Backpackable
Underwater Remote Operated Vehicle
N.E.M.O
Second Semester Report
Spring Semester 2013

-Full Report-

By
Sarah Romer
Justin Kopacz
Rachel Dondero
Michael Hake
Luke Stahler

Prepared to partially fulfill the requirements for ECE402

Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, Colorado 80523

Project Advisor: Tony Maciejewski

Approved By: ___All Team Members and Advisor as of 5/4/2013___
Abstract

Underwater caves are considered one of the last unexplored frontiers on our planet. Underwater caves are opening the door to information on some of the oldest human remains known to date. With this new frontier come new challenges for exploration. Scuba divers can only dive to 130ft before it becomes extremely unsafe. With this restriction on depth for human exploration, there is a budding market for underwater remote operated vehicles that can explore underwater caves inaccessible by humans.

Most of the Underwater Remote Operated Vehicles (UROV) on the market today are big, bulky, and heavy. While these UROV’s supply the market for open water exploration, there is still a demand for small, light, and portable UROV’s that is not being filled. This demand comes from the desire to explore underwater caves. Most underwater caves are in remote areas of the wilderness that must be backpacked to, rather than driven or boated to, in order to reach. Another constriction of underwater cave exploration is the fact that caves tend to be long and narrow which unlike open bodies of water where a vehicle can move quickly without many hazards, creates more of an obstacle course where a good control implementation is necessary.

This is where our Backpackable Underwater Remote Operated Vehicle comes in to play. Corey Jaskolski of Hydro-Technologies in Windsor, Colorado, came to Colorado State University’s Electrical and Computer Engineering department searching for a solution to his underwater cave exploration problems. Hydro-Technologies needed a UROV that was light enough to backpack through harsh wildernesses by a team of only two people. They also need a vehicle that was at maximum, one-third the cost of commercial UROV’s.

With these constrictions in mind, the faculty of the ECE department as well as Corey Jaskolski put together a design team of electrical and mechanical engineers to design, build, and test a backpack-able version of a UROV. This version of an ROV will be able to dive to at least 60 meters, withstand temperatures ranging from -10°C to 60°C, and support modularity.
### Table of Contents

Abstract .................................................................................................................. 2
Table of Contents .................................................................................................... 3
List of Figures .......................................................................................................... 5
List of Tables ........................................................................................................... 6
Acknowledgements .................................................................................................. 7
Introduction ............................................................................................................ 8
Problem Statement .................................................................................................. 9
Background .............................................................................................................. 10
Technical Breakdown ............................................................................................... 12

#### Electrical ......................................................................................................... 12
- Communication Interfaces .................................................................................. 12
  - Serial .................................................................................................................. 13
  - I2C ..................................................................................................................... 13
  - oneWire ............................................................................................................. 13
  - Analog ............................................................................................................... 13
- Surface Side Operations ...................................................................................... 14
  - Lights ............................................................................................................... 16
- Control ................................................................................................................ 16
- Sensors ............................................................................................................... 18
  - Humidity ......................................................................................................... 18
  - External Temperature ...................................................................................... 18
  - Pressure Transducer/Depth ............................................................................. 18
  - Compass ......................................................................................................... 19
- Power Analysis ................................................................................................... 19

#### Mechanical ..................................................................................................... 21
- Concept Selection ............................................................................................... 21
  - Initial Concepts ............................................................................................... 21
  - Final Concept .................................................................................................. 26
- Feasibility ........................................................................................................... 32
- Failure Modes and Effects Analysis (FMEA) ...................................................... 32
- Detail Design (Explanation, Material Selection, and Manufacturing and Assembly Considerations) .............................................................................................................. 33
  - Overview ....................................................................................................... 33
  - Pressure Housing ............................................................................................ 33
  - Electronics Sled ............................................................................................... 35
  - External Shell .................................................................................................. 36
  - Control Surfaces .............................................................................................. 37
- Mathematical Modeling and Supporting Analysis .............................................. 38
  - Pressure Stress and Wall Thickness Analysis ................................................ 38
  - Pressure Testing of NEMO ............................................................................... 40
  - Buoyancy Analysis ......................................................................................... 40
  - Buoyancy Testing ............................................................................................ 41
  - Waterproofing and Pressure Testing of the Servo Motors .............................. 41
  - Fatigue Analysis .............................................................................................. 41
  - Control Surface Stress Analysis ...................................................................... 41

#### Proof of Concept Utilizing a Prototype ............................................................. 42
Testing ................................................................. 42
  Pool Tests .................................................................... 42
  Field Tests ..................................................................... 42
Safety Considerations ...................................................... 42

Ethical Issues .................................................................. 43

Conclusions and Future Work ........................................... 44
  Manufacturing and Assembly ........................................ 46
  Testing and Refinement ................................................ 47
  Project/Development Cost Summary .............................. 47
  Conclusions and Recommendations .............................. 47

Bibliography .................................................................. 49

Appendix A: Project Abbreviations ................................. 50
Appendix B: Budget .......................................................... 51
Appendix C: Project Plans ................................................ 56
  January 2013 ................................................................. 56
  September 2012 ............................................................. 57
  ...................................................................................... 58
  ...................................................................................... 59
  ...................................................................................... 60
Appendix D: Servo Motor Pressure Test Plan ..................... 61
Appendix E: Initial Buoyancy Test Plan ............................. 65
Appendix F: Manufacturing Drawings ............................... 66
Appendix G: Project Code ................................................. 69
  NEMO Code: Onboard the Arduino ............................... 69
  NEMO Top Side ............................................................ 80
List of Figures

Figure 1: Communication Interfaces ................................................................. 12
Figure 2: Xbox Controller Diagram................................................................. 14
Figure 3: Light Diagram .................................................................................... 16
Figure 4: Control Feedback .............................................................................. 17
Figure 5: Control System ................................................................................ 17
Figure 6: Low Efficiency Estimation................................................................. 19
Figure 7: High Efficiency Estimation................................................................. 20
Figure 8: Three-thruster Design with One on Tail............................................. 21
Figure 9: Conventional Three-thruster Design ............................................... 21
Figure 10: Four-thruster, Seabotix-Style Design .............................................. 22
Figure 11: Three-thruster, Handle Design ........................................................ 22
Figure 12: Single-thruster, Canard-style Design .............................................. 22
Figure 13: Canard with CO₂ Ballast System .................................................... 23
Figure 14: Two-thruster Design ..................................................................... 23
Figure 15: NEMO Model ............................................................................... 27
Figure 16: Pressure Housing Model ................................................................. 28
Figure 17: External Shell Model ..................................................................... 28
Figure 18: Electronics Sled Model (Empty)...................................................... 29
Figure 19: Electronics Sled Model (Loaded)..................................................... 29
Figure 20: Front Dome Model ........................................................................ 30
Figure 21: Rear Dome Model ........................................................................ 30
Figure 22: Control Surface Model .................................................................. 31
Figure 23: Thruster Model .............................................................................. 31
Figure 24: LED Housing Model ..................................................................... 32
Figure 25: Pressure Housing ......................................................................... 34
Figure 26: UHMW Vessel ........................................................................................................ 34
Figure 27: Front Dome and O-Rings .................................................................................... 35
Figure 28: Rear Dome and Bulkhead Connectors ............................................................... 35
Figure 29: Electronics Sled ................................................................................................. 36
Figure 30: External shell Isometric View ............................................................................. 36
Figure 31: External Shell Bottom View ............................................................................... 37
Figure 32: Control Surface ................................................................................................. 37
Figure 33: Static Nodal Stresses at 60m ............................................................................ 39
Figure 34: Deformation at 60m ......................................................................................... 40
Figure 35: UHMW Vessel Manufacturing Drawing .............................................................. 66
Figure 36: Housing Rod Manufacturing Drawing ............................................................... 67
Figure 37: LED Holder Manufacturing Drawing ................................................................. 67

List of Tables
Table 1: Round One Pugh Analysis .................................................................................... 24
Table 2: Round Two Pugh Analysis ................................................................................... 25
Table 3: Round Three Pugh Analysis ............................................................................... 26
Table 4: Project Status ........................................................................................................ 46
Table 5: Summary of Project Costs .................................................................................... 47
Acknowledgements

The NEMO senior design team sincerely thanks Corey Jaskolski and everyone at Hydro-Technologies for establishing and funding this project. We learned a lot from Mr. Jaskolski’s expertise, and enjoyed working with him to develop a vehicle that satisfies his needs. This was a fantastic learning experience, and the team would like to thank Mr. Jaskolski for sacrificing his time and resources to assist in enriching their college education.

The team would also like to give a heartfelt thank you to all of the folks at Solid Concepts for providing a large amount of additive manufacturing for the project free of charge. Without the support of Solid Concepts, the team would not have been able to realize its vision for the project.

Thank you to Drs. Donahue, Stansloski, Notaros, and Maciejewski for their wisdom and guidance throughout the year. Also, thank you to Dan Pierson for his patience, support, and advice throughout the year.

Finally, thank you to Jacob Karas for his help with coding and topside GUI control; thank you to Tyler Brinks for his contribution to website design and organization; thank you to Jacob Thorton for his mentoring and contribution to the website design. Thank you to everyone else for your support and encouragement.
Introduction

The purpose of this project was to design a backpackable remotely-operated underwater vehicle (ROV) that can be used by researchers with limited budgets to explore areas that are difficult to access and/or too dangerous to explore with human teams.

The design project was sponsored by Corey Jaskolski—President of Hydro Technologies and National Geographic Fellow—who coordinated design requirements and specifications with the student design team. Other stakeholders interested in or impacted by the design included the student design team, the course instructors (Drs. Donahue, Stansloski, and Notaros), the project advisor (Dr. Maciejewski), the Colorado State University departments of mechanical engineering and electrical and computer engineering, as well as all potential beneficiaries of the vehicle (National Geographic, the science community, those interested in underwater exploration, etc.).

The major goal of the design was to develop a vehicle that satisfied the driving forces behind the project as well as all of the customer’s needs. Mainly, this consisted of developing an ROV (and supporting equipment) that is lightweight, robust, and easily portable, as it will be backpacked through difficult conditions. What makes NEMO unique is that it is powered onboard the vehicle (minimizing tether size and weight) and has fiber communication to the topside GUI. The vehicle can also withstand pressures at reasonable depths in addition to tolerating a reasonable range in temperature. The design team consisted of five engineering students; three electrical engineering majors developed, tested, and fine-tuned all of the vehicle’s electrical systems while two mechanical engineering majors completed the design, analysis, fabrication, and testing of the vehicle itself.
Problem Statement

National Geographic researchers have been exploring the world for decades. Although technology has advanced greatly since the first expeditions, explorers are still constrained by the amount of equipment they can take due to the size and weight of their gear. There exist bodies of water that are difficult, if not impossible, to explore with current commercial ROVs due to their sheer size and weight. In addition, these locations are often too dangerous to explore with human scuba teams. The development of a lightweight and portable ROV would allow the explorers to remain safely on the surface and would significantly reduce the cost and risk associated with these treacherous excursions.

The customer, Corey Jaskolski, is the President of Hydro-Technologies and a long-time National Geographic Fellow. He graduated from the Massachusetts Institute of Technology with a Master’s Degree in Electrical Engineering and Computer science. In 2001, Mr. Jaskolski took part in an expedition supporting James Cameron’s documentary filming of the Titanic. During the expedition, Mr. Jaskolski descended to the wreck of the Titanic (12,500 feet) to support robotic ROV operations [10]. Mr. Jaskolski approached Colorado State University with the backpackable ROV project and provided the design team with a $10,000 budget.

Mr. Jaskolski has been on expeditions to some of the dangerous locations previously mentioned and recognized a need in the science community for a lightweight and portable ROV. This need stems from the fact that currently available commercial ROVs are large, heavy, and/or expensive. NEMO is needed now because there are many difficult-to-access bodies of water believed to hold historical remains of ancient peoples and civilizations and a backpackable ROV will make the exploration of these locations much more realistic.

The end-users of NEMO are researchers with limited budgets wishing to explore the previously mentioned difficult-to-access bodies of water. As stated above, other stakeholders interested in or impacted by the design include the student design team, the senior design course instructors (Drs. Donahue, Stansloski, and Notaros), the project advisor (Dr. Maciejewski), the Colorado State University departments of mechanical engineering and computer engineering, the science community, those interested in underwater exploration, and all other potential beneficiaries of the vehicle.
Background

The remotely operated underwater vehicle (ROV) was originally developed by the United States Navy in 1961 as a means to recover torpedoes lost on the ocean floor. Until the mid-1970s, various governments were responsible for the funding and development of the majority of the world’s ROVs; however, 96 percent of the vehicles produced between 1974 and 1982 were developed, funded, or purchased by private industry [1]. The rise of the private industry within the ROV market sparked competition. As a result, ROVs were produced at an accelerated rate. The original driving force behind designing ROVs was to make bigger, more powerful, and deeper-diving vehicles for deep-sea exploration. In 1990, an ROV developed by the U.S. Navy reached a depth of 20,000 feet; shortly after, a Japanese ROV reached the bottom of the Mariana Trench (close to 36,000 feet) [1]. As of 2006, there were over 450 builders and developers of ROVs [1]. Today, ROVs are used for a variety of underwater tasks, from ocean exploration to underwater oil rig maintenance.

While ROV popularity has ballooned over the past five decades, more than two-thirds of the world is underwater and has yet to be explored [3]. A small, yet significant, portion of this unexplored territory lies within difficult terrain, such as alpine lakes and underwater caves. It would be impossible to transport the majority of current commercial ROVs to these locations due to their sheer size and weight. In addition, these locations are too dangerous to explore with human scuba teams. The increased elevation of alpine lakes poses a greater threat of nitrogen saturation and decompression sickness to the divers than similar dives at lower altitudes [4] and caves are dangerous in their own regard. Corey Jaskolski, the president of Hydro Technologies (Windsor, CO), is a National Geographic explorer and has been on expeditions to some of these dangerous locations. Mr. Jaskolski established the Backpackable Underwater ROV Senior Design project because he recognizes a need in the science community for such a device that is currently unfulfilled by commercial ROVs. Mr. Jaskolski wanted to develop a durable and easily transportable inspection-class ROV capable of modularly supporting a full-spherical imager that was recently developed. An inspection-class ROV is used to position a video camera and simple sensor package underwater; it is connected to a small base station via wire tether where it is controlled by the operators [1], [5]. Mr. Jaskolski believes the device will play a significant role in uncovering new species of organisms, ancient artifacts and remains, and other potentially historical finds. NEMO offers a unique ability to observe environments in real-time that are currently impossible to view in any cost-effective or practical manner [6].

There are many observation-class ROVs in the world today (around 3,000 in 2006 [1]); many of these devices satisfy similar needs as those desired of this project. An example of one of these devices is described by [Weaver, et al.]. This ROV minimizes tether diameter by placing batteries onboard the ROV and by utilizing fiber-optic tether; however, the device has over two kilometers of tether and would not be backpackable due to its large, cumbersome size. In contrast, the vehicle described in [3] is very compact and minimizes thruster count to improve the power-weight ratio of the vehicle; though, this device seems rather fragile (not ideal for rugged travel) and has an
unnecessarily large tether diameter because power is transmitted to the ROV via the tether.

There are also ROVs that accomplish similar, yet different, results as the desired vehicle of this project. In addition to traditional invention, ROVs have been the subject matter of many collegiate competitions throughout the country. [8] is a report that describes one such competition; the purpose of this particular competition was to design an ROV capable of diving 12 meters, opening some sort of case, and attaching a cable into a port within the case. The competition ROV is classified as work-class (as opposed to observation-class) because it performs some function other than positioning a camera underwater. Another work-class ROV is described by [Wu]. This ROV is capable of operating light hydraulic equipment to perform various tasks (such as oil rig maintenance).

In conclusion, it is evident that there are many types of ROVs in the world used for a variety of reasons; however, there were no commercial ROVs that satisfied all of Mr. Jaskolski’s criteria. The purpose of this project, then, was to design a unique ROV capable of being transported through difficult terrain to capture video and images of unexplored, remote bodies of water. It is important to note that no applicable standards regarding ROVs were found during the background research for this paper.
Technical Breakdown

Electrical

Communication Interfaces

The underwater and surface systems use a variety of communication protocols. As NEMO is tethered to a computer by a fiber optic cable, there are two stages in the communication link where the RS232 data and NTSC video is encoded and decoded to a fiber optic signal. The RS232 data link between the computer and Arduino is bidirectional, so control data is sent from the computer to NEMO at the same time that the sensor data is sent by the microcontroller to the computer. The video can only be sent in a single direction due to the limitations of the fiber optic transceiver. The NTSC live feed video stream is connected directly from the GoPro Hero 2 camera to the fiber optic transceiver. It is decoded on the surface and processed by the computer.

Figure 1: Communication Interfaces
The human-machine interface is composed of a Microsoft Xbox 360 controller and a computer screen. The computer screen displays the video feedback from NEMO, the sensor data, and a system summary. The serial Xbox 360 controller data is read and processed by the computer and sent over the fiber optic data link. See Error! Reference source not found.. The serial control data received by the Arduino is then processed and converted into the proper pulse width modulation signals necessary to control the servos and the thruster. The Arduino also communicates with many different sensors and external modules. The communication types are broken down into four categories: Serial, I2C, oneWire, and Analog.

**Serial**

Serial communication is composed of two wires: TX and RX. TX is the signal in the outgoing channel, and RX is the incoming channel. When connecting to an external module using this communication protocol, a command is sent on the TX line and the data is returned on the RX line. In this application, serial communication occurs between the topside computer and the Arduino inside of NEMO.

**I2C**

I2C, or Inter-integrated Circuit, also features two wires: SDA and SCL. SDA is a bidirectional data line used to send and receive the data. SCL is a clock line that synchronizes the data transfer between the devices. I2C is essentially a master-slave setup. The master device sends a series of commands, typically an address, and the slave sends the contents of the address to the master. In NEMO, the master is the Arduino microcontroller and each sensor is the slave. The three axis digital magnetometer and accelerometer chip uses this communication protocol. To get the acceleration data, a method has been created called LSM303_read. It uses defined addresses, such as OUT_X_L_A, and requests the contents of each register. For each direction (X, Y, and Z) two bytes of data are stored on the digital compass, thus both the lower and higher registers must be read. The read function can be seen below. The input to the function is a single byte address. The function then receives the data from that address and returns it.

