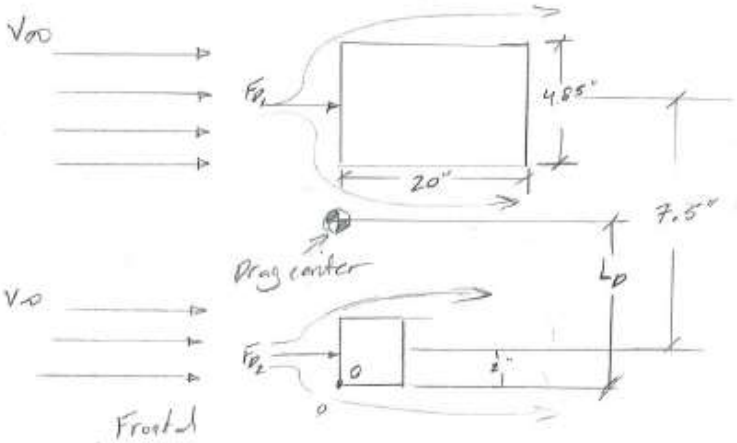
Drag Center Calculation

* Not to scale

Front View



Side Profile



$$F_{Drag} = \frac{1}{2} C_D \rho V^2 A$$ ← Frontal Area

A: Assume $C_D = 2.1$ for F_{D_2} (Square cylinder)

Fluid mechanics → Pg. 485 (F. White)

A: $\frac{L}{H} = 4.124$, $\therefore C_D = 1.15$ for F_{D_1} (Flat nose section)

$L_1 = 9.5"$
$L_2 = 2"$
$C_{D_1} = 1.2$
$C_{D_2} = 1.07$

$$\Sigma M_O = (F_{D_1} + F_{D_2}) L_D$$

$$(F_{D_1}) L_1 + (F_{D_2}) L_2 = (F_{D_1} + F_{D_2}) L_D$$

$$\frac{1}{2} C_{D_1} \rho V_1^2 A_1 L_1 + \frac{1}{2} C_{D_2} \rho V_2^2 A_2 L_2 = (F_{D_1} + F_{D_2}) L_D$$

$$\Rightarrow \frac{1}{2}(1.15)\left(1000 \tfrac{kg}{m^3}\right)(V_1)^2(4.85")(13")(L_1)\left(\frac{0.0254m}{1in}\right)^2$$

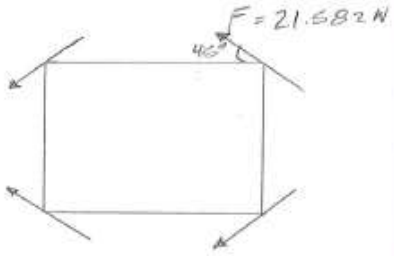
$$+ \frac{1}{2}(2.1)\left(1000 kg/m^3\right)(V_2)^2(4")(8")(L_2)\left(\frac{0.0254m}{1in}\right)^2 = (F_{D_1} + F_{D_2}) L_D$$

$$\Rightarrow \left(23.39 \tfrac{kg}{m}\right) V_1^2 L_1 + \left(21.68 \tfrac{kg}{m}\right) V_2^2 L_2 = (F_{D_1} + F_{D_2}) L_D$$

$$\left[\left(23.39 \tfrac{kg}{m}\right) L_1 + \left(21.68 \tfrac{kg}{m}\right) L_2\right] V^2 = V^2 \left(23.39 \tfrac{kg}{m} + 21.68 \tfrac{kg}{m}\right) L_D$$

Estimation of Velocity after 3 seconds

$m_{ROV} = 10.06 kg$



$$\Sigma F = m_{ROV} a_{ROV}$$

$$m_{ROV} a_{ROV} = 4(21.582 N)\cos(45)$$

$$a_{ROV} = 6.068 m/s^2$$

$$V_{ROV}\Big|_{t=3s} = (6.068 m/s^2)(3s) \Rightarrow \boxed{V_{ROV}\Big|_{t=3s} = 18.20 m/s}$$

Location of Drag center

$$\left(23.39\ \frac{kg}{m}\right)L_1 + \left(21.68\ \frac{kg}{m}\right)L_2 = \left(45.07\ \frac{kg}{m}\right)L_D$$

$$\left(23.39\ \frac{kg}{m}\right)(9.5")\left(\frac{0.0254m}{1"}\right) + \left(21.68\ \frac{kg}{m}\right)(2")(0.0254) = \left(41.07\ \frac{kg}{m}\right)$$

$$6.75\ kg = \left(41.07\ \frac{kg}{m}\right)L_D$$

$$L_D = 0.164\ m \Rightarrow \underline{L_D = 6.47\ in}$$