**oneWire**

The oneWire protocol is similar to I2C in that the data line is bidirectional. It differs in the fact that there isn’t a synchronizing clock signal. To measure the temperature of the water, a waterproof temperature sensor has been included on the sensor payload. It communicates using the oneWire protocol. Similar to the I2C protocol, it writes a one byte address on the serial port and waits for the return data on the same line.

**Analog**

The humidity sensor outputs an analog value depending on the humidity it is experiencing. To use this data transmission method, the output of the humidity sensor must be connected to one of the analog pins on the microcontroller. The Arduino features a 10-bit analog to digital converter so the resulting values are integers between 0 and 1023 where 0 corresponds to 0% humidity and 1023 corresponds to 100% humidity.
Surface Side Operations
The surface side portion of the project focused on ease of use, simplicity, and customization once the project becomes open-source. This lead to several design decisions: For instance, the code was determined to be written as an application that can be downloaded onto different machines without requiring a specific surface computing station. This allows for many different set-ups to be possible for surface operations.

The code was written in C++ in Visual Studio. This was chosen because C++ code is one of the most widely used coding languages with a solid support community that could assist any group in software customization of NEMO. Standard, the surface code can read in sensor data and display it on the GUI, display a full screen video of what the camera on NEMO is seeing, and supply NEMO with control information taken from the user.

Starting with the most top level portion on the surface side code, the graphical user interface is very simple and comprised of the full screen video for navigational purposes and sensor information displayed at the top including compass heading, depth reading, external temperature, and a warning if humidity gets above a safe value. For instance, if humidity is reading 90% there is a problem onboard NEMO and it will recommend the user to surface the vehicle immediately.

Below the GUI the software is computing values taken from the user by means of an Xbox controller. The Xbox controller is how the user controls the vehicle.

![Figure 2: Xbox Controller Diagram](image)

The joysticks on the controller read in values from -32766 to 32767. So for the left joystick, which controls thruster output, these values were scaled to 0-180 so that the Arduino would be able to directly send these values to the ESC without further computations. A value of 180 corresponds to operating the thruster at 100% (full forward), while a value of 0 corresponds to operating the thruster at -100% (full reverse).
It is important to note there that the strength of the thruster was much greater than anticipated and would move the vehicle too fast to be properly controlled so limits were placed within software to limit the output of the thruster. This limitation can be easily changed by re-defining certain values in the code so that the vehicle can be easily adapted for different situations. However, if the user does not have the capability to go into the code on the XBOX controller the right shoulder button will unlock thruster potential to a new level. For example, if the thruster was limited to 30% and the button was held down while using the joystick for thruster power, the new limit would raise to 50% power. This value can also be altered within the definitions allowing for further customization.

The ability to alter these values without going into the actual code and remaining in the definitions section allowed users not familiar with software to easily change these values.

The right joystick corresponds with the servo positions and allows the user to control the pitch and yaw of the vehicle. The controls are mapped intuitively. For instance, if I push the joystick left, computations are again done on the values to scale them to 0-180, and the servos will be told the steer the craft left. Each servo can actuate within a range of 0 to 180 where 0 corresponds to the neutral position minus 90 degrees (counterclockwise quarter turn) and 180 corresponds to the neutral position plus 90 degrees (clockwise quarter turn).

After reading in the state of the controller to determine what buttons and pushed and in the case of the joysticks, the extent with which they are pushed, the computed values are sent to NEMO and can be directly related to the servos limiting the amount of computations needing to be done on-board.

Reading in the sensors for the GUI to display also requires a slight trick. Since the sensors are being read in and displayed almost continuously, the surface code is actually reading faster than the Arduino can supply the sensor information. This results in errors being displayed on the GUI instead of the correct sensor information. To handle this problem, a simple ‘handshake’ occurs. The Arduino will first send ‘$’ before any sensor information. This means that the surface will only update the sensors when it gets ‘$’ and prevents any invalid updating when there is not sensor information to update.

The sensor information, once ‘$’ has been sent and received is then sent to the surface. This creates the problem of the surface not knowing how many numbers belong to each sensor. For instance, sometimes the heading will be 5° and sometimes it will be 270°. These varying values create a problem for how the code reads in these values since it is looking for the same number of digits per sensor every time it reads. To solve this, a value was added to the sensor value to create a stable bit number position and I then subtracted on the surface.

For the compass the maximum number of digits it can have is 3 with the highest value being 359°. The value 1000 is added to this before being sent up to the surface so that 1359 and 1005 have the same number of digits. The same value of 1000 is subtracted on the surface before further computations are done so that the sensors can be accurately read.

In order to have both of these sections of the code, the controller information being sent to the craft and the sensor information being sent to the user, operate continuously, the
code was threaded so that both functions could perform independent of the other section of code. This allows for faster communication and processing of sensors and controller data.

**Lights**

![Image of Light Diagram](image)

![Figure 3: Light Diagram](figure)

Lighting is also controlled by the user. There are 4 CREE LED’s onboard NEMO. Two of these are controlled by the left and right buttons the D-PAD and 2 are controlled by the up and down. The up and down buttons control the ‘navigational’ lights, or the center two LED’s with the up button increasing the brightness by 20% and the down button decreasing the brightness by 20%. The left and right buttons control the ‘ambient’ lights or the outer two LED’s on the light board.

The modular buttons have been left open for programming to input any go signal for modular on NEMO or any top-side operations that could be added like video mapping.

Although the majority of the surface code was written by the design team, several chunks of it were not. Serial.cpp is a pre-written file that allows for serial read and write developed by Microsoft to encourage programming, as was Serial.h. There are not dependent on running in Visual studio and should function on most commonly used operating systems. CXBOXController.cpp and CXBOXController.h were also found previously written and open-source via GitHub. These allow the Xbox controller to be read in easily not depending on a serial read. This helps simplify the software and using open-source components in the software increases the amount of support a user would have were he or she to go into the code to customize it for a specific mission. In order to process video the OpenCV library was utilized to create the simple GUI used to create NEMO. Like the pre-written libraries used for the Arduino code, OpenCV allowed easy and timelier coding of the GUI.

All code can be seen in Appendix G.

**Control**

The human machine interface is composed of a Microsoft Xbox 360 controller and a GUI. The user is able to control three aspects of the UROV: Pitch, Yaw, and Throttle. This control data is sent serially to the underwater microcontroller (Arduino Mega). At this point, the throttle and yaw can be directly written to the electronic speed controller and the yaw servo as the values do not need to be scaled or altered. Since both roll and pitch are affected by the side control surfaces, extra computations must be performed to determine the final Left and Right servo positions. The craft needs to be roll neutral at all times, thus the current roll must be measured by an Accelerometer and the resulting value is fed into a PID control system within the Arduino. See Figure 4: Control Feedback.
The internal control system accepts two inputs: current roll and desired pitch. Unfortunately, the current instantaneous roll is not easy to calculate. Initially, roll was determined by using a gyroscope; however, due to integration required to calculate roll, severe drift was introduced into the system. By utilizing an accelerometer, an instantaneous roll reading can be taken without the integration stage. Naturally, the instantaneous roll reading is not clean. It must be first passed through a low pass filter to remove the high frequency noise. The filtered roll reading is sent into the PID control system, and a corrective roll maneuver is calculated and added to the desired pitch to determine the final positions of the left and right servos. One of the design benefits of using an Arduino is the access to many premade libraries. The PID library, developed by Brett Beauregard, has been utilized in the implementation of the roll correction control system. The Kp, Ki, and Kd variables are the only inputs of the control system other than the desired roll (constant) and the current roll (from the accelerometer). These three values determine how aggressively NEMO will respond to a change in current roll.

Figure 4: Control Feedback

Figure 5: Control System
Sensors
There are four critical sensors currently onboard the UROV. These sensors are all displayed on the GUI to better help the user navigate and analyze the current environment of the UROV. All four sensors are connected to the Arduino mega. The sensor data is sent up to the control code through the fiber-optic cable in a single string of bits. This string of bits is lead by an identifier symbol: “$”. This identifier is used only to alert the control code that sensor data is received and ready to be displayed on the GUI. We decided to use this method of communication, because all of the sensors read at different rates, and by using this method we are insuring that the control code gets a packet of sensor data, rather than sporadic sensor readings whenever the individual sensors read.

Humidity
The humidity sensor we chose to use is from a company called SparkFun, out of Boulder, CO. This is the only sensor whose data is not constantly displayed on the GUI. The main use of this sensor is to detect any unwanted condensation within the vehicle. Hot electronics surrounded by cold water will create unwanted condensation within the UROV’s walls that could potentially ruin the electronics and communication between the vehicle and topside control. We have designed a custom desiccant pack to prevent condensation build up, but if this were to fail, the humidity sensor is there to alert the user possibly dangerous water levels inside of the vehicle. If the sensor detects a humidity/condensation threat, a warning notice will pop up in the GUI to alert the user.

This particular sensor has three ports attached to it. Two of those ports are VDD and GND, the other is the data line. It runs off of the 5V that the Arduino Mega puts out, and output information signal is connected to an analog port, that converts the humidity read into an integer value from 0-1023. This integer value is then divided by 1023 to get a percent humidity. This percent humidity is the value that is used to determine whether or not there is a condensation threat within the vehicle.

External Temperature
The external water temperature sensor is used for exactly what it name implies, to monitor the water temperature the UROV is currently in. This sensor is mounted on the exterior of the vehicle and attaches to the Arduino via the SeaCon connectors. It runs off of 5V and GND as well as a necessary 4.7kΩ pull-up resistor connected between VDD and the data line. The data signal uses the oneWire library described in the communication section and communicates through one of the digital ports on the Arduino Mega. The temperature is continuously updated and displayed on the GUI the user is using to operate the vehicle. We purchased this sensor from SparkFun out of Boulder, CO.

Pressure Transducer/Depth
Beckman Coulter Inc. generously donated this particular pressure transducer to our project. It is a Measurement Specialties sensor that is rated to 100psi. It works much like the humidity sensor described above. It is powered by the 5V and GND lines on the Arduino, while the data is read through one of the Arduino’s analog pins. However this particular sensor outputs a signal ranging from 0.5V-4.5V. Meaning that the entire 0-1023 integer range on the Arduino’s A/D converter is not used. However once the input signal is read and converted into an integer value, that integer value is used to identify the
there are a few components pulling most of the current from the battery:  

Thruster operating at 100%: **4.88A**

Fiber Optic Transceiver: **0.298A**

Servos: **1A** (with constant rapid actuation)

LED (x4) at 50%: **1.2A**

Estimated Operating Time: **1 hour and 52 minutes**

---

Compass

The compass on our UROV has dual purposes. It is a combination accelerometer/tilt-compensated compass. We use the accelerometer function of this particular sensor for the stability and control of the vehicle. The compass function is used to display the vehicle’s current heading to the user via the GUI. The sensor outputs a value from 0-360, which we then use to calculate the vehicle’s heading based on a standard compass breakdown. The GUI displays both the degree and heading (N, E, S, W, NE, SE, SW, NW) at any given time.

All of the sensors that we purchased for the vehicle (i.e. not donated sensors) were purchased from a company out of Boulder, Colorado called SparkFun. We decided to buy sensors from this particular company, as compared to other specialized sensor companies, because of cost. With one of the key requirements of this project being a minimal budget, we did not feel it necessary to spend extra money on expensive, more precise sensors, when the inexpensive alternatives would complete the job just fine.

Power Analysis

NEMO is powered by two 14.8V 5800mAh lithium polymer batteries operating in parallel. Because many of the components are operating at 5V rather than 14.8V, the voltage level must be dropped. NEMO originally used a linear voltage regulator. Because of the high input voltage level and the low desired output voltage level, the linear voltage regulator operated at approximately 33% efficiency. Due to the dark environment NEMO will be operating in, the high power LED module must also be included to illuminate the environment. These LEDs draw substantial current. See Figure 6: Low Efficiency Estimation chart below:
It is possible to increase the 14.8V to 5V conversion efficiency from 33% to 96% by using a standard Buck Converter. Only the 5V components will be affected by this change: fiber optic transceiver and three servos. Also, it is unlikely that the thruster will be operating at 100% during normal operation because of the intended use of the vehicle. A more likely thruster operation level of 25% has been listed below. The estimated operating time has been adjusted to a more reasonable estimate. See Error! Reference source not found. below.

<table>
<thead>
<tr>
<th>Component</th>
<th>Current (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thruster operating at 25%</td>
<td>2.44A</td>
</tr>
<tr>
<td>Fiber Optic Transceiver</td>
<td>0.105A</td>
</tr>
<tr>
<td>Servos</td>
<td>0.352A (with constant rapid actuation)</td>
</tr>
<tr>
<td>LED (x4)</td>
<td>1.2A</td>
</tr>
<tr>
<td>Estimated Operating Time</td>
<td>2 hours and 49 minutes</td>
</tr>
</tbody>
</table>

Figure 7: High Efficiency Estimation
Mechanical

Concept Selection

Initial Concepts

After the formation of the morphological chart, several initial design concepts were developed:

The three-thruster design allows for full range of motion of the device. The thruster on the tail would enable the ROV to tilt, allowing for unique video-capturing capabilities. Communication is achieved via a fiber optic tether. The body of the ROV and the front dome can be machined on a lathe. The electronics are located on a single board that can slide in and out of the removable back end of the device.

Figure 8: Three-thruster Design with One on Tail

Similar to the previous concept, this three-thruster design allows for full range of motion of the ROV. Water-tightness of the vehicle is achieved using the double O-ring design mentioned in Table 3. This design features a handle for easy carrying and rails to protect the device and the 360° camera. The GoPro camera is used to provide HD live-streaming video and a ring of LEDs provides ambient lighting while LED spotlights provide distant lighting. The body and dome can be machined on a lathe, motors mount to the body, and the electronics are located in the rear housing. However, the two pressure housing design may not be practical or desirable.

Figure 9: Conventional Three-thruster Design
This four-thruster design is very similar to the Seabotix model used as the datum in Table 4. Four thrusters provide the full range of motion of the vehicle, but add a considerable weight load to the system. The body of this device would have to be custom-built and, therefore, would be rather expensive. While it is a neat design, it did not seem practical for the purposes of this project.

Figure 11: Single-thruster, Canard-style Design

Figure 12 shows a concept similar to that of Figure 9 but with a single, continuous body due to the vertical thruster being located in the plastic handle. All other features (camera, lighting, waterproofing, etc.) are the same as the conventional three-thruster design and, therefore, could be manufactured similarly. The handle, however, would have to be custom-built due to its complex shape. This could be accomplished with 3D printing.

Figure 12 shows a single-thruster concept that utilizes Canard-style controls (i.e. control surfaces on the front of the vehicle). These control surfaces provide the vehicle with a full
range of motion and the single-thruster design drastically increases the operable time of
the device compared to three-thruster designs. This design also utilizes the GoPro camera,
LED lighting, and waterproofing of the previous concepts. This concept could also be
machined on a lathe and the electronics would be located on a single board that slides
into the vehicle through the removable back end. The Canard surfaces would be
controlled with waterproofed servo motors located outside of the pressure housing.
Multiple sizes of control surfaces could be utilized depending on the current payload of
the vehicle (i.e. whether the 360° camera is attached or not).

Figure 13 shows another Canard-control
design, similar to that shown in Figure 11. The
distinguishing feature of this concept is a dual-
chamber body. The outer chamber will serve
as a mounting surface for the control surface
servo motors while the inner chamber serves
as the pressure housing for the electronics. The
upper half of the gap between the two
chambers will serve as a ballast tank. This area
will take in water for the device to sink.
Conversely, several CO₂ tanks (like those
used in paintball guns) will force the water out
for the device to rise (used as an abort feature
if communication is lost). This concept will be
manufactured in a similar fashion as the
previous design.

Finally, Figure 10 shows a two-thruster
concept. This vehicle achieves a full range of
motion through two fully-rotational thrusters
mounted on either side of the vessel. It
features the same waterproofing, live-stream
camera, and LED lighting as previous
concepts. Also, this concept can be produced
in a similar manner as the previous concepts
(machined on a lathe). Before material is
selected for this, or any, concept, calculations
must be made; the material’s strength, weight,
and machinability are all important
considerations in making the final material
selection.
First Round Pugh Analysis

Five design concepts were compared against a datum (Seabotix LBV150-4) in the first round Pugh analysis; the results are summarized in Figure 15 below. Concepts 1-4 rated better than the datum for lightweight because they all have fewer than four thrusters. All concepts rated lower than the datum for operating time because the datum is powered from the surface whereas all concepts are powered onboard the vehicle (and will eventually need to be recharged). Concept 3 is the only concept that rated better than the datum for compact size because it only has one thruster (in-line with the body) and the control surfaces are removable; all other concepts have three or more thrusters making any size variation from the datum negligible. Similarly, Concept 3 is the only design concept that rated better than the datum in terms of production cost because it utilizes a single thruster. Concept 4 rated lower than the datum for maneuverability because, given the position of the rear thruster, the thrusters would have to counteract one another to achieve certain motion (i.e. going up or down). Concept 3 rated the best for low sediment disturbance (which is important for video quality near the bottom of lakes, etc.) because it only has one thruster and it is axially aligned with the body. Concept 3 also rated the best in terms of ease of maintenance because of its simple design and single thruster. Finally, Concepts 1-4 all rated better aerodynamically than the datum because they have more streamlined designs.

Table 1: Round One Pugh Analysis

<table>
<thead>
<tr>
<th>Criteria</th>
<th>DATUM: Seabotix LBV150-4</th>
<th>Concept 1: Three-thruster (Figure 5)</th>
<th>Concept 2: Handle thruster (Figure 7)</th>
<th>Concept 3: Canard-controls (Figure 8)</th>
<th>Concept 4: Tail thruster (Figure 4)</th>
<th>Concept 5: Seabotix style (Figure 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightweight</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Operating time</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Compact size</td>
<td>S</td>
<td>S</td>
<td>+</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Production cost</td>
<td>S</td>
<td>S</td>
<td>+</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Maneuverability</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>-</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Low sediment disturbance</td>
<td>S</td>
<td>S</td>
<td>+</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Ease of maintenance</td>
<td>S</td>
<td>S</td>
<td>+</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Aerodynamic</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td><strong>TOTAL POSITIVES (+)</strong></td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>TOTAL NEGATIVES (-)</strong></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

It can be seen from Table 1 that Concept 3 has six positives and only one negative comparison against the datum. Therefore, Concept 3 is the best design concept from the
first round Pugh analysis and will become the datum design for the second round Pugh analysis.

**Second Round Pugh Analysis**

Three new design concepts were compared against the new datum (Canard-controls) in the second round Pugh analysis; the results are summarized in Table 2 below. Concept 2 is the only design concept that rated lower than the datum for lightweight because it utilizes two thrusters as opposed to one. Concept 1 rated better than the datum for operating time because its CO$_2$ ballast system will allow for a “dead drop”; the ROV could be dropped in the water, take water into the ballast tank, sink (unpowered) until a certain depth is reached, wake up, blow water out of the ballast tank with CO$_2$, and stabilize its buoyancy. Concept 2 was worse than the datum for operating time and compact size because it utilizes two thrusters instead of one. All design concepts rated lower than the datum for production cost and ease of maintenance because they all have extra features (a CO$_2$ ballast, a second thruster, and a balloon ballast, respectively). Concept 2 rated better than the datum in maneuverability because its two-thruster design allows for zero turns; however, it rated lower than the datum for low sediment disturbance because its thrusters will kick up a high amount of sediment near the bottom of bodies of water. Finally, Concepts 2 and 3 rated lower than the datum for aerodynamic because the side-mounted thrusters of Concept 2 and the balloon ballast of Concept 3 result in greater drag for those designs.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>DATUM: Canard-controls (Figure 8)</th>
<th>Concept 1: Canard, CO$_2$ ballast (Figure 9)</th>
<th>Concept 2: Two-thruster (Figure 10)</th>
<th>Concept 3: Canard, balloon ballast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightweight</td>
<td>S</td>
<td>-</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>Operating time</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Compact size</td>
<td>S</td>
<td>-</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>Production cost</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Manoeuverability</td>
<td>S</td>
<td>+</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>Low sediment disturbance</td>
<td>S</td>
<td>-</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>Ease of maintenance</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Aerodynamic</td>
<td>S</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TOTAL POSITIVES (+)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TOTAL NEGATIVES (-)</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

It can be seen from Table 2 that the datum and Concept 1 scored similarly. However, this particular matrix does not properly account for the advantages of the ballast system of Concept 1; therefore, a third round, with different judging criteria, will be conducted between the datum and Concept 1.
Third Round Pugh Analysis

The third round Pugh analysis is a re-comparison of the original Canard-control design and the Canard-control design featuring a CO2 ballast system. Any criterion that ranked “same” between the two designs in the previous round was replaced with a criterion that factors in the advantages of a ballast system. A summary of the results can be found in Table 3 below. Concept 1 ranked better than the datum for hover ability, retrievable, and abort capability due to its CO2 ballast system. The datum would have to naturally be slightly buoyant to return to the surface if the tether broke; this reduces its ability to “hover” to capture images of its surroundings with the 360° camera. Because the buoyancy of Concept 1 can be dynamically controlled with the CO2 ballast system, it is more readily retrievable than the datum design and it can support an abort feature to retrieve the device if something goes wrong.

Table 3: Round Three Pugh Analysis

<table>
<thead>
<tr>
<th>Criteria</th>
<th>DATUM: Canard-controls (Figure 8)</th>
<th>Concept 1: Canard, CO2 ballast (Figure 9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating time</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Production cost</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Hover ability</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Retrievable</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Abort capability</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Ease of maintenance</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>TOTAL POSITIVES (+)</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>TOTAL NEGATIVES (-)</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Table 3 shows that Concept 1 has a slight edge over the datum after considering the new judging criteria. Therefore, the Canard-control design featuring a CO2 ballast system was regarded as the best ROV design at this stage in the design phase; unfortunately, a CO2 ballast system was too complicated to implement in the allotted time. Instead, NEMO was made with slightly positive buoyancy.

Final Concept

Geometric Modeling

The computer-aided design (CAD) program SolidWorks was used to model all of the vehicle components. Many of these part models were then sent to Solid Concepts as their primary reference for creating the components via additive manufacturing. Details pertaining to material selection and manufacturability can be found later in this report.
Figure 15, below, shows the complete NEMO CAD model. The orange external shell serves as a mounting surface for the three control surfaces located at the front of the vehicle, the thruster located at the rear of the vehicle, and any external modules. The pressure housing is the white tube located within the external shell. It is composed of a tube, two end domes, and four housing rods. The GoPro camera is visible behind the front dome; it is supported by the electronics rack that is located within the pressure housing.

The pressure housing model can be seen below in Figure 16. As mentioned above, it consists of a tube, two end domes, and four housing rods. A cylindrical shape was chosen to maximize the volume and pressure integrity of the vehicle.
Figure 16: Pressure Housing Model

The CAD model of the external shell is shown in Figure 17, below. The external shell is the second chamber of the dual-chamber design. It acts as a mounting surface for the thruster, control surfaces, and any external modules. By incorporating a mounting chamber to the design, the student team increased the pressure integrity of NEMO.

Figure 17: External Shell Model

Figures 18 and 19 illustrate the model of the electronics sled. The electronics sled was designed to organize and support all of the vehicle’s electronics (batteries, camera, circuitry, etc.), as shown in Figure 19. The sled was designed such that it contours to the inside surface of the pressure housing, providing a solid and secure surface for the electronics.
The model of the front dome is illustrated by Figure 20. The front dome has two key features: (1) the outer ring, which acts as a mounting surface for the external shell, and (2) the inner extrusion, which creates a double piston seal with the pressure housing tube and two Buna-N O-rings, thus sealing the pressure housing from water, dirt, etc.
In addition to the features of the front dome, the rear dome (Figure 21) is where the bulkhead connectors (three electrical, one fiber optic) penetrate the pressure housing. The mounting surfaces for the bulkhead connectors are visible in Figure 21, below. The rear dome was designed thicker than the front dome to accommodate the four breaches for the connectors.

The control surface model can be seen in Figure 22. The control surfaces provide directional control of NEMO via redirection of force. The student team wanted the control surfaces to be easily removable for quick replacement, exchanging of sizes, etc. Therefore, the control surfaces were designed with a snapping mechanism (visible in Figure 22). The snapping mechanism allows for quick removal and attachment of the control surfaces, yet maintains a secure fit to the servo motor.
Figure 23 shows the CAD model of the thruster used on NEMO. A strong thruster was selected because of the single-thruster design of the vehicle. By utilizing only one thruster, the team minimized the size, weight, and power consumption of the vehicle.

Finally, the LED housing model is illustrated in Figure 24, below. The team decided to mount the LEDs to the exterior vehicle to dissipate their high heat output and to eliminate light diffraction within the front dome. The LED housing will mount to the bottom of the vehicle, and it will provide both ambient and cruising light for video and navigational purposes.
Feasibility
A feasibility analysis was necessary before the team could proceed with the single-thruster, control surface concept. Much of this feasibility analysis consisted of mathematical modeling and analysis, which can be found later in this report. The most beneficial means of feasibility testing, however, was the development of a prototype. The prototype was geometrically similar to the final concept, and provided assurance of design and proof of concept to the student team. The prototype was constructed of PVC piping and Plexiglas, featured the same thruster as the final concept, and was controlled via the same methods as the final concept. Further details on prototype testing can be found later in this report. After the mathematical analysis and prototype testing, the design was deemed feasible by the team and Mr. Jaskolski and NEMO was developed.

Failure Modes and Effects Analysis (FMEA)
The failure modes that are considered to be most detrimental to the final vehicle, the effects of those failures, and plans for mitigation are listed below:

1. **O-ring failure**: Any failure of an O-ring between the pressure housing and the end domes of the vehicle would be catastrophic. Failure of an O-ring would break the watertight seal of the ROV and allow water into the interior of the pressure housing, likely destroying the electrical components of the vehicle. In an effort to prevent this failure mode, all O-rings will be maintained and exchanged with new parts as necessary. Furthermore, a system of three O-rings will be used on each end of the pressure housing to provide three layers of water protection.

2. **Pressure housing damage**: Any damage to the pressure housing will sacrifice the integrity of the vehicle. The main pressure housing and the rear dome are well protected by the outer shell of the vehicle; however, the front dome is exposed and vulnerable to damage. Therefore, the entire pressure housing (but especially the front dome) will be thoroughly examined for damage after any activity. Several spare domes and pressure housings will be manufactured in case any replacements are required.
3. **Servo failure:** Servo failure would disable the user from being able to fully control the vehicle. While this would not be a catastrophic failure, the vehicle would not perform as desired; therefore, this failure needs to be prevented. To mitigate servo failures, the servos will be waterproofed using marine epoxy and mineral oil. Furthermore, the servos are well protected by the outer shell of the vehicle. Finally, many spare servos will be on hand for any necessary replacement.

4. **Thruster failure:** Thruster failure would fully disable the vehicle. Again, this would not be a catastrophic failure, but the vehicle would not perform as desired; therefore, this failure needs to be mitigated. To prevent thruster failure, all regular maintenance suggested by the manufacturer will be performed. The thruster is well-protected by the outer shell of the vehicle, decreasing the chances of any physical damage to the part. Also, a spare thruster has been purchased in case replacement is required.

5. **Tether damage:** Tether (the fiber-optic cable) damage would fully disable the vehicle and would make the vehicle difficult to retrieve. In order to prevent tether damage, a mesh will be added around the thruster. Also, fishing line will be added to the tether to strengthen it and allow the vehicle to be pulled back to the user via the tether. Finally, a spare tether will be purchased in case replacement is required.

6. **Control surface damage:** If a control surface is damaged or destroyed during operation, some loss of control of the vehicle will occur. This failure is not at all catastrophic because the vehicle can be retrieved and easily repaired. To help mitigate this failure, the team designed the control surfaces such that they are very securely fastened to the servo. The team also ordered several spare control surfaces should one be damaged or destroyed in operation.

**Detail Design (Explanation, Material Selection, and Manufacturing and Assembly Considerations)**

**Overview**

As stated above, NEMO features a double-chamber design, single thruster, and three control surfaces. The double-chamber design allows the thruster and servo motors to be mounted to the vehicle without sacrificing its pressure integrity. The single thruster and three control surfaces provide full directional control of NEMO in a small, lightweight, and power-efficient package. See below for further details of the individual components.

**Pressure Housing**

Figure 25 depicts the fully-assembled pressure housing. It consists of an ultra-high molecular weight (UHMW) polyethylene body, cast urethane front and rear domes, four aluminum housing rods, eight corrosion-resistant fasteners, four Buna-N O-rings, three electrical bulkhead connectors, and one fiber optic bulkhead connector. Analysis suggests that the pressure housing can withstand pressures at depths greater than 190m in fresh water (calculations can be found later in the report).
As stated above, the pressure-housing vessel is made of UHMW polyethylene; an image of the vessel can be seen in Figure 26 below. UHMW was chosen for its high strength-to-weight ratio, extreme temperature tolerance, and machinability. The UHMW vessel was machined on a lathe by the design team.

The four aluminum-housing rods were also machined on a lathe by the design team. Aluminum was chosen for its high strength-to-weight ratio and corrosion resistance. The rods serve to keep compression on the pressure housing at low pressures; they also assist in resisting buckling at high pressures.

The front and rear domes were manufactured by Solid Concepts by using a combination of processes. First, molds of the domes were generated via additive manufacturing. Next, patterns of the domes were created with the molds. Finally, the domes were cast with urethane. Cast urethane was chosen for its optical clarity, high impact strength, and repeatability. Both domes feature an extrusion that holds two Buna-N O-rings and slides into the UHMW vessel. These double-seal extrusions ensure that no water enters the pressure housing.
The front dome (Figure 27, above) simply serves as a window for capturing video; however, the rear dome (Figure 28, below) is a bit more complex. All of the bulkhead connectors (three electrical, one fiber optic) penetrate the vehicle at the rear dome. Because of this, the rear dome was cast thicker than the front dome. The bulkhead connectors were purchased from SeaCon and are rated to depths far greater than NEMO can reach.

Electronics Sled

Figure 29 shows the electronics sled. The purpose of the electronics sled is to house and organize all of the vehicle’s electronics (two lithium polymer batteries, fiber optic transceiver, Arduino Mega, thruster electronic stability controller, and circuitry) within the pressure housing; it also offers some cable management. The electronics sled was manufactured by Solid Concepts utilizing additive manufacturing (specifically, fused deposition modeling). Additive manufacturing was used because of the complex geometry of the electronics sled and the lightweight, yet sturdy nature of the plastic.
**External Shell**

Figure 30, below, shows the external shell with the thruster and control surfaces mounted. As mentioned above, the purpose of the external shell is to minimize the number of breaches in the pressure housing by providing a mounting surface for the servo motors and thruster; the external shell also protects the pressure housing by acting as a barrier between the pressure housing and any external agents (i.e. rocks, the bottom of a lake, etc.). The external shell was manufactured by Solid Concepts via additive manufacturing.

Another function of the external shell is to provide a mounting surface for any external modules. Figure 31, below, shows the picatinny rail system that was designed into the external shell to account for such modules. The picatinny rail system allows for quick attachment and removal of all modules. If, for some reason, the desired module does not fit into the picatinny rails, it can be attached to NEMO via screws and the two rows of holes surrounding the picatinny rails.
Control Surfaces

Finally, an image of one of the control surfaces can be seen in Figure 32, below. The control surfaces enable directional control of NEMO via redirection of force. The control surfaces were manufactured by Solid Concepts using additive manufacturing. The control surfaces simply snap on and off of the servos allowing for simple exchanging of new surfaces, different sizes, etc.
**Mathematical Modeling and Supporting Analysis**

It is critical to have analysis that confirms that NEMO will exceed the design requirements. This supporting analysis can be found below:

**Pressure Stress and Wall Thickness Analysis**

An arbitrary value of 0.25" for the wall thickness of the pressure housing was originally chosen by the design team when generating the CAD model of the vehicle. The following hand calculations were performed to validate that value:

- Can the ROV be treated as a thin-walled pressure vessel?
  \[
  \frac{Wall\ Thickness}{Inner\ Diameter} = \frac{0.25"}{5.0"} = 0.05 < 0.1
  \]
  So, the thin-walled assumption holds. However, tanks under external pressure fail by buckling (i.e. collapse), not by yielding. Therefore, they should not be designed using the simplistic formulas commonly used for thin-walled tanks under internal pressure. Instead, the hull should be designed as a thick-walled pressure vessel.

- For external pressure alone on a thick-walled pressure vessel:
  \[
  \sigma_{t,max} = \frac{-p_o(r_o^2 + r_i^2)}{r_o^2 - r_i^2}
  \]
  \[
  \sigma_{r,max} = -p_o
  \]
  Where \(\sigma_t\) is tangential stress, \(\sigma_r\) is radial stress, \(p_o\) is the external pressure acting on the vessel, \(r_o\) is the outer radius of the vessel, and \(r_i\) is the inner radius of the vessel.

- As mentioned above, the hull will be constructed of UHMW-PE
  - Elastic Modulus (E) of the material: 0.69 GPa
  - Yield Strength (S_y) of the material: 21 MPa

- Critical load for buckling:
  \[
  p_{cr} = \frac{\pi^2 EI}{l^2}
  \]

- Moment of Inertia (I) of a hollow cylinder:
  \[
  I = \frac{\pi}{64}[(d_o)^4 - (d_i)^4]
  \]
  - For the ROV hull:
    \[
    I = \frac{\pi}{64}[(0.1397m)^4 - (0.127m)^4] = 5.93 \times 10^{-6} m^4
    \]

- Critical stress for buckling:
  \[
  \sigma_{cr} = \frac{p_{cr}}{A}
  \]
  - For the ROV hull:
    \[
    \sigma_{cr} = \frac{p_{cr}}{A} = \frac{\pi^2 EI}{l^2 A} = \frac{2\pi EI}{l^2(r_o^2 - r_i^2)} = \frac{2\pi(0.69 \times 10^9)(5.93 \times 10^{-6})}{0.3048^2(0.06985^2 - 0.0635^2)} = 326 MPa
    \]
    - However, this value exceeds the yield strength of the material (21 MPa); therefore, proceed with 21 MPa for the calculations

- Rearranging the equation for tangential stress, substituting for pressure, and solving for depth yields:
Rearranging the equation for radial stress, substituting for pressure, and solving for depth yields:

\[ h = \frac{\sigma_{r,max}(r^2 - r_i^2)}{\rho g (r^2 + r_i^2)} - \frac{p_{atm}}{\rho g} \]

\[ = \frac{(21 \times 10^6)[0.06985^2 - 0.0635^2]}{(1000)(9.81)[0.06985^2 + 0.0635^2]} - \frac{1.013 \times 10^5}{(1000)(9.81)} = 193.2 \text{ meters} \]

- Using the lesser of the two calculated depths, the UHMW-PE hull (neglecting the domes) can reach a depth of \(~190\ \text{meters}\).
- This means that the 0.25” pressure housing wall thickness provides a factor of safety of \(~3.2\).

The following results were collected from an FEA analysis of the pressure housing:

Figure 33, below, depicts the static nodal stresses acting on the pressure housing at a depth of 60m. The strongest stress intensity acting on the pressure housing at 60m is less than 10 MPa, much less than the yield strength of the material (21 MPa).

Below, Figure 34 shows the resultant deformation of the pressure housing at a depth of 60m. The greatest deformation of the pressure housing occurs near the domes and has a magnitude of about 0.025”. This value is not a concern and suggests that the pressure housing will be quite robust to depths well below the desired value of 60m.
By looking at Figures 33 and 34, it can be seen that the body of the pressure housing will yield long before the domes. This is the case because buckling will be the failure mode under external pressure. The body will buckle before the domes because it has a longer and more slender geometry than the domes. The design team is very satisfied with the estimated achievable depth of 190m; this value is three times greater than the depth requested by Mr. Jaskolski.

**Pressure Testing of NEMO**

The design team has been unable to pressure test the final vehicle by this time. However, the team is confident that NEMO can attain depths of 60m because of what the analysis and testing included in this report suggests. That being said, it is still very important to pressure test the vehicle prior to any field use, but the team was unable to manufacture a second pressure housing for testing due to unfortunate circumstances. The team is currently discussing options for pressure testing with each other and Mr. Jaskolski.

**Buoyancy Analysis**

The SolidWorks model was used to retrieve the following approximate dimensions:

- Volume of the UHMW vessel: 291.35 in$^3$
- Volume of the domes: 110.54 in$^3$
- Volume of the housing rods: 5.44 in$^3$
- Volume of the thruster: 8.99 in$^3$
- Volume of the control surfaces: 10.63 in$^3$
- Volume of the external shell: 60.24 in$^3$
• Miscellaneous volume: 5.00 in\(^3\)

This yields a total approximate vehicle volume of: \(492.19 \text{ in}^3 = 0.00807 \text{ m}^3\)

According to Archimedes’ Principle, the buoyant force is equal to the weight of the displaced water. This can be expressed by:

\[
F_B = V \times \rho_{\text{water}} \times g
\]

Therefore:

\[
F_B = 0.00807 \text{ m}^3 \times 1000 \frac{kg}{m^3} \times 9.81 \frac{m}{s^2} = 79N
\]

This means, for NEMO to be neutrally buoyant, it must weigh:

\[
\frac{79N}{9.81 \frac{m}{s^2}} = 8.05 \text{ kg} = 17.75 \text{ lb}
\]

**Buoyancy Testing**
In addition to the above buoyancy analysis, the design team has performed several buoyancy tests to ensure that NEMO is slightly positively buoyant. Mr. Jaskolski wants the vehicle to be slightly positively buoyant so that it will float to the water’s surface (in a lake) should something go wrong. See Appendix E for the Initial Buoyancy Test Plan.

**Waterproofing and Pressure Testing of the Servo Motors**
Several pressure tests were conducted on the servo motors to confirm that they can withstand the required pressures of the project. These pressure tests were performed in a pressure chamber located at Mr. Jaskolski’s office. After these tests, the design team is confident that the HiTec servo motors (model HS-5086WP) will remain fully-operational at depths up to 60m. See Appendix D for the Servo Motor Pressure Test Plan. The servo pressure tests can be viewed at:

- [http://www.youtube.com/watch?v=QvTyrSFWmN4](http://www.youtube.com/watch?v=QvTyrSFWmN4)
- [http://www.youtube.com/watch?v=ACsyoxJ1bR8](http://www.youtube.com/watch?v=ACsyoxJ1bR8)

**Fatigue Analysis**
Fatigue did not have to be considered because it is not an issue due to the absence of any tensile stress cycle. As [11] states, “As a submarine hull is subjected to fluctuating compressive stresses, fatigue problems [are not] expected (57)”.

**Control Surface Stress Analysis**
The design team did not perform physical stress testing on the control surfaces for several reasons: (1) if a control surface was damaged or lost during operation, the vehicle could still be retrieved and fixed (non-catastrophic failure), (2) due to the geometry of the
control surface, it tends to rotate when impacted rather than breaking off, (3) the team did not have enough spare parts to conduct a useful stress test. However, it is still important to have an estimate of how much force the control surfaces can withstand. The team is currently working on hand calculations and an FEA analysis of maximum fin stresses.

**Proof of Concept Utilizing a Prototype**

A fully operational prototype was created early in the project to prove the functionality of the single-thruster, control surface design. The prototype was constructed of PVC pipe and Plexiglas. It was pool-tested several times, and the design team was very pleased with its functionality and maneuverability. By developing the prototype, the team was reassured that its design was practical and would fulfill the needs of Mr. Jaskolski. That being said, the development of a prototype was critical to the success of the project. The prototype pool tests can be viewed at:

- [http://www.youtube.com/watch?v=SWlion303_0](http://www.youtube.com/watch?v=SWlion303_0)
- [http://www.youtube.com/watch?v=3jVEWzIQsqI](http://www.youtube.com/watch?v=3jVEWzIQsqI)
- [http://www.youtube.com/watch?v=5V9QrHo4Hbk](http://www.youtube.com/watch?v=5V9QrHo4Hbk)

**Testing**

**Pool Tests**

Now that buoyancy testing has been completed, the design team plans on conducting several pool tests with NEMO similar to those that were performed with the prototype. However, a couple electronics issues need to be corrected before pool testing can take place.

**Field Tests**

After pool testing is complete, Mr. Jaskolski would like to take the design team to Chasm Lake to conduct a field test with NEMO. Chasm Lake is an alpine lake in Estes Park, CO that has yet to be explored. It would be an ideal testing location because NEMO would have to be backpacked to the lake, and it provides the opportunity to potentially make new discoveries.

**Safety Considerations**

Several safety precautions regarding NEMO have been taken:

- A warning label was included on the thruster to prevent potential injuries to extremities
- A safety mesh for the thruster has been constructed to prevent injuries to extremities and damage to the fiber optic tether
- The backpack for the vehicle was selected with ergonomics in mind
- The team is looking into designing an ergonomic strap for carrying the Pelican case
Ethical Issues

This project does raise several ethical questions that were considered during the design process of the vehicle. Since NEMO will be exploring delicate underwater environments, the ethics of trespassing in such a place must be examined. Every safety measure has been taken so that NEMO has a minimal impact on any underwater environment. There are no fluids that could leak into the water and contaminate the ecosystem, the underwater signals are fiber-optic so there is less actual electricity flowing through the water as to not disrupt species, and the navigational system has been limited to a single thruster so that the force from the thruster will not disturb the waterbed. The system has also been made slightly positively buoyant so that were the craft to fail, it would float to the surface instead of remaining in the ecosystem to contaminate it. These are just some of the design choices that were made to be able to ethically explore new areas without disturbing the environment.

An ethical dilemma that appeared in the middle of design was found because the thruster was significantly more powerful than originally intended. This was scaled in software so that the thruster wouldn’t disrupt the lakebed but the potential remained and when brought to the customer’s attention, the customer brought up the idea of being able to retrieve bodies with the vehicle. Every year people die cave diving, which is unsurprising considering it is one of the most dangerous activities people can do. Sadly, because it is so dangerous, when cave divers go missing their bodies are left in the caves. The ability to retrieve the bodies by use of a remote operated vehicle could grant the families of the lost solace. This is ethically ambiguous because the recovery would have to be handled with care and consider different religious backgrounds. Since the leading option to be able to actually recover bodies would be the use of a sort of hook, it might not be the most delicate operation attempting the hook the body onto the vehicle.

There is also the issue of what is to be done when something of value is discovered by NEMO. To bring it to the surface would definitely disrupt the current ecosystem but the fossils or artifacts that could be discovered would provide researchers with new information and understanding about a variety of historical or sociological things. Once the project is released, due to the open-source nature, there will no longer be any control over how this is handled. Also, the ability of the craft to handle modularity for different water samples or lighting, etc. would also have no control. If a potentially hazardous module were attached, the vehicle would become dangerous to the environment. The only limit that the design team could really place on this was a limited voltage and current provided by the vehicle in the hopes that the modularity would only be used for further learning.

Overall this project is seen to be a positive ethical contribution to the world, granting the ability to explore to natives of a country who might not have been able to previously. There are ethical dilemmas but we believe that the project is still sound ethically and will positively contribute to the worlds understanding without harming underwater environments.
Conclusions and Future Work

In order to determine the success of the project, the design objectives and constraints must be evaluated. A summary of the project status can be found in Table 4, below.

Objective 1: Dive Depth

A target attainable depth of 60m was set at the beginning of the project. While pressure testing of the final vehicle has not been completed yet, the design team is confident that NEMO can achieve depths of at least 60m. As mentioned previously, the servo motors have been fully tested and proven operational to depths of 60m, the thruster is rated to over 100m, and analysis of the pressure housing suggests it can withstand pressures at depths greater than 190m. However, it is still important to pressure test the final vehicle; that being said, Objective 1 is likely satisfied but still in progress.

Objective 2: Operable Time

Mr. Jaskolski requested operable times of 30-60 minutes. This objective has been satisfied by utilizing two lithium polymer batteries for power and a single thruster for movement. Analysis and testing shows that NEMO can consistently operate for 90-120 minutes on a single charge, offering 2-3 times the amount of requested operable time.

Objective 3: Size of Topside

A total target maximum volume of 8,000 in$^3$ was established at the beginning of the project; therefore, 4,000 in$^3$ was allotted to both the topside equipment and the vehicle itself. The topside equipment measures 2,820 in$^3$. The entire system (topside equipment and the vehicle) measures 6,820 in$^3$. Therefore, Objective 3 has been satisfied.

Objective 4: Size of Vehicle

The vehicle measures 4,000 in$^3$ (when contained in the backpack). As mentioned above, the entire system measures 6,820 in$^3$. Therefore, Objective 4 has been satisfied.

Objective 5: Temperature Tolerance

Mr. Jaskolski wanted a vehicle that could withstand temperatures ranging from -10$^\circ$C to 60$^\circ$C. In order to fulfill this objective, the design team selected materials that maintain desired properties (i.e. do not become brittle, etc.) within a wide range of temperatures.

Objective 6: Weight of Equipment

A total target maximum weight of 80 lbs was established at the beginning of the project; therefore, 40 lbs was allotted to both the topside equipment and the vehicle itself. The topside equipment weighs 22 lbs, well within the 40 lb maximum. The entire system (topside equipment and vehicle) weighs 40 lbs, well below the 80 lb limit. Therefore, Objective 6 has been satisfied.
Objective 7: Weight of Vehicle

The vehicle weighs 18 lbs, far below the 40 lb maximum. As mentioned above, the entire system weighs 40 lbs. So, Objective 7 has been satisfied.

Constraint 1: Backpackable and Robust

The vehicle and topside equipment needs to be backpackable and robust because it will be hiked through difficult environments and possibly checked as luggage on an airliner. The design team ensured that NEMO was backpackable by restricting its size and weight. For robustness, the team designed the topside equipment to fit inside a Pelican case. A foam liner was created for the vehicle itself, and the vehicle and foam liner will be transported via a hiking backpack. That being said, Constraint 1 has been satisfied.

Constraint 2: Budget

The design team was allocated $10,000 by Mr. Jaskolski for the completion of one complete underwater remotely-operated vehicle. The team met this constraint by only spending $7,012. See Appendix B for detailed budget information.

Constraint 3: Modular Capability

To ensure that NEMO is capable of supporting modular payloads, the team designed a picatinny rail structure on the underside of the vehicle to support various modules. The possibility of adding modular payloads was also accounted for in the software. Therefore, Constraint 3 has been satisfied.

Constraint 4: Onboard Power

This constraint was met by utilizing two lithium polymer batteries onboard NEMO for power.

Constraint 5: Time

Although it is fast approaching, it appears that NEMO will be fully functional and ready for service by the May 10th deadline. So, Constraint 5 is likely satisfied but still in progress. A final fully integrated test on location must still be completed.
### Table 4: Project Status

<table>
<thead>
<tr>
<th>Objective or Constraint</th>
<th>Name</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective 1</strong></td>
<td>Dive Depth</td>
<td>Likely satisfied, but In-Progress</td>
</tr>
<tr>
<td><strong>Objective 2</strong></td>
<td>Operable Time</td>
<td>Satisfied</td>
</tr>
<tr>
<td><strong>Objective 3</strong></td>
<td>Size of Topside</td>
<td>Satisfied</td>
</tr>
<tr>
<td><strong>Objective 4</strong></td>
<td>Size of Vehicle</td>
<td>Satisfied</td>
</tr>
<tr>
<td><strong>Objective 5</strong></td>
<td>Temperature Tolerance</td>
<td>Satisfied</td>
</tr>
<tr>
<td><strong>Objective 6</strong></td>
<td>Weight of Equipment</td>
<td>Satisfied</td>
</tr>
<tr>
<td><strong>Objective 7</strong></td>
<td>Weight of Vehicle</td>
<td>Satisfied</td>
</tr>
<tr>
<td><strong>Constraint 1</strong></td>
<td>Backpackable and Robust</td>
<td>Satisfied</td>
</tr>
<tr>
<td><strong>Constraint 2</strong></td>
<td>Budget</td>
<td>Satisfied</td>
</tr>
<tr>
<td><strong>Constraint 3</strong></td>
<td>Modular Capacity</td>
<td>Satisfied</td>
</tr>
<tr>
<td><strong>Constraint 4</strong></td>
<td>Onboard Power</td>
<td>Satisfied</td>
</tr>
<tr>
<td><strong>Constraint 5</strong></td>
<td>Time</td>
<td>Likely satisfied, but In-Progress</td>
</tr>
</tbody>
</table>

**Manufacturing and Assembly**

The only parts manufactured by the design team are the UHMW vessel, the aluminum housing rods, and the LED housing. These parts were simple to manufacture, and they could easily be reproduced. Due to the complex geometries of the other parts, the design team decided to have them 3D printed by Solid Concepts. While these parts were free for this project due to a full sponsorship, Solid Concepts estimates that an external shell, electronics sled, two nose domes, and three control surfaces would normally cost $7,500. This value is low compared to currently available ROVs, but it is still a large amount of money. Some of the components could be simplified and produced in-house should an end-user wish to reduce this cost.

The design team wanted it to be simple for an end-user to assemble, disassemble, and fix the vehicle. To incorporate this desire, the team made several design decisions:

- The electronics sled houses the electronics so they are centrally-located and easy to access and remove
- The control surfaces snap on and off of the servo motors quickly and easily
- The servo motors are located such that they are easy to access and remove
- All bulkhead connectors are accessible with the vehicle fully assembled
- The picatinny rail system makes attaching and accessing external modules quick and easy

That being said, it still takes a fair amount of time to assemble and disassemble NEMO due to the number of fasteners required to fully secure the vehicle. However, the number of fasteners was unavoidable due to the design of the vehicle.
Testing and Refinement
As previously mentioned, several pool tests were conducted with the prototype. The team gained a lot of insight from these tests, and was able to make necessary refinements to the prototype. These refinements included weight balance, waterproofing improvements, strengthening of materials, and electrical adjustments. Similar discoveries and refinements will be made when the team begins pool testing NEMO.

The team has been able to test and refine the buoyancy of the final vehicle by performing several buoyancy tests. From the buoyancy tests, the team discovered that weight needed to be added to certain parts of the vehicle to make it stable and neutrally buoyant.

Finally, by conducting several waterproofing and pressure tests of the servo motors, the team was able to choose a servo and develop a waterproofing method that withstands pressures at depths of 60m. This refinement process was critical to the success of the project by ensuring that the vehicle will be operable at the defined objective depths.

Project/Development Cost Summary
As mentioned in Design Constraint 2, Mr. Jaskolski allocated $10,000 to the student team for the production of one backpackable, remotely-operated underwater vehicle. A summary of the project costs can be found in Table 5, below. For a detailed breakdown of the project budget, refer to Appendix B.

<table>
<thead>
<tr>
<th>Table 5: Summary of Project Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Budget:</td>
</tr>
<tr>
<td>Cost of Vehicle Parts:</td>
</tr>
<tr>
<td>Cost of Prototype Parts:</td>
</tr>
<tr>
<td>Cost of Miscellaneous Electrical:</td>
</tr>
<tr>
<td>Cost of Miscellaneous Mechanical:</td>
</tr>
<tr>
<td>Total Cost:</td>
</tr>
<tr>
<td>Remaining Funds:</td>
</tr>
</tbody>
</table>

Conclusions and Recommendations
The design team is very satisfied with what they achieved this year. While it remains uncertain whether NEMO will be field-tested prior to the end of the school year, the team and Mr. Jaskolski believe that the project was a success. The project started out as an idea, and the student team created a vehicle that is capable of being backpacked into a location, exploring dangerous waters, making new discoveries, and saving human lives. Mr. Jaskolski has said that NEMO and the student team have greatly surpassed his expectations; while that alone does not define project success, it is encouraging to hear and says a lot about Colorado State University.
The student team enjoyed the interdisciplinary nature of this project. It is beneficial to work with students of different majors because it simulates a real working environment, allows for the exchange of different perspectives on the project, and can result in a more sophisticated final product. Therefore, the student team recommends more interdisciplinary senior design projects in the future (preferably, with a few different majors represented in a group).

The team kept the project open-source with the hope of NEMO becoming the benchmark for a portable and affordable remotely-operated underwater vehicle. Ideally, Mr. Jaskolski will develop a small fleet of NEMOs in conjunction with National Geographic. Then, because NEMO is open-source, other academic institutions can develop improvements and modules for NEMO, making it much more realistic to explore, map, sample, photograph, etc. the dangerous waters mentioned at the beginning of this report. Should this be a continuation project next year, the team would recommend exploring the development of modules for NEMO (water sampler, soil sampler, robotic arm, etc.), developing several different sizes of control surfaces for different payloads, strengthening the external shell, protecting the control surfaces from impact, developing an ergonomic strap for the Pelican case, and exploring ways to speed up assembly and disassembly of the vehicle.
Bibliography


Appendix A: Project Abbreviations

UROV: Underwater Remote Operated Vehicle
NEMO: Nautical Exploratory Modular Observer
GUI: Graphical User Interface
## Appendix B: Budget

<table>
<thead>
<tr>
<th>Date</th>
<th>Category</th>
<th>Description</th>
<th>Quantity</th>
<th>Total Cost</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/28/2012</td>
<td>Vehicle Parts</td>
<td>Arduino Mega</td>
<td>2</td>
<td>$134.35</td>
<td></td>
</tr>
<tr>
<td>8/31/2012</td>
<td>Vehicle Parts</td>
<td>Thruster</td>
<td>1</td>
<td>$918.17</td>
<td></td>
</tr>
<tr>
<td>9/5/2012</td>
<td>Vehicle Parts</td>
<td>LiPo Battery</td>
<td>1</td>
<td>$165.00</td>
<td></td>
</tr>
<tr>
<td>9/5/2012</td>
<td>Vehicle Parts</td>
<td>LiPo Battery Charger</td>
<td>1</td>
<td>$119.95</td>
<td></td>
</tr>
<tr>
<td>9/6/2012</td>
<td>Vehicle Parts</td>
<td>Fiber-Optic Cable (2m)</td>
<td>1</td>
<td>$24.39</td>
<td></td>
</tr>
<tr>
<td>9/6/2012</td>
<td>Miscellaneous</td>
<td>Bullet Connectors</td>
<td>1</td>
<td>$8.12</td>
<td></td>
</tr>
<tr>
<td>9/6/2012</td>
<td>Prototype Parts</td>
<td>Thumb Joystick and Breakout Board</td>
<td>1</td>
<td>$9.71</td>
<td></td>
</tr>
<tr>
<td>9/11/2012</td>
<td>Vehicle Parts</td>
<td>GoPro Hero 2</td>
<td>1</td>
<td>$308.69</td>
<td></td>
</tr>
<tr>
<td>9/11/2012</td>
<td>Miscellaneous</td>
<td>Toggle Switch and Heat Shrink</td>
<td>1</td>
<td>$8.26</td>
<td></td>
</tr>
<tr>
<td>9/12/2012</td>
<td>Vehicle Parts</td>
<td>XBOX Controller</td>
<td>1</td>
<td>$32.19</td>
<td></td>
</tr>
<tr>
<td>9/13/2012</td>
<td>Vehicle Parts</td>
<td>LiPo Battery Bag</td>
<td>1</td>
<td>$15.24</td>
<td></td>
</tr>
<tr>
<td>9/14/2012</td>
<td>Vehicle Parts</td>
<td>External Temperature Sensor</td>
<td>1</td>
<td>$9.95</td>
<td></td>
</tr>
<tr>
<td>9/14/2012</td>
<td>Vehicle Parts</td>
<td>Humidity Sensor</td>
<td>1</td>
<td>$16.95</td>
<td></td>
</tr>
<tr>
<td>9/14/2012</td>
<td>Vehicle Parts</td>
<td>Tilt Compensated Compass</td>
<td>1</td>
<td>$29.95</td>
<td></td>
</tr>
<tr>
<td>9/14/2012</td>
<td>Vehicle Parts</td>
<td>Internal Pressure Sensor</td>
<td>1</td>
<td>$24.95</td>
<td>Not Used</td>
</tr>
<tr>
<td>9/14/2012</td>
<td>Vehicle Parts</td>
<td>RCS-BNC Connectors</td>
<td>1</td>
<td>$2.88</td>
<td></td>
</tr>
<tr>
<td>9/14/2012</td>
<td>Vehicle Parts</td>
<td>USB-RCA</td>
<td>1</td>
<td>$9.94</td>
<td></td>
</tr>
<tr>
<td>9/14/2012</td>
<td>Vehicle Parts</td>
<td>Fiber-Optic Transceiver</td>
<td>1</td>
<td>$410.00</td>
<td></td>
</tr>
<tr>
<td>9/15/2012</td>
<td>Vehicle Parts</td>
<td>GoPro Cable</td>
<td>1</td>
<td>$30.81</td>
<td></td>
</tr>
<tr>
<td>9/17/2012</td>
<td>Vehicle Parts</td>
<td>MAX3232 Breakout</td>
<td>1</td>
<td>$9.76</td>
<td></td>
</tr>
<tr>
<td>9/21/2012</td>
<td>Prototype Parts</td>
<td>Prototype Body and Components</td>
<td>1</td>
<td>$36.91</td>
<td></td>
</tr>
<tr>
<td>9/22/2012</td>
<td>Miscellaneous</td>
<td>Solder, Wire, Multimeter</td>
<td>1</td>
<td>$46.13</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Category</td>
<td>Description</td>
<td>Quantity</td>
<td>Price</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>---------------------</td>
<td>--------------------------------------------------</td>
<td>----------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>9/22/2012</td>
<td>Miscellaneous</td>
<td>Male Header, Heat Shrink</td>
<td>1</td>
<td>$6.01</td>
<td></td>
</tr>
<tr>
<td>9/24/2012</td>
<td>Vehicle Parts</td>
<td>Thruster</td>
<td>1</td>
<td>$901.57</td>
<td></td>
</tr>
<tr>
<td>9/25/2012</td>
<td>Vehicle Parts</td>
<td>Fiber-Optic Connector</td>
<td>1</td>
<td>$15.20</td>
<td></td>
</tr>
<tr>
<td>9/27/2012</td>
<td>Miscellaneous</td>
<td>Liquid Tape, Toggle Switch, Heat Shrink</td>
<td>1</td>
<td>$19.20</td>
<td></td>
</tr>
<tr>
<td>9/27/2012</td>
<td>Prototype Parts</td>
<td>Thumb Joystick and Breakout Board</td>
<td>1</td>
<td>$9.71</td>
<td></td>
</tr>
<tr>
<td>9/30/2012</td>
<td>Miscellaneous</td>
<td>Silicon</td>
<td>1</td>
<td>$4.27</td>
<td></td>
</tr>
<tr>
<td>10/1/2012</td>
<td>Miscellaneous</td>
<td>Radio Shack</td>
<td>1</td>
<td>$15.04</td>
<td></td>
</tr>
<tr>
<td>10/1/2012</td>
<td>Miscellaneous</td>
<td>Radio Shack</td>
<td>1</td>
<td>$3.74</td>
<td></td>
</tr>
<tr>
<td>10/5/2012</td>
<td>Vehicle Parts</td>
<td>LiPo Battery Bag</td>
<td>1</td>
<td>$14.24</td>
<td></td>
</tr>
<tr>
<td>10/5/2012</td>
<td>Vehicle Parts</td>
<td>Multi-Mode Patch Cable</td>
<td>1</td>
<td>$33.55</td>
<td></td>
</tr>
<tr>
<td>10/5/2012</td>
<td>Prototype Parts</td>
<td>Arduino Nano</td>
<td>1</td>
<td>$23.88</td>
<td></td>
</tr>
<tr>
<td>10/6/2012</td>
<td>Vehicle Parts</td>
<td>Laptop</td>
<td>1</td>
<td>$16.10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corey Paid for most of the laptop</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10/7/2012</td>
<td>Miscellaneous</td>
<td>ACE Hardware</td>
<td>1</td>
<td>$20.47</td>
<td></td>
</tr>
<tr>
<td>10/7/2012</td>
<td>Prototype Parts</td>
<td>Plexiglass</td>
<td>1</td>
<td>$21.44</td>
<td></td>
</tr>
<tr>
<td>10/8/2012</td>
<td>Miscellaneous</td>
<td>Hobby Drill, 1-72 Tap</td>
<td>1</td>
<td>$26.29</td>
<td></td>
</tr>
<tr>
<td>10/9/2012</td>
<td>Miscellaneous</td>
<td>1-72 Tap</td>
<td>2</td>
<td>$12.12</td>
<td></td>
</tr>
<tr>
<td>10/10/2012</td>
<td>Miscellaneous</td>
<td>Black Wire</td>
<td>1</td>
<td>$8.05</td>
<td></td>
</tr>
<tr>
<td>10/11/2012</td>
<td>Miscellaneous</td>
<td>Perfboard</td>
<td>1</td>
<td>$3.75</td>
<td></td>
</tr>
<tr>
<td>10/11/2012</td>
<td>Miscellaneous</td>
<td>Servo Extension, 18in</td>
<td>1</td>
<td>$4.83</td>
<td></td>
</tr>
<tr>
<td>10/11/2012</td>
<td>Miscellaneous</td>
<td>Marine Epoxy, Silicon, Caulk Gun</td>
<td>1</td>
<td>$14.17</td>
<td></td>
</tr>
<tr>
<td>10/11/2012</td>
<td>Miscellaneous</td>
<td>1-72 Tap</td>
<td>1</td>
<td>$38.95</td>
<td></td>
</tr>
<tr>
<td>10/11/2012</td>
<td>Miscellaneous</td>
<td>IC-Pos, Harness</td>
<td>1</td>
<td>$9.05</td>
<td></td>
</tr>
<tr>
<td>10/12/2012</td>
<td>Miscellaneous</td>
<td>1-72 Tap</td>
<td>1</td>
<td>$10.40</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Category</td>
<td>Item Description</td>
<td>Quantity</td>
<td>Price</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------------------</td>
<td>-------------------------------------------------------</td>
<td>----------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>10/14/2012</td>
<td>Prototype Parts</td>
<td>Arduino Nano</td>
<td>1</td>
<td>$32.02</td>
<td></td>
</tr>
<tr>
<td>10/15/2012</td>
<td>Vehicle Parts</td>
<td>Waterproof Switch (PT)</td>
<td>1</td>
<td>$13.20</td>
<td></td>
</tr>
<tr>
<td>10/19/2012</td>
<td>Vehicle Parts</td>
<td>Breakaway Male Header</td>
<td>1</td>
<td>$6.44</td>
<td></td>
</tr>
<tr>
<td>10/19/2012</td>
<td>Miscellaneous</td>
<td>Zinc Washers, Black Screen Spline, Hardware Cloth</td>
<td>1</td>
<td>$16.03</td>
<td></td>
</tr>
<tr>
<td>10/19/2012</td>
<td>Prototype Parts</td>
<td>Soft Weights</td>
<td>3</td>
<td>$28.82</td>
<td></td>
</tr>
<tr>
<td>11/12/2012</td>
<td>Vehicle Parts</td>
<td>Breadboard Mini Self-Adhesive</td>
<td>2</td>
<td>$11.77</td>
<td></td>
</tr>
<tr>
<td>11/12/2012</td>
<td>Vehicle Parts</td>
<td>CP2102 Mini USB to Serial</td>
<td>1</td>
<td>$31.77</td>
<td></td>
</tr>
<tr>
<td>11/14/2012</td>
<td>Prototype Parts</td>
<td>Gyroscope</td>
<td>1</td>
<td>$29.90</td>
<td></td>
</tr>
<tr>
<td>2/2/2013</td>
<td>Vehicle Parts</td>
<td>UHMW-PE Rod</td>
<td>1</td>
<td>$120.92</td>
<td></td>
</tr>
<tr>
<td>2/5/2013</td>
<td>Vehicle Parts</td>
<td>O-Rings and Screws</td>
<td>1</td>
<td>$25.58</td>
<td></td>
</tr>
<tr>
<td>2/7/2013</td>
<td>Vehicle Parts</td>
<td>Aluminum Rod</td>
<td>1</td>
<td>$37.11</td>
<td></td>
</tr>
</tbody>
</table>
| 2/11/2013  | Vehicle Parts     | Solid Concepts Body and Spares                        | 1        | $0.00  | Solid Concepts gave the team a full sponsorship
| 3/1/2013   | Vehicle Parts     | Pressure Transducer                                   | 1        | $0.00  | Donated by Beckman Coulter, Inc.
<p>| 3/13/2013  | Vehicle Parts     | Arduino Mega Shield                                   | 1        | $14.95 |
| 3/28/2013  | Vehicle Parts     | LiPo Battery                                          | 1        | $155.94|
| 3/28/2013  | Vehicle Parts     | LiPo Battery Bag                                      | 1        | $10.99 |
| 3/28/2013  | Vehicle Parts     | Fiber-Optic Cable (60m)                               | 2        | $148.16|
| 3/28/2013  | Miscellaneous     | FedEx (Mail UHMW to Solid Concepts)                   | 1        | $65.27 |
| 4/3/2013   | Vehicle Parts     | Backpack                                              | 1        | $64.36 |
| 4/4/2013   | Vehicle Parts     | Pelican Case                                          | 1        | $125.00|
| 4/5/2013   | Vehicle Parts     | Video Capture Device                                  | 1        | $34.99 |
| 4/8/2013   | Vehicle Parts     | Silica Gel Desiccant Pack                             | 1        | $19.99 |</p>
<table>
<thead>
<tr>
<th>Date</th>
<th>Category</th>
<th>Description</th>
<th>Quantity</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/8/2013</td>
<td>Vehicle Parts</td>
<td>Fasteners and Thread Rod</td>
<td>1</td>
<td>$67.14</td>
</tr>
<tr>
<td>4/8/2013</td>
<td>Miscellaneous</td>
<td>Michael Reimbursement</td>
<td>1</td>
<td>$41.72</td>
</tr>
<tr>
<td>4/8/2013</td>
<td>Miscellaneous</td>
<td>Assembly Materials</td>
<td>1</td>
<td>$89.84</td>
</tr>
<tr>
<td>4/12/2013</td>
<td>Miscellaneous</td>
<td>McMaster-Carr</td>
<td>1</td>
<td>$12.25</td>
</tr>
<tr>
<td>4/13/2013</td>
<td>Miscellaneous</td>
<td>Solder, Resistors</td>
<td>1</td>
<td>$11.25</td>
</tr>
<tr>
<td>4/17/2013</td>
<td>Vehicle Parts</td>
<td>Right Angle Plug</td>
<td>1</td>
<td>$9.55</td>
</tr>
<tr>
<td>4/18/2013</td>
<td>Miscellaneous</td>
<td>Glue for E-Days Poster</td>
<td>1</td>
<td>$2.15</td>
</tr>
<tr>
<td>4/18/2013</td>
<td>Miscellaneous</td>
<td>Kinko’s (Binding for E-Days Booklets)</td>
<td>1</td>
<td>$11.79</td>
</tr>
<tr>
<td>4/18/2013</td>
<td>Miscellaneous</td>
<td>Wire, Heat Shrink, Epoxy, Fasteners</td>
<td>1</td>
<td>$86.04</td>
</tr>
<tr>
<td>4/18/2013</td>
<td>Miscellaneous</td>
<td>Poster Supplies</td>
<td>1</td>
<td>$1.50</td>
</tr>
<tr>
<td>4/18/2013</td>
<td>Miscellaneous</td>
<td>Poster Board</td>
<td>1</td>
<td>$9.58</td>
</tr>
<tr>
<td>4/18/2013</td>
<td>Miscellaneous</td>
<td>Poster Supplies</td>
<td>1</td>
<td>$2.15</td>
</tr>
<tr>
<td>4/18/2013</td>
<td>Miscellaneous</td>
<td>Ramtech</td>
<td>1</td>
<td>$20.93</td>
</tr>
<tr>
<td>4/19/2013</td>
<td>Vehicle Parts</td>
<td>Seacon Connectors</td>
<td>4</td>
<td>$1,345.79</td>
</tr>
<tr>
<td>4/26/2013</td>
<td>Miscellaneous</td>
<td>McMaster-Carr</td>
<td>1</td>
<td>$62.27</td>
</tr>
<tr>
<td>Multiple</td>
<td>Vehicle Parts</td>
<td>Servos</td>
<td>12</td>
<td>$438.82</td>
</tr>
</tbody>
</table>

**Total Budget:** $10,000.00

Cost of Vehicle Parts: $5,969.00
Cost of Prototype Parts: $192.39
Cost of Miscellaneous Electrical: $718.56
Cost of Miscellaneous Mechanical: $131.56

**Total Cost:** $7,011.51

**Remaining Funds:** $2,988.49

Hydro-Technologies very generously supplied all funding for this project. However it was Solid Concepts donation and acquired partnership that allowed us to build the vehicle under budget. By supplying our team with all the necessary 3-D printed parts at
no cost, we were able to come in well under budget at only $7,011.51. Completing NEMO at a fraction of what other UROV’s on the market cost.

Completing this project under our original budget of $10,000 was critical for this particular project. The customer’s hopes as well as our own, is that this vehicle will be made available to international institutions and universities that are under funded. Some of the most interesting unexplored caves are located in remote and impoverished areas of the world. Our hopes is that by engineering this inexpensive UROV, these communities and nations will be able to explore and learn about the caves that have been to dangerous for humans to enter.
## Appendix C: Project Plans

<table>
<thead>
<tr>
<th>Project</th>
<th>Student</th>
<th>Cost</th>
<th>Start</th>
<th>Finish</th>
<th>January</th>
<th>February</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research and Development</td>
<td>Rachel, Sarah, Justin, Luke, Michael</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LED’s</td>
<td>Justin, Michael</td>
<td>1.23.13</td>
<td>2.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Navigation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensors</td>
<td>Rachel</td>
<td>1.23.13</td>
<td>2.14.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Underwater pressure/Depth</td>
<td>Rachel</td>
<td>1.23.13</td>
<td>2.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal temperature</td>
<td>Rachel</td>
<td>1.23.13</td>
<td>2.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gyroscope</td>
<td>Sarah</td>
<td>1.23.13</td>
<td>2.14.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCB</td>
<td>Rachel, Sarah, Justin</td>
<td>1.23.13</td>
<td>3.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design in by</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surface mount sensors</td>
<td>company in denver</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attach to arduino</td>
<td>Rachel, Sarah, Justin</td>
<td>1.23.13</td>
<td>2.12.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convent connections</td>
<td>Justin</td>
<td>2.13.13</td>
<td>3.15.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transmitters/ receivers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All around easy assembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUI</td>
<td>Jacob, Sarah</td>
<td>1.23.13</td>
<td>3.31.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perfected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hovering Mode</td>
<td>Justing</td>
<td>1.23.13</td>
<td>3.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Website Design</td>
<td>Rachel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td>Rachel, Sarah, Justin</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensors</td>
<td>Rachel, Sarah</td>
<td>1.23.13</td>
<td>2.14.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pressure casing</td>
<td>Luke, Michael</td>
<td>2.16.13</td>
<td>3.15.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timing Analysis</td>
<td>Justin, Sarah, Rachel</td>
<td>1.23.13</td>
<td>3.15.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Body test (i.e. Mechanical system test)</td>
<td>Luke, Michael</td>
<td>2.16.13</td>
<td>3.15.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pool Tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alpine Lake Tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>Student</td>
<td>Cost</td>
<td>Start</td>
<td>Finish</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------</td>
<td>--------------</td>
<td>-------</td>
<td>--------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research and Development</td>
<td>Rachel, Sarah, Justin, Luke, Michael</td>
<td>1.23.13</td>
<td>2.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LED's</td>
<td>Rachel, Sarah, Justin, Luke, Michael</td>
<td>1.23.13</td>
<td>2.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ambient</td>
<td>Justin, Michael</td>
<td>1.23.13</td>
<td>2.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Navigation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensors</td>
<td>Rachel</td>
<td>1.23.13</td>
<td>2.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Underwater pressure/Depth</td>
<td>Rachel</td>
<td>1.23.13</td>
<td>2.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal temperature</td>
<td>Rachel</td>
<td>1.23.13</td>
<td>2.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gyroscope</td>
<td>Sarah</td>
<td>1.23.13</td>
<td>2.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PC3</td>
<td>Rachel, Sarah, Justin</td>
<td>1.23.13</td>
<td>3.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design in by</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Surface mount sensors</td>
<td>company in design</td>
<td>1.23.13</td>
<td>2.12.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convenient connections</td>
<td>Justin</td>
<td>2.13.13</td>
<td>3.15.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transmitter/recivers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All around easy assembly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUI</td>
<td>Jacob, Sarah</td>
<td>1.23.13</td>
<td>3.31.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perfected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hovering Mode</td>
<td>Justing</td>
<td>1.23.13</td>
<td>3.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Website Design</td>
<td>Rachel</td>
<td>1.23.13</td>
<td>3.1.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td>Rachel, Sarah</td>
<td>1.23.13</td>
<td>2.14.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensors</td>
<td>Rachel</td>
<td>1.23.13</td>
<td>2.14.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pressure casing</td>
<td>Luke, Michael</td>
<td>2.16.13</td>
<td>3.15.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timing Analysis</td>
<td>Justin, Sarah, Rachel</td>
<td>1.23.13</td>
<td>3.15.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Body test (i.e. Mechanical system test)</td>
<td>Luke, Michael</td>
<td>2.16.13</td>
<td>3.15.13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pool Tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alpine Lake Tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>Student</td>
<td>Cost</td>
<td>Start</td>
<td>Finish</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>-------</td>
<td>-----------</td>
<td>-----------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Student</th>
<th>Cost</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Research/Preliminary</strong></td>
<td>Rachel, Sarah, Justin, Luke, Michael</td>
<td>$0</td>
<td>8.22.12</td>
<td>10.23.12</td>
</tr>
<tr>
<td><strong>Electrical Design</strong></td>
<td>Justin, Sarah, Rachel</td>
<td>3,550</td>
<td>8.22.12</td>
<td>9.7.12</td>
</tr>
<tr>
<td><strong>Mechanical (Final Concept)</strong></td>
<td>Luke and Michael</td>
<td></td>
<td>8.22.12</td>
<td>10.23.12</td>
</tr>
<tr>
<td><strong>Pressurized casing</strong></td>
<td>Luke and Michael</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Motors</strong></td>
<td>Luke and Michael</td>
<td>4,560</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sensors</strong></td>
<td>Rachel</td>
<td>$100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Processor (Arduino)</strong></td>
<td>Justin</td>
<td>4,560</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Batteries</strong></td>
<td>Justin</td>
<td>3,550</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fiberoptic Cable</strong></td>
<td>Justin</td>
<td>2,530</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GoPro Camera</strong></td>
<td>Justin</td>
<td>2,5200</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Lights</strong></td>
<td>Sarah</td>
<td>$100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Top Side Equipment</strong></td>
<td>Sarah and Rachel</td>
<td>$100</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>360 Camera</strong></td>
<td>Justin</td>
<td>$0</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Website Design</strong></td>
<td>Rachel</td>
<td>$0</td>
<td>8.31.12</td>
<td>9.19.12</td>
</tr>
<tr>
<td><strong>Design/Testing</strong></td>
<td>Justin, Sarah, Rachel</td>
<td></td>
<td>9.17.12</td>
<td>10.17.12</td>
</tr>
<tr>
<td><strong>Communication</strong></td>
<td>Justin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sensors</strong></td>
<td>Rachel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Top side implementation</strong></td>
<td>Sarah</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Motor operation</strong></td>
<td>Luke and Michael</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Pressure casing</strong></td>
<td>Luke and Michael</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Arduino Programming</strong></td>
<td>Justin, Sarah, Rachel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GUI</strong></td>
<td>Sarah</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fiberoptic Cable</strong></td>
<td>Justin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mechanical Analysis and Feasibility</strong></td>
<td>Luke and Michael</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Implementation</strong></td>
<td>Justin, Sarah, Rachel</td>
<td>10,24.12</td>
<td>11.16.12</td>
<td>11.29.12</td>
</tr>
<tr>
<td><strong>PCB Design</strong></td>
<td>Justin, Sarah, Rachel</td>
<td>10,10.12</td>
<td>12.16.12</td>
<td>10.29.12</td>
</tr>
<tr>
<td><strong>Working Schematic to send in</strong></td>
<td>Justin, Sarah, Rachel</td>
<td>10,10.12</td>
<td>11.16.12</td>
<td>10.29.12</td>
</tr>
<tr>
<td><strong>Populate PCB Board</strong></td>
<td>Justin, Sarah, Rachel</td>
<td>11,12.12</td>
<td>12.30.12</td>
<td></td>
</tr>
<tr>
<td><strong>Test</strong></td>
<td>Justin, Sarah, Rachel</td>
<td>11,12.12</td>
<td>11.30.12</td>
<td></td>
</tr>
<tr>
<td><strong>Insure box is ready for electronics</strong></td>
<td>Sarah, Luke, Lisa, Mike, Michael</td>
<td>11,29.12</td>
<td>12.7.12</td>
<td></td>
</tr>
<tr>
<td><strong>Mechanical Prototype Fabrication</strong></td>
<td>Luke and Michael</td>
<td>1,16.13</td>
<td>2,15.13</td>
<td></td>
</tr>
<tr>
<td><strong>Preliminary Testing of prototype</strong></td>
<td></td>
<td>11,16.12</td>
<td>12.7.12</td>
<td></td>
</tr>
<tr>
<td><strong>Dry tests</strong></td>
<td>Rachel, Sarah, Justin, Luke, Michael</td>
<td>11,26.12</td>
<td>12.7.12</td>
<td></td>
</tr>
<tr>
<td><strong>Wet tests</strong></td>
<td>Rachel, Sarah, Justin, Luke, Michael</td>
<td>1,16.13</td>
<td>3,1.13</td>
<td></td>
</tr>
<tr>
<td><strong>Field Testing</strong></td>
<td></td>
<td>2,15.13</td>
<td>4,12.13</td>
<td></td>
</tr>
<tr>
<td><strong>Pool Tests</strong></td>
<td></td>
<td>2,15.13</td>
<td>3,1.13</td>
<td></td>
</tr>
<tr>
<td><strong>Alpine Lake Tests</strong></td>
<td></td>
<td>3,1.13</td>
<td>4,12.13</td>
<td></td>
</tr>
<tr>
<td><strong>Fine tuning/complete functionality</strong></td>
<td></td>
<td>1,16.13</td>
<td>4,12.13</td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>Student</td>
<td>Cost</td>
<td>Start</td>
<td>Finish</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------------</td>
<td>-------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td><strong>Underwater Rover Project Plan</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research/Preliminary</td>
<td>Rachel, Sarah, Justin, Luke, Michael</td>
<td>$0</td>
<td>8.22.12</td>
<td>10.23.12</td>
</tr>
<tr>
<td>Electrical Design</td>
<td>Justin, Sarah, Rachel</td>
<td>$0</td>
<td>8.22.12</td>
<td>10.23.12</td>
</tr>
<tr>
<td>Mechanical (Final Concept)</td>
<td>Luke and Michael</td>
<td>$0</td>
<td>8.22.12</td>
<td>10.23.12</td>
</tr>
<tr>
<td>Fiberoptic Cable</td>
<td>Justin</td>
<td>$250</td>
<td>8.22.12</td>
<td>9.14.12</td>
</tr>
<tr>
<td>GoPro Camera</td>
<td>Justin</td>
<td>$0</td>
<td>8.22.12</td>
<td>9.14.12</td>
</tr>
<tr>
<td>360 Camera</td>
<td>Justin</td>
<td>$0</td>
<td>8.22.12</td>
<td>9.14.12</td>
</tr>
<tr>
<td><strong>Website Design</strong></td>
<td>Rachel</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
</tr>
<tr>
<td><strong>Design/Testing</strong></td>
<td>Justin, Sarah, Justin</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
</tr>
<tr>
<td>Communication</td>
<td>Justin</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
</tr>
<tr>
<td>Top side Implementation</td>
<td>Sarah</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
</tr>
<tr>
<td>Arduino Programming</td>
<td>Justin, Sarah, Justin</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
</tr>
<tr>
<td>GUI</td>
<td>Sarah</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
</tr>
<tr>
<td>Implementation</td>
<td>Justin, Sarah, Rachel</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
</tr>
<tr>
<td>PCB Design</td>
<td>Justin, Sarah, Rachel</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
</tr>
<tr>
<td>Working Schematic to send in</td>
<td>Justin, Sarah, Rachel</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
</tr>
<tr>
<td>Populate PCB Board</td>
<td>Justin, Sarah, Rachel</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
</tr>
<tr>
<td>Test</td>
<td>Justin, Sarah, Rachel</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
</tr>
<tr>
<td>Field Testing</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
<td>8.31.12, 10.19.12</td>
</tr>
<tr>
<td>Alpine Lake Tests</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
<td>8.31.12, 10.19.12</td>
</tr>
<tr>
<td>Fine tuning/complete functionality</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.19.12</td>
<td>8.31.12, 10.19.12</td>
</tr>
<tr>
<td>Project</td>
<td>Student</td>
<td>Cost</td>
<td>Start</td>
<td>Finish</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------------------</td>
<td>------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Underwater Rover Project Plan</td>
<td></td>
<td>$5,250</td>
<td>8.20.12</td>
<td>4.12.13</td>
</tr>
<tr>
<td>Research/Preliminary</td>
<td>Rachel, Sarah, Austin, Luke, Matthew</td>
<td>$0</td>
<td>8.22.12</td>
<td>10.23.12</td>
</tr>
<tr>
<td>Electrical Design</td>
<td>Justin, Sarah, Rachel</td>
<td>$0</td>
<td>8.22.12</td>
<td>9.7.12</td>
</tr>
<tr>
<td>Mechanical (Final Concept)</td>
<td>Luke and Michael</td>
<td>$0</td>
<td>8.22.12</td>
<td>10.33.12</td>
</tr>
<tr>
<td>Pressurized casing</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Motors</td>
<td>Luke and Michael</td>
<td>4($600)</td>
<td>8.31.12</td>
<td>10.15.12</td>
</tr>
<tr>
<td>Sensors</td>
<td>Rachel</td>
<td>$100</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Processor (Arduino)</td>
<td>Justin</td>
<td>4($600)</td>
<td>9.17.12</td>
<td>10.17.12</td>
</tr>
<tr>
<td>Batteries</td>
<td>Justin</td>
<td>3($350)</td>
<td>9.17.12</td>
<td>10.17.12</td>
</tr>
<tr>
<td>Fiber optic cable</td>
<td>Justin</td>
<td>2($430)</td>
<td>9.17.12</td>
<td>10.17.12</td>
</tr>
<tr>
<td>GoPro Camera</td>
<td>Justin</td>
<td>2($200)</td>
<td>9.17.12</td>
<td>10.17.12</td>
</tr>
<tr>
<td>Lights</td>
<td>Sarah</td>
<td>$100</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Top side equipment</td>
<td>Sarah and Rachel</td>
<td>$200</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>360 Camera</td>
<td>Justin</td>
<td>$50</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Website Design</td>
<td>Rachel</td>
<td>$0</td>
<td>8.31.12</td>
<td>10.15.12</td>
</tr>
<tr>
<td>Design/Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td>Justin, Sarah, Rachel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensors</td>
<td>Rachel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top side implementation</td>
<td>Sarah</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Motor operation</td>
<td>Luke and Michael</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pressure casing</td>
<td>Luke and Michael</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arduino Programming</td>
<td>Justin, Sarah, Rachel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GUT</td>
<td>Sarah</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fiber optic cable</td>
<td>Justin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mechanical Analysis and Feasibility</td>
<td>Luke and Michael</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCB Design</td>
<td>Justin, Sarah, Rachel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working Schematic to send in</td>
<td>Justin, Sarah, Rachel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Populate PCB Board</td>
<td>Justin, Sarah, Rachel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>Justin, Sarah, Rachel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Waterproof Test rig for Electrical Testing</td>
<td>Luke and Michael</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insure box is ready for electronics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.25.12</td>
<td>11.25.12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mechanical Prototype Fabrication</td>
<td>Luke and Michael</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preliminary Testing of prototype</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dry tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wet tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field Testing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pool Tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alpine Lake Tests</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix D: Servo Motor Pressure Test Plan

Test Date: April 6, 2013
Creator: Luke Stahler
Version: 1

Pressure Testing of the Servo Motors

Purpose
The purpose of this test is to observe the performance and functionality of the servo motors under simulated depth conditions.

Materials/Equipment
- 9V Battery
- Arduino Nano
- Container of water
- GoPro camera
- GoPro housing
- LED flashlight
- Marine epoxy
- Mineral oil
- Plasti Dip
- Pressure apparatus
- Servo motors (several to test, one as control)
- Stopwatch
- Wires

Pressure Apparatus

Procedure
Setup/Calibrations
1. Waterproof the servo motors
   a. Disassemble the servo motor
   b. Coat the microchip inside the servo with marine epoxy
   c. Let the epoxy dry
   d. Submerge the servo motor components in a container of mineral oil
   e. Reassemble the servo while it is submerged to ensure no air is trapped within the servo
   f. Coat the servo motor with Plasti Dip to seal in the mineral oil
2. Write the code and program the Arduino
3. Assemble circuit as shown below
   a. Power the Arduino with the 9V battery
   b. Connect 5V, ground, and one digital pin from the Arduino to the servo motor
4. Gather the servo motor, test circuit, GoPro camera and housing, pressure apparatus, container of water, flashlight, and stopwatch
5. Ensure test rig is properly calibrated
6. Begin the stopwatch and GoPro recording at the same time (for post-analysis)
7. Place the GoPro camera in its pressure housing (see below)
8. Place the materials in the test rig as shown below
   a. Ensure that the servo is placed in the container of water and functioning
   b. Make sure the GoPro camera is positioned to record the servo motor

   Setup within the pressure apparatus

Test
1. Seal the test rig and begin pressurizing (see below)
2. Use the attached worksheet to record the time once a simulated depth of 5m (7psi) is achieved
3. Repeat previous step for simulated depths of 10m, 15m, 20m, …, 100m
4. Turn off and depressurize the test rig
5. Clean test station
6. Analyze servo motor performance by using the GoPro test footage and recorded times
7. Proceed to proper post-test procedure, below

   The sealed pressure chamber
Analysis
Failure criteria: (1) the servo motor loses all functionality at a simulated depth of less than 60m; (2) the servo motor behaves sluggishly or non-ideally at a simulated depth of less than 60m.

Post-test procedure if the test is successful:
1. Waterproof several (5-7) servo motors using the methods described above
2. Attach the waterproofed servo motors to the project vehicle
3. Check the condition of the servo motors after any vehicle use
4. Provide sponsor with documentation on how to waterproof and maintain the servo motors

Post-test procedure if the test is a failure:
1. Waterproof another servo motor more carefully
2. Test again
3. If second test fails, pursue other means of waterproofing

<table>
<thead>
<tr>
<th>Data Point</th>
<th>Depth (m)</th>
<th>Gage Pressure (kPa)</th>
<th>Gage Pressure (psi)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>49</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>98</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>147</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>196</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>245</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>294</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>343</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>392</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>45</td>
<td>441</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>491</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>55</td>
<td>540</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>60</td>
<td>589</td>
<td>85</td>
<td></td>
</tr>
</tbody>
</table>
Appendix E: Initial Buoyancy Test Plan
Test Date: April 24, 2013
Creator: Luke Stahler
Version: 1

Initial Buoyancy Test

Purpose
The purpose of this test is to determine how much weight needs to be added to the vehicle to make it neutrally buoyant.

Materials/Equipment
- Bathtub or pool
- Vehicle
- Weights (5-10 pounds total in 1 pound increments)

Procedure
Setup/Calibrations
9. Gather testing materials
10. Fill bathtub with water (if applicable)

Test
8. Submerge the vehicle in water
9. If the vehicle floats, add weight to the interior of the pressure housing
10. Repeat steps 1 and 2 until the vehicle sinks
11. Record how much additional weight is required to make the vehicle neutrally buoyant
12. Make sure to account for 2-4 pounds of adjustable weights

Post-Test Procedure
5. Add the required weight to the interior of the pressure housing
6. Manufacture the adjustable weights
7. Repeat test to ensure proper buoyancy
Appendix F: Manufacturing Drawings

As mentioned previously in the report, many of the project components were developed by Solid Concepts via additive manufacturing. They requested part model files rather than manufacturing drawings, so no drawings were developed for those components. However, drawings for the parts manufactured by the design team can be found below.

Figure 35: UHMW Vessel Manufacturing Drawing

Figure 35 is the drawing used to manufacture the UHMW pressure housing body. It is a simple drawing that calls out the inner diameter, outer diameter, and length of the vessel.
Figure 36: Housing Rod Manufacturing Drawing

Figure 36, above, shows the drawing used to manufacture the four aluminum-housing rods. It calls out the length and diameter of the rod, as well as the details of the tapped hole.

Figure 37: LED Holder Manufacturing Drawing
Finally, Figure 37 shows the manufacturing drawing of the LED holder that is currently being manufactured by the design team. It is a separate plastic module that will be attached to the bottom of the vehicle. The LEDs will provide ambient and cruising light for video and navigational purposes.
Appendix G: Project Code

NEMO Code: Onboard the Arduino

#include <Servo.h>
#include <OneWire.h>   //external temp library
#include <Wire.h>
#include <PID_v1.h>
#include <math.h>
#include <SignalFilter.h>
#include <LSM303.h>

//PID Set-up
SignalFilter Filter;
LSM303 compass;

#define SCALE 2  // accel full-scale, should be 2, 4, or 8
#define LSM303_ACC 0x18  // assuming SA0 grounded
#define LSM303_MAG 0x1E
#define X 0
#define Y 1
#define Z 2

/* LSM303 Register definitions */

#define CTRL_REG1_A 0x20
#define CTRL_REG2_A 0x21
#define CTRL_REG3_A 0x22
#define CTRL_REG4_A 0x23
#define CTRL_REG5_A 0x24
#define HP_FILTER_RESET_A 0x25
#define REFERENCE_A 0x26
#define STATUS_REG_A 0x27
#define OUT_X_L_A 0x28
#define OUT_X_H_A 0x29
#define OUT_Y_L_A 0x2A
#define OUT_Y_H_A 0x2B
#define OUT_Z_L_A 0x2C
#define OUT_Z_H_A 0x2D
#define INT1_CFG_A 0x30
#define INT1_SOURCE_A 0x31
#define INT1_THS_A 0x32
#define INT1_DURATION_A 0x33
#define CRA_REG_M 0x00
#define CRB_REG_M 0x01
#define MR_REG_M 0x02
#define OUT_X_H_M 0x03
#define OUT_X_L_M 0x04
#define OUT_Y_H_M 0x05
#define OUT_Y_L_M 0x06
#define OUT_Z_H_M 0x07
#define OUT_Z_L_M 0x08
#define SR_REG_M 0x09
#define IRA_REG_M 0x0A
#define IRB_REG_M 0x0B
#define IRC_REG_M 0x0C

//------------------------VARIABLE SETUP----------------------------------/
/*Global variables for PID*/
int accel[3]; // we'll store the raw acceleration values here
int mag[3]; // raw magnetometer values stored here
float realAccel[3]; // calculated acceleration values here

//Control Set Up
Servo thruster_1; // create servo object to control one of the thrusters
Servo servo_L; // create servo object to control one of the servos
Servo servo_R; // create servo object to control the other servo
Servo servo_T;

// I/O buffers
byte sub_write_buffer[7]; //buffer for outgoing data via hardware tx pin
byte controls[10]; //xyz control array
byte temp; //value used to look for the special character
byte temp2;

//Input servo positions
int Lposition;
int Rposition;
int Tposition;
int Thrust_val;
int ledPin = 13;
int innerLEDPin = 22;
int outerLEDPin = 23;
//Input Lights
int innerlights;
int outerlights;
//PID
int Loffset = 0; // offset to center the left servo
int Roffset = 11;       // offset to center the right servo
int requested_pitch = 0;  // pitch input from the computer
int req_roll = 0;
int InKi = 0;
int InKp = 0;
int InKd = 0;
int InState = 1;
int Tuned_Latch = 0;
byte checkByte = 0;
int servoVal = 0;

//PID Values

double current_roll;     // value from gyro

double requested_roll = 1.45;  //Set to maintain vertical orientation

double final_roll;      //Result of the PID (roll component of the control surfaces)

//PID Stuff

const int xpin = A0;       // x-axis of the accelerometer
const int ypin = A1;       // y-axis

double Filtered_Roll = 0;
double Final_Roll = 0;
double Requested_Roll = 0;  //during pid

float Kp = 4;
float Ki = 2;
float Kd = 0.8;
float pitch = 0;
float roll = 0;
double Roll_Correction = 0;

//PID myPID(&Smoothed_Roll, &Final_Roll, &Requested_Roll, Kp, Ki, Kd, DIRECT);

PID myPID(&Filtered_Roll, &Final_Roll, &Requested_Roll, Kp, Ki, Kd, DIRECT);

//SENSORS
OneWire  ds(52);  // on pin 10

void setup()
{
  //-----------------ATTACH SERVOS-----------------------------------------------//
thruster_1.attach(4, 1000, 2000); // attaches the thruster on pin 4 to the servo object
servo_L.attach(5); // attaches the servo on pin 5 to the servo object
servo_R.attach(6); // attaches the servo on pin 6 to the servo object
servo_T.attach(7);
Serial.begin(9600);
pinMode(ledPin, OUTPUT);
pinMode(innerLEDPin, OUTPUT);
pinMode(outerLEDPin, OUTPUT);

//------------------PID Stabilty--------------------------

  Filter.begin();
  Filter.setFilter('b');
  Filter.setOrder(2);
  myPID.SetMode(AUTOMATIC);
  //Serial.begin(9600);
  //Set auto tune stuff

  Wire.begin(); // Start up I2C, required for LSM303 communication for 2-axis compass reading
  initLSM303(SCALE); // Initialize the LSM303, using a SCALE full-scale range

  compass.init();
  compass.enableDefault();
  compass.m_min.x = -480; compass.m_min.y = -490; compass.m_min.z = +368;
  compass.m_max.x = 295; compass.m_max.y = +381; compass.m_max.z = +484;
}

void loop()
{
  //------------------EXTERNAL TEMPERATURE SENSOR-----------------------------
  /*
   byte i;
   byte present = 0;
   byte data[12];
   byte addr[8];
   int Temp;
   //int Tempf;
   if ( !ds.search(addr)) {
     //Serial.print("No more addresses.\n");
     ds.reset_search();
     return;
   }
  */
for (i = 0; i < 8; i++) {
}

if (OneWire::crc8(addr, 7) != addr[7]) {
    Serial.print("CRC is not valid!\n");
    return;
}

if (addr[0] != 0x28) {
    Serial.print("Device is not a DS18S20 family device.\n");
    return;
}

ds.reset();
ds.select(addr);
ds.write(0x44,1); // start conversion, with parasite power on at the end

delay(1000); // maybe 750ms is enough, maybe not
// we might do a ds.depower() here, but the reset will take care of it.

present = ds.reset();
ds.select(addr);
ds.write(0xBE); // Read Scratchpad

for (i = 0; i < 9; i++) { // we need 9 bytes
    data[i] = ds.read();
}
Temp=(data[1]<<8)+data[0];//take the two bytes from the response relating to temperature

Temp=Temp>>4;//divide by 16 to get pure celcius readout

//next line is Fahrenheit conversion
//Tempf=Temp*1.8+32; // comment this line out to get celcius

// Serial.print("Tf=");
//Serial.print(Tempf); //output farenheit integer to serial port
//Serial.print(" ");
//Serial.println();

// Serial.print("Tc=");
//Serial.print(Temp); //output the temperature to serial port
/**/

//----------HUMIDITY SENSOR----------------------------------------------//
int humidPin = 5;  //Output of the humidity sensor is read from the Analog 5 pin
int humid;  //temporary value to hold the current input
humid = analogRead(humidPin);  // reads the value of the humidity sensor (0 to 1023)
//Serial.print("Humidity=");  //Humidity (low values indicate higher humidity?)
//Serial.print(humid);  // hardware serial print - it will print to the Serial Monitor while connected via USB

//--------------Pressure/Depth SENSOR---------------------------------------------

/*int PressurePin = 7;  //Output of the pressure sensor is read from the Analog 7 pin
int pressure;  //temporary value to send up to controll code
pressure = analogRead(PressurePin);  // reads the value of the pressure sensor (0 to 1023)
//Serial.print("Pressure=");
//Serial.print(pressure);**/

//--------------COMPASS HEADING-----------------------------------------------
compass.read();
int heading = compass.heading((LSM303::vector){0,-1,0});
delay(100);

//--------------ROLL---------------------------------------------------------
getLSM303_accel(accel);  // get the acceleration values and store them in the accel array
while(!(LSM303_read(SR_REG_M) & 0x01));  // wait for the magnetometer readings to be ready
getLSM303_mag(mag);  // get the magnetometer values, store them in mag
for (int i=0; i<3; i++)
realAccel[i] = accel[i] / pow(2, 15) * SCALE;  // calculate real acceleration values, in units of g
//Serial.print(getTiltHeading(mag, realAccel), 3);  // get tilt compensated heading

//--------------DATA TRANSMISSION-----------------------------------------
Serial.print('$');  //Begining of data transmittion string that control code looks for
//Serial.print(Temp+100);  //output the temperature to serial port (add 100 to send constant bit stream that control code can identify
//Serial.print("Humidity=");
Serial.print(humid+10000);  //Humidity sensor datat sent to control code (+10000 for constant number of bits sent up)will be an integer 0-1023
//Serial.print(" Heading=");
//Serial.print(pressure+10000);    //Pressure reading(+10000 for constant number of
bits sent up) will be an integer 0-1023
//Serial.print("Pressure=");
//Serial.print((getTiltHeading(mag, realAccel)+1000), 0);    // get tilt compensated
heading
Serial.print(heading+1105);    // Get 2-axis compass reading and send to control code
delay(1200);   // delay for serial readability

/**/

--------------------------DATA RECEPTION/THRUSTER SERVO CONTROL------------------------
---------------------------

if (Serial.available() >= 6) // Wait for at least 6 bytes to be available (full package)
{
    for(int i=0; i<6; i++)
    {
        controls[i] = Serial.read();  // read in data
    }
}

Thrust_val = controls[0];
Rposition = controls[1];
Lposition = controls[2];
Tposition = controls[3];
innerlights = controls[4];
outerlights = controls[5];

//---------------------LEDS--------------------------------------------------
//need to map from 0-100% to 0-255 for analog write
innerlights = map(innerlights, 0, 100, 0, 255);
outerlights = map(outerlights, 0, 100, 0, 255);
analogWrite(innerLEDPin, innerlights);
analogWrite(outerLEDPin, outerlights);

//------------------WRITE SERVO POSITIONS---------------------------------------/
thruster_1.write(Thrust_val);
servo_T.write(Tposition);
servo_L.write(Lposition);
servo_R.write(Rposition);
//--------------PID Stability---------------------
getLSM303_accel(accel); // get the acceleration values and store them in the accel array
for (int i=0; i<3; i++)
    realAccel[i] = accel[i] / pow(2, 15) * SCALE; // calculate real acceleration values, in units of g
getFilteredRoll(realAccel);

Tposition = Tposition;
requested_pitch = 90 - Lposition;

if (InState == 0){  //No PID
    Roll_Correction = 0;  //clear out PID factor
    getLRPosition();
    writeServoVals();
}
else if (InState == 1){  //Try Values
    if (Kp != (InKp/10)){
        Kp = InKp/10;
        Ki = InKi/10;
        Kd = InKd/10;
        myPID.SetTunings(Kp, Ki, Kd);
    }
    getRollCorrectionFactor();
    getLRPosition();
    writeServoVals();
}

//---------------------------------------------

void getFilteredRoll(float * accelValue)
{
    float pitch = asin(-accelValue[X])+0.06;
    float roll = asin(accelValue[Y]/cos(pitch));
    pitch = pitch - 0.1;
    Filtered_Roll = Filter.run(1000*(pitch));
    Filtered_Roll = Filtered_Roll/1000;
    if (Filtered_Roll <= 0.02 && Filtered_Roll >= -0.02){
        Filtered_Roll = 0;
    }
}
void getLRPosition()
{
    Lposition = Loffset+90+(Roll_Correction-requested_pitch)/2;
    if (Lposition >= 155){
        Lposition = 155;
    }
    else if (Lposition <= 25){
        Lposition = 25;
    }
    Rposition = Roffset+90+(Roll_Correction+requested_pitch)/2;
    if (Rposition >= 155){
        Rposition = 155;
    }
    else if (Rposition <= 25){
        Rposition = 25;
    }
}

void writeServoVals()
{
    thruster_1.write(Thrust_val);
    servo_T.write(Tposition);
    servo_L.write(Lposition);
    servo_R.write(Rposition);
}

void getRollCorrectionFactor()
{
    myPID.Compute();
    Roll_Correction = Final_Roll*60;   //Note: Final_Roll is the output of the PID
}

void initLSM303(int fs){
    LSM303_write(0x27, CTRL_REG1_A);  // 0x27 = normal power mode, all accel axes on
    if ((fs==8)||(fs==4))
        LSM303_write((0x00 | (fs-fs/2-1)<<4), CTRL_REG4_A);  // set full-scale
    else
        LSM303_write(0x00, CTRL_REG4_A);
    LSM303_write(0x14, CRA_REG_M);  // 0x14 = mag 30Hz output rate
    LSM303_write(0x00, MR_REG_M);  // 0x00 = continouous conversion mode
}
void getLSM303_accel(int * rawValues){
  rawValues[Z] = ((int)LSM303_read(OUT_X_L_A) << 8) | (LSM303_read(OUT_X_H_A));
  rawValues[X] = ((int)LSM303_read(OUT_Y_L_A) << 8) | (LSM303_read(OUT_Y_H_A));
  rawValues[Y] = ((int)LSM303_read(OUT_Z_L_A) << 8) | (LSM303_read(OUT_Z_H_A));
  // had to swap those to right the data with the proper axis
}

byte LSM303_read(byte address){
  byte temp;
  if (address >= 0x20)
    Wire.beginTransmission(LSM303_ACC);
  else
    Wire.beginTransmission(LSM303_MAG);
  Wire.write(address);
  if (address >= 0x20)
    Wire.requestFrom(LSM303_ACC, 1);
  else
    Wire.requestFrom(LSM303_MAG, 1);
  while(!Wire.available())
    ;
  temp = Wire.read();
  Wire.endTransmission();
  return temp;
}

void LSM303_write(byte data, byte address)
{
  if (address >= 0x20)
    Wire.beginTransmission(LSM303_ACC);
  else
    Wire.beginTransmission(LSM303_MAG);

  Wire.write(data);
  Wire.endTransmission();
}
float getTiltHeading(int * magValue, float * accelValue)
{
    // see appendix A in app note AN3192
    float pitch = asin(-accelValue[X]);
    float roll = asin(accelValue[Y]/cos(pitch));
    float xh = magValue[X] * cos(pitch) + magValue[Z] * sin(pitch);
    float yh = magValue[X] * sin(roll) * sin(pitch) + magValue[Y] * cos(roll) -
               magValue[Z] * sin(roll) * cos(pitch);
    float zh = -magValue[X] * cos(roll) * sin(pitch) + magValue[Y] * sin(roll) +
              magValue[Z] * cos(roll) * cos(pitch);
    float heading = 180 * atan2(yh, xh)/PI;
    if (yh >= 0)
        return heading;
    else
        return (360 + heading);
}

/////2 axis-compass reading
//float getHeading(int * magValue)
//{
//    // see section 1.2 in app note AN3192
//    float heading = 180*atan2(magValue[Y], magValue[X])/PI;  // assume pitch, roll are 0
//    //
//    // if (heading <0)
//    //    heading += 360;
//    //
//    // return heading;
//}

void getLSM303_mag(int * rawValues)
{
    Wire.beginTransmission(LSM303_MAG);
    Wire.write(OUT_X_H_M);
    Wire.endTransmission();
    Wire.requestFrom(LSM303_MAG, 6);
    for (int i=0; i<3; i++)
        rawValues[i] = (Wire.read() << 8) | Wire.read();
}/**/
#ifndef _XBOX_CONTROLLER_H_
define _XBOX_CONTROLLER_H_

// No MFC
#define WIN32_LEAN_AND_MEAN

// We need the Windows Header and the XInput Header
#include <windows.h>
#include <XInput.h>

// Now, the XInput Library
// NOTE: COMMENT THIS OUT IF YOU ARE NOT USING A COMPILER THAT SUPPORTS THIS
// METHOD OF LINKING LIBRARIES
#pragma comment(lib, "XInput.lib")

// XBOX Controller Class Definition
class CXBOXController
{
private:
    XINPUT_STATE _controllerState;
    int _controllerNum;
public:
    CXBOXController(int playerNumber);
    XINPUT_STATE GetState();
    bool IsConnected();
    void Vibrate(int leftVal = 0, int rightVal = 0);
};

#endif
class Serial
{
    private:
    HANDLE hSerial; //Serial comm handler
    bool connected; //Connection status
    COMSTAT status; //Get various information about the connection
    DWORD errors; //Keep track of last error

    public:
    Serial(char *portName); //Initialize Serial communication with the given COM port
    ~Serial(); //Close the connection
    bool IsConnected(); //NOTA: for some reason you can't connect again before exiting
    //the program and running it again
    int ReadData(char *buffer, unsigned int nbChar); //Read data in a buffer, if nbChar is greater than the
    //maximum number of bytes available, it will return only the
    //bytes available. The function return -1 when nothing could
    //be read, the number of bytes actually read.
    bool WriteData(int *buffer, unsigned int nbChar); //Writes data from a buffer through the Serial connection
    //return true on success.

};

#endif // SERIALCLASS_H_INCLUDED
CXBOXController.cpp (taken from GitHub)
#include "CXBOXController.h"

CXBOXController::CXBOXController(int playerNumber)
{
    // Set the Controller Number
    _controllerNum = playerNumber - 1;
}

XINPUT_STATE CXBOXController::GetState()
{
    // Zeroise the state
    ZeroMemory(&_controllerState, sizeof(XINPUT_STATE));

    // Get the state
    XInputGetState(_controllerNum, &_controllerState);
    return _controllerState;
}

bool CXBOXController::IsConnected()
{
    // Zeroise the state
    ZeroMemory(&_controllerState, sizeof(XINPUT_STATE));

    // Get the state
    DWORD Result = XInputGetState(_controllerNum, &_controllerState);

    if (Result == ERROR_SUCCESS)
    {
        return true;
    }
    else
    {
        return false;
    }
}

void CXBOXController::Vibrate(int leftVal, int rightVal)
{
    // Create a Vibraton State
    XINPUT_VIBRATION Vibration;

    // Zeroise the Vibration
    ZeroMemory(&Vibration, sizeof(XINPUT_VIBRATION));

    // Set the Vibration Values
    Vibration.wLeftMotorSpeed = leftVal;
    Vibration.wRightMotorSpeed = rightVal;

    // Vibrate the controller
    XInputSetState(_controllerNum, &Vibration);
}
Serial.cpp (taken from Microsoft website)

```cpp
#include "SerialClass.h"

Serial::Serial(char *portName)
{
    // We're not yet connected
    this->connected = false;

    // Try to connect to the given port through CreateFile
    this->hSerial = CreateFileA( static_cast<LPCSTR>(portName),
                                   GENERIC_READ | GENERIC_WRITE,
                                   0,
                                   NULL,
                                   OPEN_EXISTING,
                                   FILE_ATTRIBUTE_NORMAL,
                                   NULL);

    // Check if the connection was successful
    if (this->hSerial == INVALID_HANDLE_VALUE)
    {
        // If not successful display an Error
        if (GetLastError() == ERROR_FILE_NOT_FOUND)
        {
            // Print Error if necessary
            printf("ERROR: Handle was not attached. Reason: %s not available.\n", portName);
        }
        else
        {
            printf("ERROR!!!");
        }
    }
    else
    {
        // If connected we try to set the comm parameters
        DCB dcbSerialParams = {0};

        // Try to get the current
        if (!GetCommState(this->hSerial, &dcbSerialParams))
        {
            // If impossible, show an error
            printf("failed to get current serial parameters!");
        }
        else
        {
            // Define serial connection parameters for the arduino board
            dcbSerialParams.BaudRate=CBR_9600;
            dcbSerialParams.ByteSize=8;
            dcbSerialParams.StopBits=ONESTOPBIT;
            dcbSerialParams.Parity=NOPARITY;

            // Set the parameters and check for their proper application
            if (!SetCommState(hSerial, &dcbSerialParams))
            {
                printf("ALERT: Could not set Serial Port parameters");
            }
            else
```
{ //If everything went fine we're connected
this->connected = true;
//We wait 2s as the arduino board will be reseting
Sleep(ARDUINO_WAIT_TIME);
}
}
}

Serial::Serial()
{
    //Check if we are connected before trying to disconnect
    if(this->connected)
    {
        //We're no longer connected
        this->connected = false;
        //Close the serial handler
        CloseHandle(this->hSerial);
    }
}

int Serial::ReadData(char *buffer, unsigned int nbChar)
{
    //Number of bytes we'll have read
    DWORD bytesRead;
    //Number of bytes we'll really ask to read
    unsigned int toRead;

    //Use the ClearCommError function to get status info on the Serial port
    ClearCommError(this->hSerial, &this->errors, &this->status);

    //Check if there is something to read
    if(this->status.cbInQue>0)
    {
        //If there is we check if there is enough data to read the required number
        //of characters, if not we'll read only the available characters to
        prevent
        //locking of the application.
        if(this->status.cbInQue>nbChar)
        {
            toRead = nbChar;
        }
        else
        {
            toRead = this->status.cbInQue;
        }

        //Try to read the require number of chars, and return the number of read
        bytes on success
        if(ReadFile(this->hSerial, buffer, toRead, &bytesRead, NULL) &&
         bytesRead != 0)
        {
            return bytesRead;
        }
    }
}
// If nothing has been read, or that an error was detected return -1
return -1;
}

bool Serial::WriteData(int *buffer, unsigned int nbChar)
{
    DWORD bytesSend;
    // Try to write the buffer on the Serial port
    if(!WriteFile(this->hSerial, (void *)buffer, nbChar, &bytesSend, 0))
    {
        // In case it don't work get comm error and return false
        ClearCommError(this->hSerial, &this->errors, &this->status);
        return false;
    }
    else
    { return true; }
}

bool Serial::IsConnected()
{
    // Simply return the connection status
    return this->connected;
}
Main.cpp

/*Main.cpp created by Sarah Romer with the help of //
/* Jacob Karas for the control and interface with ///////
/* NEMO, open source, national geographic project*/////

#include "CXBOXController.h"
#include "SerialClass.h"
#include <iostream>
#include <string>
#include <cstring>
#include <math.h>
#include "CXBOXController.h"
#include "SerialClass.h"
#include <iostream>
#include <stdio.h>
#include <string>
#include <cstring>
#include <math.h>
#include <time.h>
#include "cv.hpp"
#include "cv.h"
#include "cxcore.hpp"
#include "highgui.h"

using namespace cv;

#define SERVOMAX 135
#define SERVOMIN 45
#define THRUSTFORWARD 100
#define THRUSTREVERSE 60
#define THRUSTFORWARDMAX 140
// MAX and MIN are for extra power when user also holds down right shoulder. Not standard
#define THRUSTREVERSEMIN 40
#include <windows.h>
#include <tchar.h>
#include <strsafe.h>

#define BUF_SIZE 255
#define NUMBER_OF_THREADS 1
#define CAMERA 0

DWORD WINAPI runController(LPVOID lpParam); // code that runs the controller

CXBOXController* Player1;

int main(int argc, char* argv[])  
// int doStuff(int argc, char* argv[])  
{
char* arduinoPortName = new char[5]; //Port COM3, Selects port to send serial data to for the Arduino Nano in port COM4
arduinoPortName = "COM5";

Serial* arduino = new Serial(arduinoPortName); //Port COM5 passed into serial, pointer to arduino. COM5 is the fiber transciever????

char arduinoread[10];

DWORD threadIDArray[NUMBER_OF_THREADS];
HANDLE hThreadArray[NUMBER_OF_THREADS];

// Create the thread to begin execution of code.

hThreadArray[0] = CreateThread(
    NULL, // default security attributes
    0, // use default stack size
    runController, // thread function name
    arduino, // argument to thread function
    0, // use default creation flags
    &threadIDArray[0]); // returns the thread identifier

//-------------------Camera Frame Set-Up

int x = 1340;
int y = 770;
CvCapture* capture = cvCreateCameraCapture(CAMERA);
cvNamedWindow("NEMO", 0);
cvResizeWindow("NEMO",x,y);
cvMoveWindow("NEMO",0,0);
clock_t init, final;

//-------------------timer functionality
init = clock();
final = clock()-init;
int secsRunning = 0;
char headingDir[3];
//char buffer[70] = "";
long int extTemp = 0;
long double humidity = 00.0;
long int headingNum = 0;
long int depth = 0;
while(1) //WHILE LOOP
{
    final = clock()-init;
    //std::cout << (double)final / ((double)CLOCKS_PER_SEC) << std::endl;
//std::cout<<"here"<<std::endl;
secsRunning = (int)((double)final / ((double)CLOCKS_PER_SEC));

--------------------------
Grabs frames for video
CvArr* frame = cvQueryFrame( capture );
if( !frame )
{
 break;
}

CvPoint *org = new CvPoint();
org->x = 5;
org->y = 20;
CvScalar color = cvScalar(0,0,0);
CvScalar color2 = cvScalar(225,225,225);
CvScalar red = cvScalar(0,0,225);

CvPoint *org2 = new CvPoint();
org2->x = 10;
org2->y = 50;

------------------------
Sensor Read In and Initializes variables for textual display

arduino->ReadData(arduinoread, 1);
if(arduinoread[0]=='$')
{ //begining of handshaking to identify senor data ready for read
  //arduino->ReadData(arduinoread, 3);
  //external temperature read from arduino (3 characters)
  //arduinoread[3] = '\0';
  //double extTemp = 5.7;
  //extTemp = (atoi(arduinoread)-100);
  extTemp = 20;

  arduino->ReadData(arduinoread, 5);
  //humidity sensor read from arduino (5 characters)
  ardinoread[5] = '\0';
  humidity = ((atof(arduinoread)-1000)/1023.0) * 100.0;

  arduino->ReadData(arduinoread, 5);
  //Pressure sensor read in from arduino (5 characters)
  //arduinoread[5] = '\0';
  //depth = (atoi(arduinoread)-10000);
  //this is where the pressure to depth conversion calculation will go
depth = 8.0;
  //std::cout<< "this is depth "<<arduinoread<<std::endl;

  arduino->ReadData(arduinoread, 4);
  //Compass read in from arduino (4 characters)
  arduinoread[4] = '\0';
  headingNum = (atof(arduinoread)-1000);
  headingNum = headingNum%360;
if(headingNum > 0 & headingNum <= 22)
{
    headingDir[0] = 'N';
    headingDir[1] = ' ';  
}
else if (headingNum >=23 & headingNum <= 67)
{
    headingDir[0] = 'N';
    headingDir[1] = 'E';
}
else if (headingNum >=68 & headingNum <= 112)
{
    headingDir[0] = 'E';
    headingDir[1] = ' '; 
}
else if (headingNum >=113 & headingNum <= 157)
{
    headingDir[0] = 'S';
    headingDir[1] = 'E';
}
else if (headingNum >=158 & headingNum <= 202)
{
    headingDir[0] = 'S';
    headingDir[1] = 'W';
}
else if (headingNum >=203 & headingNum <= 247)
{
    headingDir[0] = 'W';
    headingDir[1] = ' '; 
}
else if (headingNum >=248 & headingNum <= 292)
{
    headingDir[0] = 'W';
    headingDir[1] = 'N';  
}
else if (headingNum >=293 & headingNum <= 337)
{
    headingDir[0] = 'N';
    headingDir[1] = 'W';
}
else if (headingNum >=338 & headingNum <= 359)
{
    headingDir[0] = 'N';
    headingDir[1] = ' ';  
}
else
{
    headingDir[0] = '?';
    headingDir[1] = '?';
}

//-------------------------------Displays Text onto video image
}
char buffer[70] = "";
//strcat(buffer, arduinoread);
sprintf(buffer, " HEADING: %3d %c%c   EXT.TEMP: %3d C   DEPTH: %3d m   OP TIME: %d:%02d:%02d ", headingNum, headingDir[0], headingDir[1], extTemp, depth, (secsRunning/60)/60, (secsRunning/60)%60, secsRunning%60);

    char humiditytext[30] = "Warning! Humidity is high";

CvFont font;
double textSize = 1.0;
double textThickness = 3.0;
int f = 1;
cvInitFont(&font, f, textSize, textSize,0.0, 3.0, 8);
cvPutText(frame,  buffer ,  *org, &font, color);
cvInitFont(&font, f, textSize, textSize,0.0, .25, 8);
cvPutText(frame,  buffer ,  *org, &font, color2);

if (humidity >= 80) {
    cvInitFont(&font, f, textSize, textSize,0.0, 3.0, 8);
cvPutText(frame,  humiditytext ,  *org2, &font, color);
cvInitFont(&font, f, textSize, textSize,0.0, .25, 8);
cvPutText(frame,  humiditytext ,  *org2, &font, red);
std::cout<<"ahh!!\n";
}

cvResizeWindow("NEMO",x,y);
cvShowImage( "NEMO", frame);

    char c = cvWaitKey(33);
if( c == 27 )
{
    break;
}/**/

// Check the return value for success.
// If CreateThread fails, terminate execution.
// This will automatically clean up threads and memory.
for(int i = 0; i<NUMBER_OF_THREADS; i++)
{
    if (hThreadArray[i] == NULL)
    {
        std::cout<<"CreateThread has failed";
        ExitProcess(3);
    }
}

// Wait until all threads have terminated.
WaitForMultipleObjects(NUMBER_OF_THREADS, hThreadArray, TRUE, INFINITE);

// Close all thread handles and free memory allocations.
for(int i = 0; i<NUMBER_OF_THREADS; i++)
{
    CloseHandle(hThreadArray[i]);
}/**/

return 0;
}

//-------------------
Controller Input and Output
DWORD WINAPI runController(LPVOID lpParam)
{
    //Variables: More will be added later as lighting and modular capabilities are achieved
    int ServoT=90;       //should be set to the null value for each
    int ServoR=90;
    int ServoL=90;
    int Thruster=0;
    int Buttons=0;
    int InnerLights=0;
    int OuterLights=0;

    Player1 = new CXBOXController(1);
    //string port="COM3";
    char*portName=new char[5];            //Selects port to send serial data to for the Arduino Nano in port COM4
    portName = "COM4";

    //char* arduinoPortName=new char[5];     //Port COM3, Selects port to send serial data to for the Arduino Nano in port COM4
    //arduinoPortName = "COM3";
    //Serial* arduino = new Serial(arduinoPortName);
    Serial* arduino =(Serial*) lpParam ;     //Port COM5 passed into serial, pointer to arduino. COME5 to fiber transciever

    int hold = 0;
    int* arduinowrite = &hold;
    bool lightsDown = false;
    while(true)
    {

        if(Player1->IsConnected())
        {
            Thruster=0;
            ServoR=90;
            ServoL=90;
            ServoT=90;

            //Thruster Control:Linear

            if(Player1->GetState().Gamepad.wButtons &
XINPUT_GAMEPAD_RIGHT_SHOULDER)
            {

            

}
Thruster=(Player1->GetState().Gamepad.sThumbLY*((THRUSTFORWARDMAX-THRUSTREVERSEMIN)/2) / 32767) +90;

} 
else
{
Thruster=(Player1->GetState().Gamepad.sThumbLY*((THRUSTFORWARD-THRUSTREVERSE)/2) / 32767) +90;
}/**/ 

//Thruster Control: Exponential //Removed

/*if(Player1->GetState().Gamepad.wButtons & XINPUT_GAMEPAD_RIGHT_SHOULDER)
{
    int Thrustexp=(Player1->GetState().Gamepad.sThumbLY*((THRUSTFORWARDMAX-THRUSTREVERSEMIN)/2) / 32767);
    bool Boo=0;
    if (Thrustexp <= 0 )
    {
        Boo=1;
    }
    Thrustexp = Thrustexp * Thrustexp;
    Thruster= Thrustexp/((THRUSTFORWARDMAX - THRUSTREVERSEMIN)/2);
    if (Boo==1)
    {
        Thruster = Thruster * -1;
    }
    Thruster = Thruster +90;
    std::cout<<Thruster<<"\n";
}
else
{
    int Thrustexp=(Player1->GetState().Gamepad.sThumbLY*((THRUSTFORWARD-THRUSTREVERSE)/2) / 32767);
    bool Boo=0;
    if (Thrustexp <= 0 )
    {
        Boo=1;
    }
    Thrustexp = Thrustexp * Thrustexp;
    Thruster= Thrustexp/((THRUSTFORWARD - THRUSTREVERSE)/2);
    if (Boo==1)
    {
        Thruster = Thruster * -1;
    }
    Thruster = Thruster +90;
    std::cout<<Thruster<<"\n";
}/**/
// Servo Controls

if (Player1->GetState().Gamepad.sThumbRY != 0)
    // Pitch of the UROV (Nose goes up or down)
    {
        // pitch
        ServoL = (Player1->GetState().Gamepad.sThumbRY
            * ((SERVOMAX - SERVOMIN) / 2) / 32767) + 90;  // servos here move together
        ServoR = -(Player1->GetState().Gamepad.sThumbRY
            * ((SERVOMAX - SERVOMIN) / 2) / 32767) + 90;  // servos here move together
    }

if (Player1->GetState().Gamepad.sThumbRX != 0)
    // Yaw of UROV (turning left or right)
    {
        // turn to top servo
        ServoT = -(Player1->GetState().Gamepad.sThumbRX
            * ((SERVOMAX - SERVOMIN) / 2) / 32767) + 90;
    }

// Buttons-------------------------Lights

if (Player1->GetState().Gamepad.wButtons != 0)
    {
        if (!lightsDown)
            {
                lightsDown = true;
                if (Player1->GetState().Gamepad.wButtons == 1 & InnerLights != 100)
                    {
                        InnerLights = (InnerLights + 20);
                    }
                else if (Player1->GetState().Gamepad.wButtons == 2 & InnerLights > 0)
                    {
                        InnerLights = InnerLights - 20;
                    }
                else if (Player1->GetState().Gamepad.wButtons == 8 & OuterLights != 100)
                    {
                        OuterLights = OuterLights + 20;
                    }
                else if (Player1->GetState().Gamepad.wButtons == 4 & OuterLights > 0)
                    {
                        OuterLights = OuterLights - 20;
                    }
                }
    }
else
{
    lightsDown = false;
}

//std::cout<<Player1->GetState().Gamepad.wButtons<<"\n";
//std::cout<<Player1->GetState().Gamepad.bLeftTrigger;
std::cout<<Player1->GetState().Gamepad.bRightTrigger;
std::cout<<Player1->GetState().Gamepad.sThumbLX;
std::cout<<Player1->GetState().Gamepad.sThumbLY;
std::cout<<Player1->GetState().Gamepad.sThumbRX;
std::cout<<Player1->GetState().Gamepad.sThumbRY;
std::cout<<"\n";/**/

if(Player1->GetState().Gamepad.wButtons & XINPUT_GAMEPAD_BACK)
/*Exits Program/**/
{
    break;
}
}
else
{
    std::cout << "\n\tERROR! PLAYER 1 - XBOX 360 Controller Not Found!\n";
    std::cout << "Press Any Key To Exit."
    std::cin.get();
    break;
}

//Data Send: This is what sends the data to the arduino. IT MUST STAY IN THIS ORDER
//std::cout<<hold<<"\n";
hold = Thruster;
arquivo->WriteData(arduinowrite, (unsigned int) 1);
//std::cout<<hold<<"\n";
hold = ServoR;
arquivo->WriteData(arduinowrite, (unsigned int) 1);
//std::cout<<hold<<"\n";
hold = ServoL;
arquivo->WriteData(arduinowrite, (unsigned int) 1);
//std::cout<<hold<<"\n";
hold = ServoT;
arquivo->WriteData(arduinowrite, (unsigned int) 1);
//std::cout<<hold<<"\n";
hold = InnerLights;
arquivo->WriteData(arduinowrite, (unsigned int) 1);
//std::cout<<hold<<"\n";
hold = OuterLights;
arquivo->WriteData(arduinowrite, (unsigned int) 1);
//std::cout<<hold<<"\n";
//hold = Buttons;
arquivo->WriteData(arduinowrite, (unsigned int) 1);
//std::cout<<hold<<"\n";

} //while loops end bracket
```cpp
delete(Player1);
return 0;
}
```

**Note**

The code has yet to be ‘cleaned up’ due to the need for a final integrated test. Many of the lines commented out are for ease of testing while some are customizable options like having an exponential mapping with thruster power instead of a linear mapping. It remains in the code but commented out so that a user can easily input it into the code.